HOSPITAL MANAGEMENT SYSTEM

A Project work submitted to the

DEPARTMENT OF COMPUTER APPLICATIONS
P.K.N. ARTS AND SCIENCE COLLEGE
THIRUMANGALAM.

Submitted by

E.ARUL BENJAMIN CHANDRU (Reg No : A7108624)
R.BALAJI (Reg No : A7108625)
BCA-IIIrd year
P.K.N. Arts and Science College
Thirumangalam.

Guided by

Mrs.V.PANDISELVI M.Sc.,M.Phil.,
Head of the Department
Department Of computer Applications
P.K.N. Arts and Science College
Thirumangalam

DEPARTMENT OF COMPUTER APPLICATIONS
P.K.N. ARTS AND SCIENCE COLLEGE
(Affiliated to Madurai Kamaraj University)
Thirumangalam.

DEPARTMENT OF COMPUTER APPLICATIONS
P.K.N. ARTS AND SCIENCE COLLEGE
THIRUMANGALAM

BONAFIDE CERTIFICATE

Certified that the project work entitled **" HOSPITAL MANAGEMENT SYSTEM"** submitted by **E.ARUL BENJAMIN CHANDRU** (Reg No : **A7108624**) to the Department of Computer Applications, P.K.N. Arts and Science College, under the guidance of **Mrs.V.PANDISELVI, M.Sc, M.Phil., Head of the Department** in partial fulfillment of the requirements for the Under Graduate degree in Bachelor of Computer Applications.

Mrs.V.PANDISELVI M.Sc, M.Phil.,	Mrs.V.PANDISELVI M.Sc, M.Phil.,
PROJECT GUIDE	**HEAD OF THE DEPARTMENT**

Submitted for viva voice examination held on __________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DEPARTMENT OF COMPUTER APPLICATIONS
P.K.N. ARTS AND SCIENCE COLLEGE
THIRUMANGALAM

BONAFIDE CERTIFICATE

Certified that the project work entitled **" HOSPITAL MANAGEMENT SYSTEM"** submitted by **R.BALAJI** (Reg no : **A7108625**) to the Department of Computer Applications, P.K.N. Arts and Science College, under the guidance of **Mrs.V.PANDISELVI, M.Sc, M.Phil., Head of the Department** in partial fulfillment of the requirements for the Under Graduate degree in Bachelor of Computer Applications.

Mrs.V.PANDISELVI M.Sc, M.Phil.,	Mrs.V.PANDISELVI M.Sc, M.Phil.,
PROJECT GUIDE	**HEAD OF THE DEPARTMENT**

Submitted for viva voice examination held on __________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DEPARTMENT OF COMPUTER APPLICATIONS
P.K.N. ARTS AND SCIENCE COLLEGE
THIRUMANGALAM

DECLARATION

We hereby declare that this work was carried out by us under the guidance and Supervision of **Mrs.V.PANDISELVI M.Sc.,M.Phil., Head of the Department**. The period of project work is From **N**ovember 2009 To April 2010. This project work is submitted to P.K.N. Arts and Science College, Department of Computer Applications in partial fulfillment for the requirement of Degree in Bachelor of Computer Applications.

We declare that this work has not been submitted anywhere else for the award of any other degree.

E.ARUL BENJAMIN CHANDRU **R.BALAJI**

ACKNOWLEDGEMENT

First of all We would like to express our heartful thanks to " The Almighty God" for this opportunity, which he rendered to us and gives the physical strength and pleasant mind to complete this project work.

We thank our honorable Principal **Dr.J. MAHATMAN RAO, Ph.D,** for his inspiration. And we thank **Mrs.V.PANDISELVI M.Sc., M.Phil,** our Head of the Department and also my project guide for the correct guidance and advice.

We also thank our department staff

- **Mr.A.Balaji , MS(IT&M), M.Phil.,**
- **Ms.D.Murugasundari , M.Sc(IT&CS).,**
- **Mr.R.Prabakaran, MCA.,**
- **Mr.K.Janakaraj, MCA.,**

who all encourage and satisfy our needs to finish this project work.

We are very happy to thank our lab co-ordinator, lab assistants for giving a well equipped lab for developing this project work.

We extent our thanks and gratitude to our parents, Friends and those who helped us directly and indirectly for the successful completion of this project work.

CONTENTS

INTRODUCTION

SYNOPSIS

This project will automate the daily operations of LIFE LINE hospital. The project keeps track of the staff and patient (in-patient, out-patient) details. It also takes care of the ward, medical, invoice and the doctor’s appointment details. The system generates the daily ward availability, the status of the operation theatres and ICU.

HOSPITAL MANAGEMENT is an integrated Hospital Information System, which addresses all the major functional areas of multi-specialty hospitals. The HOSPITAL MANAGEMENT enables better patient care, patient safety, patient confidentiality, efficiency, reduced costs and better management information system. It provides easy access to critical information thus enabling the management to take better decisions on time.

This project deals with processing of each and every department in the hospital. This project sincerely aims to reduce the manual processing of each department.

The Scope of the project takes care of the details of each and every department. These details gives the doctor, staffs, specialists and patient details including their salary, attendance , doctor’s appointments and the billing system. The details of Doctor and staff help the hospital to maintain the record of every person. Their attendance details help them to know about their attentive presence while salary is calculated. The billing system provides an efficient way for calculating bill details of the patients.

COMPANY PROFILE

Vinayagam Hospitals is a multi-specialty, tertiary care hospital extending quality health care service to people in and around Madurai for the past 12 years. The hospital was founded by Dr.S.Rajendran who is a leading Laser and Laparoscopic surgeon. The hospital is located at Dhanappa Mudali Street which is very near to Madurai Meenakshi Amman Temple. Its head office is located at Royapettai, Chennai. Its motive is to "Reach the Unreached".

History of the Vinayagam Hospitals:

Vinayagam Hospital was first started at 1998 with ten staff and five Beds for inpatients. In the beginning it started providing services to the outpatients and inpatients.

In the year 2000, the hospital extends its services such as X-Ray and Scan Centre. After that, the hospital developed rapidly. In the year 2005 Laser Cancer Treatment Facility was started by the famous leading tamil actor Surya who is the relative to Dr.S.Rajendran. Last year the hospital is well furnished and it provides the following facilities.

- Capsule Endoscopy
- Endoscopy (Diagnostic & Therapeutic)
- Colonoscopy
- Sigmoidoscopy
- Bronchoscopy
- Laser Surgery (ND,YAG)
- Laparoscopy (Key Hole) Surgery
- General Surgeries
- Ultra Sound and Color Doppler (Scan & ECHO)

- Dental Care
- Psychiatric Treatments

Now the hospital is famous for its laser and laparoscopic surgeries. The hospital now consists of 12 specialists , 4 duty doctors and 40 staff members. It also provides ambulance services to the patients. The hospital is functionally available to the public 24 hours a day.

The hospital also provides star health insurance plan for government employees and also kalaignar kappeettu thittam. The hospital is rapidly developing towards the motive " Service to the human is the service to the God ".

In future, the hospital have plans to implement the following programmes.

- Identifying Anemia in rural Women and Children
- Preventive health care in rural areas.

PROJECT OBJECTIVE

PROJECT OBJECTIVE

- To computerize all details regarding patient details & hospital details.
- To automate the process of ward entries.
- To maintain records effectively.
- To manage current status of staff and doctor availablity.
- The project has information regarding the inpatient details, outpatient details, Billing details and Ambulance details.

This project includes modules such as

1) Admission
   - Inpatient
   - Outpatient
2) Staff Details
   - Payroll
   - Personnel Details
   - Attendance
   - On Duty
   - Shift
3) Billing
   - Inpatient
   - Outpatient
4) Consultation

5) Ward Details

6) Ambulance Service
   - Driver
   - Ambulance

MODULES REQUIRED

Admission :

This module records basic patient related information, which is collected when the patient visits the hospital for the first time. Each patient is allocated a unique patient identification numbers also known as Hospital No.

Patient Details:

It keeps track of all details about both in-patient and out-patient. Patient id, patient name, address, admitted date, doctor name, room no are entered in a form and stored for future reference. Also particular patient details can be viewed in the table using a separate form with a attribute patient id.

Outpatient:

This module manages activities related to patient who visits the Hospital Resident Doctor or Consultant Doctor for Medical Consultations, diagnosis and treatment.

Inpatient:

Admission request will be made here. Request for admission is made before patient admitting the hospital.

Staff Details:

It keeps track of all details about doctors and staffs of the hospital. staffs, Doctors, Nurses name, staff id, address, qualification, cell no, e-mail are entered and stored in a separate form. Individual staff details can be viewed in the table using a separate form with a attribute Staff id.

Salary Details:

This module contains the details salary for the doctors and nurse. This salary calculated basic salary, PF, HRA, and year increment for the staffs from date of joining automatically calculated.

Consultation Details :

This module contains the details for the inpatient which includes disease name, type of treatment and medicine given to them.

Billing Details:

This module bills the both inpatient and outpatient who comes to hospital.

Ward Details:

This module enters and stores the details about each ward of the hospital for future reference. Individual ward detail can be viewed in the table using ward id .The attributes used in storing a ward detail is ward id, ward name, floor no, no of rooms.

Ambulance Services:

Another service for Ambulance availability, out time, in time, search for the ambulance went to place for petrol and fuel expenses.

SYSTEM SPECIFICATION

SYSTEM CONFIGURATION

HARDWARE REQUIREMENTS

PROCESSOR	:INTEL PENTIUM 4 (OR)HIGHER
RAM	:512 MB & ABOVE
HARD DISK DRIVE	: 500 MB FREE SPACE OR ABOVE
PRINTER	: INK-JET PRINTER
PEN DRIVE	: 512MB.

SOFTWARE REQUIREMENTS

PROGRAMMING LANGUAGE	: VISUAL PROGRAMMING
BACKEND	:MS SQL SERVER-2005
FRONT END	:VISUAL BASIC.NET
OPERATING SYSTEM	: WINDOWS XP & HIGHER VERSION

SOFTWARE SPECIFICATION

VISUAL STUDIO . NET

Visual Studio .NET is a complete set of development tools for building

- ASP Web applications
- XML Web services
- desktop applications
- mobile applications

Visual Basic .NET, Visual C++ .NET, and Visual C# .NET all use the same integrated development environment (IDE), which allows them to share tools and facilitates in the creation of mixed-language solutions.

Visual Basic

Visual Basic has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, and custom attributes. In addition, Visual Basic supports multithreading. Multithreading is the ability to assign individual tasks to separate processing threads.

C#

Visual C#, pronounced C sharp, is a new object-oriented programming language that is an evolution of C and C++, providing a simple and type-safe language for developing applications.

C++

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework. Attributes, like C++ keywords, are used in your source files and interpreted by the compiler. Attributes are designed to provide a quick and efficient method to simplify COM programming with Visual C++.

JScript

JScript has been updated to be a class-based, object-oriented scripting language that maintains full backwards compatibility with previous versions of JScript. JScript now provides class-based objects, typed variables, true compiled code, and cross-language support through Common Language Specification (CLS) compliance. The primary role of JScript is development of Web sites with ASP.NET and customization of applications with Script for the .NET Framework.

Types of forms used in .Net

Web Forms

Web Forms are an ASP.NET technology that you use to create programmable Web pages. Web Forms render themselves as browser-compatible HTML and script, which allows any browser on any platform to view the pages. Using Web Forms, you create Web pages by dragging and dropping controls onto the designer and then adding code, similar to the way that you create Visual Basic forms

Windows Forms

Windows Forms is the new platform for Microsoft Windows application development, based on the .NET Framework. This framework provides a clear, object-oriented, extensible set of classes that enables you to develop rich Windows applications. Additionally, Windows Forms can act as the local user interface in a multi-tier distributed solution.

The .NET Framework

Overview

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 is a multi-language environment for building, deploying, and running XML Web services and applications.

- **Common Language Runtime** Despite its name, the runtime actually has a role in both a component's runtime and development time experiences. While the component is running, the runtime is responsible for managing memory allocation, starting up and stopping threads and processes, and enforcing security policy, as well as satisfying any dependencies that the component might have on other components. At development time, the runtime's role changes slightly; because it automates so much (for example, memory management), the runtime makes the developer's experience very simple, especially when compared to COM as it is today. In particular, features such as reflection dramatically reduce the amount of code a developer must write in order to turn business logic into a reusable component.
- **Unified programming classes** The framework provides developers with a unified, object-oriented, hierarchical, and extensible set of class libraries (APIs). Currently, C++ developers use the Microsoft Foundation Classes and Java developers use the Windows Foundation Classes. The framework unifies these disparate models and gives Visual Basic and JScript programmer’s access to class libraries as well. By creating a common set of APIs across all programming languages, the common language runtime enables cross-language inheritance, error handling, and debugging. All programming languages, from JScript to C++, have similar access to the framework and developers are free to choose the language that they want to use.
- **ASP.NET** ASP.NET builds on the programming classes of the .NET Framework, providing a Web application model with a set of controls and infrastructure that make it simple to build ASP Web applications. ASP.NET includes a set of controls that encapsulate common HTML user

interface elements, such as text boxes and drop-down menus. These controls run on the Web server, however, and push their user interface as HTML to the browser. On the server, the controls expose an object-oriented programming model that brings the richness of object-oriented programming to the Web developer. ASP.NET also provides infrastructure services, such as session state management and process recycling that further reduce the amount of code a developer must write and increase application reliability. In addition, ASP.NET uses these same concepts to enable developers to deliver software as a service.

Debugging

Visual Studio .NET provides a single integrated debugger for all Visual Studio languages, including Visual Basic and C#. A new, unified interface combines features of the Visual C++ and Visual Basic 6.0 debuggers, as well as many new features.

- **Cross-Language Debugging**

Using the integrated debugger, you can debug projects that are part of the same solution but are written in different languages. For example, you can debug a solution that contains a project consisting of a Visual Basic or Visual C# user interface application and a Visual C++ server application, and you can step back and forth between these projects, for example from Managed Extensions for C++ to Visual Basic and back to Managed Extensions for C++.

- **Attaching to a Running Program**

You can attach the debugger to a program that is already running and debug the program. Attaching to a running program works the same way whether the program is running on a host machine or a remote machine. The program does not need to be launched in the Visual Studio .NET integrated development environment (IDE).

- **Remote Debugging**

You can attach to and debug a process that is running on a different computer from the one on which you are running Visual Studio. For example, if you are debugging a Windows Forms client application and an XML Web service it uses, you can run Visual Studio on the client and then attach to the server that is providing the XML Web service. You typically perform this kind of debugging for Web Forms projects or ASP.NET Web Service projects.

- **Debugging Multithreaded Applications**

New for Visual Basic users is the ability to write and debug multithreaded applications. The Threads window can be used to view the threads that are running and to switch context.

- **Debugging Multiple Programs**

You can debug multiple programs by attaching to running programs or by launching multiple programs from the Visual Studio IDE.

- **Debugging ASP.NET Web Applications**

Configuring debugging for ASP.NET Web applications has been significantly improved. In most cases you can simply choose Start from the Debug menu and the debugger will automatically attach to the ASP.NET worker process for debugging Web Forms, even if the ASP.NET worker process is running on a different server. Security configuration has been made easier by the addition of a debugger users group.

- **.NET Framework Classes for Debugging and Code Tracing**
  You can include instrumentation code in your Microsoft .NET application. Several .NET Framework classes are available for use both in debugging and in instrumenting your code.

Visual Basic.Net

New in Visual Basic

Web Development: Visual Basic and C# include support for Web Forms and XML Web services. Web Forms allow you to easily and quickly create the browser-based interface for ASP.NET Web applications. XML Web services allow you to package a Visual Basic or C# method and make it accessible on the Web.

New in Data: ADO.NET provides scalable, high-performance data access for all Visual Studio
Applications based on the .NET Framework. You can access data directly in the data source or create an in-memory cache — a dataset — for working with disconnected data. You can also work directly with XML as relational data or by creating and editing XML files directly.

Windows Forms and Controls: You can use Windows Forms and controls to create the presentation layer of a distributed application. Windows Forms provides a clear, object-oriented, extensible set of classes that enable you to develop rich Windows applications.

New in Projects: You can use the Project templates in this version to easily create various types of Windows and ASP.NET Web applications and controls. Project templates set necessary references for you, import namespaces, and add default items to your projects.

Components and Component Authoring: You can use nonvisual components and associated features to incorporate resources such as message queues, event logs, and performance counters into your applications. This version also offers RAD support for component creation via the Component Designer and framework classes that help you create controls and components.

New in Debugging: You can now debug projects that are part of the same solution but written in different languages, perform remote debugging, and implement trace functionality throughout your code to retrieve finely tuned output.

Extensibility and Automation: Visual Studio .NET includes a programmable object model that provides access to the underlying components and events of the integrated development environment (IDE). This model allows you to extend the functionality of the IDE, automate repetitive tasks, and integrate the IDE with other applications.

ADO (Activex Data Objects).Net

Most Visual Basic and Visual C# applications revolve around reading and updating information in databases. To allow data integration in distributed, scalable applications, Visual Studio .NET provides support for a new generation of data access technology: ADO.NET.

Data Access with ADO.NET

As you develop applications using ADO.NET, you will have different requirements for working with data. In some cases, you might simply want to display data on a form. In other cases, you might need to devise a way to share information with another company.

No matter what you do with data, there are certain fundamental concepts that you should understand about the data approach in ADO.NET. You might never need to know some of the details of data handling — for example, you might never need to directly edit an XML file containing data — but it is very useful to understand the data architecture in ADO.NET, what the major data components are, and how the pieces fit together.

New in ADO.Net

ADO.NET Does Not Depend On Continuously Live Connections

In traditional client/server applications, components establish a connection to a database and keep it open while the application is running. For a variety of reasons, this approach is impractical in many applications:

- Open database connections take up valuable system resources. In most cases, databases can maintain only a small number of concurrent connections. The overhead of maintaining these connections detracts from overall application performance.
- Similarly, applications that require an open database connection are extremely difficult to scale up. An application that does not scale up well might perform acceptably with four users but will likely not do so with hundreds. ASP.NET Web applications in particular need to be easily scalable, because traffic to a Web site can go up by orders of magnitude in a very short period.
- In ASP.NET Web applications, the components are inherently disconnected from each other. The browser requests a page from the server; when the server has finished processing and sending the page, it has no further connection with the browser until the next request. Under

these circumstances, maintaining open connections to a database is not viable, because there is no way to know whether the data consumer (the client) requires further data access.

- A model based on always-connected data can make it difficult and impractical to exchange data across application and organizational boundaries using a connected architecture. If two components need to share the same data, either have to be connected, or a way must be devised for the components to pass data back and forth.

For all these reasons, data access with ADO.NET is designed around an architecture that uses connections sparingly. Applications are connected to the database only long enough to fetch or update the data. Because the database is not holding on to connections that are largely idle, it can service many more users.

Benefits of ADO.NET

Interoperability

ADO.NET applications can take advantage of the flexibility and broad acceptance of XML. Because XML is the format for transmitting datasets across the network, any component that can read the XML format can process data. In fact, the receiving component need not be an ADO.NET component at all: The transmitting component can simply transmit the dataset to its destination without regard to how the receiving component is implemented. The destination component might be a Visual Studio application or any other application implemented with any tool whatsoever. The only requirement is that the receiving component be able to read XML. As an industry standard, XML was designed with exactly this kind of interoperability in mind.

Maintainability

In the life of a deployed system, modest changes are possible, but substantial, architectural changes are rarely attempted because they are so difficult. That is unfortunate, because in a natural course of events, such substantial changes can become necessary. For example, as a deployed application becomes popular with users, the increased performance load might require architectural changes. As the performance load on a deployed application server grows, system resources can become scarce and response time or throughput can suffer. Faced with this problem, software architects can choose to divide the server's business-logic processing and user-interface processing onto separate tiers on separate machines. In effect, the application server tier is replaced with two tiers, alleviating the shortage of system resources.

Programmability

ADO.NET data components in Visual Studio encapsulate data access functionality in various ways that help you program more quickly and with fewer mistakes. For example, data commands abstract the task of building and executing SQL statements or stored procedures.

Performance

For disconnected applications, ADO.NET datasets offer performance advantages over ADO disconnected recordsets. When using COM marshalling to transmit a disconnected recordset among tiers, a significant processing cost can result from converting the values in the recordset to data types recognized by COM. In ADO.NET, such data-type conversion is not necessary.

Scalability

Because the Web can vastly increase the demands on your data, scalability has become critical. Internet applications have a limitless supply of potential users. Although an application might serve a dozen users well, it might not serve hundreds —or hundreds of thousands — equally well. An application that consumes resources such as database locks and database connections will not serve high numbers of users well, because the user demand for those limited resources will eventually exceed their supply.

Comparison of ADO.NET and ADO

You can understand the features of ADO.NET by comparing them to particular features of ActiveX Data Objects (ADO).

In-memory Representations of Data

In ADO, the in-memory representation of data is the recordset. In ADO.NET, it is the dataset. There are important differences between them.

Number of Tables

A recordset looks like a single table. If a recordset is to contain data from multiple database tables, it must use a JOIN query, which assembles the data from the various database tables into a single result table.

In contrast, a dataset is a collection of one or more tables. The tables within a dataset are called data tables; specifically, they are DataTable objects. If a dataset contains data from multiple database tables, it will typically contain multiple **DataTable** objects. That is, each **DataTable** object typically corresponds to a single database table or view. In this way, a dataset can mimic the structure of the underlying database.

A dataset usually also contains relationships. A relationship within a dataset is analogous to a foreign-key relationship in a database —that is, it associates rows of the tables with each other. For example, if a dataset contains a table about investors and another table about each investor's stock purchases, it could also contain a relationship connecting each row of the investor table with the corresponding rows of the purchase table.

Because the dataset can hold multiple, separate tables and maintain information about relationships between them, it can hold much richer data structures than a recordset, including self-relating tables and tables with many-to-many relationships.

Data Navigation and Cursors

In ADO you scan sequentially through the rows of the recordset using the ADO **MoveNext** method. In ADO.NET, rows are represented as collections, so you can loop through a table as you would through any collection, or access particular rows via ordinal or primary key index. **DataRelation** objects maintain information about master and detail records and provide a method that allows you to get records related to the one you are working with. For example, starting from the row of the Investor table for "Nate Sun," you can navigate to the set of rows of the Purchase table describing his purchases.

A *cursor* is a database element that controls record navigation, the ability to update data, and the visibility of changes made to the database by other users. ADO.NET does not have an inherent cursor object, but instead includes data classes that provide the functionality of a traditional cursor. For example, the functionality of a forward-only, read-only cursor is available in the ADO.NET **DataReader** object. For more information about cursor functionality, see Data Access Technologies.

Minimized Open Connections

In ADO.NET you open connections only long enough to perform a database operation, such as a Select or Update. You can read rows into a dataset and then work with them without staying connected to the data source. In ADO the recordset can provide disconnected access, but ADO is designed primarily for connected access.

There is one significant difference between disconnected processing in ADO and ADO.NET. In ADO you communicate with the database by making calls to an OLE DB provider. In ADO.NET you communicate with the database through a data adapter (an OleDbDataAdapter or SqlDataAdapter object), which makes calls to an OLE DB provider or the APIs provided by the underlying data source. The important difference is that in ADO.NET the data adapter allows you to control how the changes to the dataset are transmitted to the database — by optimizing for performance, performing data validation checks, or adding any other extra processing.

Sharing Data Between Applications

Transmitting an ADO.NET dataset between applications is much easier than transmitting an ADO disconnected recordset. To transmit an ADO disconnected recordset from one component to another, you use COM marshalling. To transmit data in ADO.NET, you use a dataset, which can transmit an XML stream.

The transmission of XML files offers the following advantages over COM marshalling:

Richer data types

COM marshalling provides a limited set of data types — those defined by the COM standard. Because the transmission of datasets in ADO.NET is based on an XML format, there is no restriction on data types. Thus, the components sharing the dataset can use whatever rich set of data types they would ordinarily use.

Performance

Transmitting a large ADO recordset or a large ADO.NET dataset can consume network resources; as the amount of data grows, the stress placed on the network also rises. Both ADO and ADO.NET let you minimize which data is transmitted. But ADO.NET offers another performance advantage, in that ADO.NET does not require data-type conversions. ADO, which requires COM marshalling to transmit records sets among components, does require that ADO data types be converted to COM data types.

Penetrating Firewalls

A firewall can interfere with two components trying to transmit disconnected ADO recordsets. Remember, firewalls are typically configured to allow HTML text to pass, but to prevent system-level requests (such as COM marshalling) from passing.

Because components exchange ADO.NET datasets using XML, firewalls can allow datasets to pass.

SQL SERVER

SQL SERVER:

SQL server is a client/server relational database management system (RDBMS) that uses transact-SQL to send request between a client and SQL server.

Client/server Architecture

SQL server is designed to be a client/server system. Client/server systems are constructed so that the database can reside on a central computer, know as a server, and be shared among several users. When users want to access the date in SQL server, they run an application on their local computer, know as a client that connects over a network to the server running SQL server.

SQL server can work with thousands of client applications simultaneously. The server has features to prevent the logical problems that occur if a user tries to read or modify data currently being used by others.

While SQL server is designed to work as a server in a client/server network. It also capable of working as a stand-alone database directly on the client. The scalability and ease-of-use features of SQL server allow it to work efficiently on a client without consuming too many resources. SQL server efficiently allocates the available resources, such as memory, network bandwidth, and disk I/O, among the multiple users.

- **Additional facility Like TIMESTAMP and TIMESTAMP with TIMEZONE for storing Time.**

- **Flexibility in intervals setting**

- Code’s 12 rules are satisfied. That is,

    - Data Representation
    - Rule of guaranteed access
    - Proper treatment of null value
    - Security
    - Versioning
    - Physical Independence
    - Logical Independence
    - Integrity constraint independence
    - View Updating
    - Data Description
    - Comprehensive data sub language
    - Insert and update rule
    - Distribution

SYSTEM ANALYSIS

System Analysis

System analysis is the first and foremost step performed in developing the software to solve a particular problem. In the analysis part, a software developer examines the requirements. Carrying out preliminary investigation identifies these requirements

Analysis consists of two sub phases

- Planning
- Requirement definition

During planning phase, cost estimates and work schedules will be planned. Requirement definition is a specification that describes the processing environment, the required software functions, performance constraints (size, speed, machine configuration) and exception handling.

EXISTING SYSTEM

The existing system uses manual transaction processing.

Drawbacks

- Large amount of clerical time is required.
- The record maintainability is difficult.
- Accessibility of accurate information from the past record is difficult.
- There is always delay in information search and retrieval. It requires many people to carry out a single problem.
- Lot of human resources is required.
- Data reliability and maintainability is difficult.
- Lot amount of records need much place to save.
- The paper works have to be taken care.
- The patient entry form may miss
- The doctor appointment cannot be maintained in properly
- The patient fix the appointment to consulting with doctor such things may misplaced
- The patients records verification is too complicated
- Compare with the patient and doctor record verification and validation is too complicated
- Cannot be maintaining manual record for long time

PROPOSED SYSTEM

The proposed system has been designed to overcome all the drawbacks found in the existing system. The new system has been proposed to use VISUAL BASIC.NET as front end and SQL SERVER as backend.

The proposed system has enhanced features, which was not found in the existing system. The salient features are

- Security for the data is done easily.
- Validation is done to enter correct data.
- Memory consumption is very less and the processing speed is fast.
- Data reports are presented in a neat format.
- It is apt for this modern world.

It is easy to combine the database of other software and to view the records in the files and also it is easy to get the reports by giving input data.

SYSTEM DESIGN

SYSTEM DESIGN

THE SYSTEM DESIGN PROCESS:

System design develops the architectural detail required to build a system or product.

The system design process encompasses the following activities:

• Partition the analysis model into subsystems.

• Identify concurrency that is dictated by the problem.

• Allocate subsystems to processors and tasks.

• Develop a design for the user interface.

• Choose a basic strategy for implementing data management.

• Identify global resources and the control mechanisms required to access them.

• Design an appropriate control mechanism for the system, including task management.

• Consider how boundary conditions should be handled.

• Review and consider trade-offs.

Input design

Input design is a part of overall system design, requires the very careful analysis of the input data items. The goal of the input design is to make the data entry easier, logical and free from errors. The user controls input data.

The commonly used input, output devices are mouse, keyboard and the visual display unit. The well designed, well organized screen formats are used to acquire the inputs. The data accepted is stored on database file.

Our system is classified into subsystem such as

- Admission
- Staff Details
- Billing
- Consultation Details
- Ward Details
- Ambulance Service
- Data Report

Output Design

Output is the most important and direct source of information the user.

Efficient & intelligent output design improves the system relationships with the users and helps in decision-making. The output is collected in order to help the user to make a wise decision.

DATABASE DESIGN

INPATIENT

NAME	**TYPE**	**CONSTRAINTS**
Mrdno	Bigint	Primary key
Hno	Bigint	Not null
Name	Varchar(100)	Not null
Age	Int	Not null
Addr	Varchar(120)	Not null
Dob	Datetime	Not null
Gender	Varchar(20)	Not null
State	Varchar(54)	Not null
District	Varchar(100)	Not null
Concession	Varchar(100)	Not null
Referal	Varchar(100)	Not null
Date of admit	Datetime	Not null

OUT PATIENT

NAME	**TYPE**	**CONSTRAINTS**
Hno	Bigint	Primary key
Name	Varchar(100)	Not null
Age	Int	Not null
Addr	Varchar(120)	Not null
Dob	Datetime	Not null
Gender	Varchar(20)	Not null
State	Varchar(54)	Not null
District	Varchar(100)	Not null
Concession	Varchar(100)	Not null
Referal	Varchar(100)	Not null
Date	Datetime	Not null

STAFF

NAME	TYPE	CONSTRAINTS
Idno	Int	Primary key
Name	Varchar(50)	Not null
Department	Varchar(100)	Not null
Doj	Datetime	Not null
Gender	Varchar(30)	Not null
Address	Varchar(150)	Not null
Salary	Int	Not null
Dtc	Datetime	Not null
Reason	Varchar(200)	Not null
Age	Int	Not null
Email	Varchar(100)	Not null
Phone	Varchar(90)	Not null
Cell	Varchar(90)	Not null

PAYROLL

NAME	TYPE	CONSTRAINTS
Idno	Bigint	Primary key
Name	Varchar(100)	Not null
Department	Varchar(50)	Not null
Doj	Datetime	Not null
Gender	Varchar(20)	Not null
Dt	Datetime	Not null
Sperday	Bigint	Not null
Odpay	Bigint	Not null
Salary	Bigint	Not null

ALLOWANCE

NAME	**TYPE**	**CONSTRAINTS**
Idno	Bigint	Primary key
Name	Varchar(100)	Not null
Date	Datetime	Not null
Pf	Bigint	Not null
Da	Bigint	Not nul
Hra	Bigint	Not nul
Ta	Bigint	Not null
Ga	Bigint	Not null

INPATIENT BILL

NAME	**TYPE**	**CONSTRAINTS**
Bno	Bigint	Primary key
Hno	Int	Not null
Mrdno	Bigint	Not null
Pname	Varchar(100)	Not null
Department	Varchar(50)	Not null
Date	Datetime	Not null
Rrent	Int	Not null
Lab	Int	Not null
Ad	Datetime	Not null
Dd	Datetime	Not null
Amount	Bigint	Not null

OUTPATIENT BILL

NAME	**TYPE**	**CONSTRAINTS**
Bno	Int	Primary key
Hno	Int	Not null
Patientname	Varchar(50)	Not null
Docname	Varchar(50)	Not null
Date	Datetime	Not nul
Amount	Int	Not null

WARD

NAME	**TYPE**	**CONSTRAINTS**
Mrdno	Bigint	Primary key
Hno	Bigint	Not null
Name	Varchar(100)	Not null
Department	Varchar(100)	Not null
Wardno	Int	Not null
Rtype	Varchar(100)	Not null
Bno	Int	Not null
Adt	Datetime	Not null
Ddt	Datetime	Not null

AMBULANCE

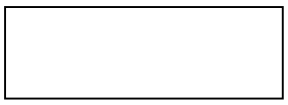
NAME	**TYPE**	**CONSTRAINTS**
Amno	Int	Primary key
Type	Varchar(50)	Not null
Designation	Varchar(100)	Not null
Stime	Datetime	Not null
Rtime	Datetime	Not null
Pdexp	Int	Not null
Stentry	Bigint	Not null
Rtentry	Bigint	Not null
Drivername	Nchar(100)	Not null
Driverno	Bigint	Not null
Num	Nchar(50)	Not nul

DATA FLOW DIAGRAM

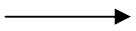
DATA FLOW DIAGRAM

Data Flow Diagram (DFD) is a design tool constructed to show how data within the system. It is designed from the data which is collected during data collection phase. DFD is otherwise called as “Bubble Chart”.

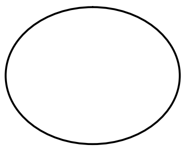
There are five symbol used in DFD. They are Rectangle, Open Rectangle, Circle, arrow, small circle. Each one has its own meaning.



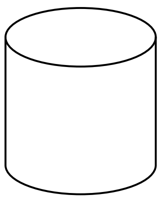
- Source or Destination



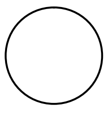
- Data flow



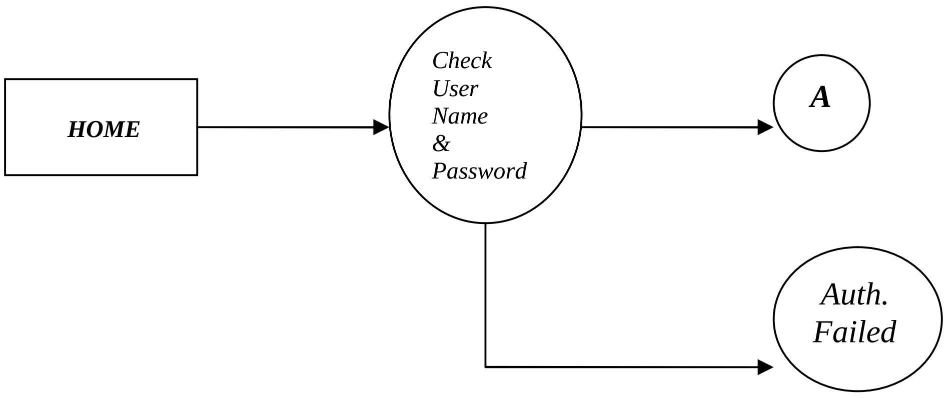
- Process

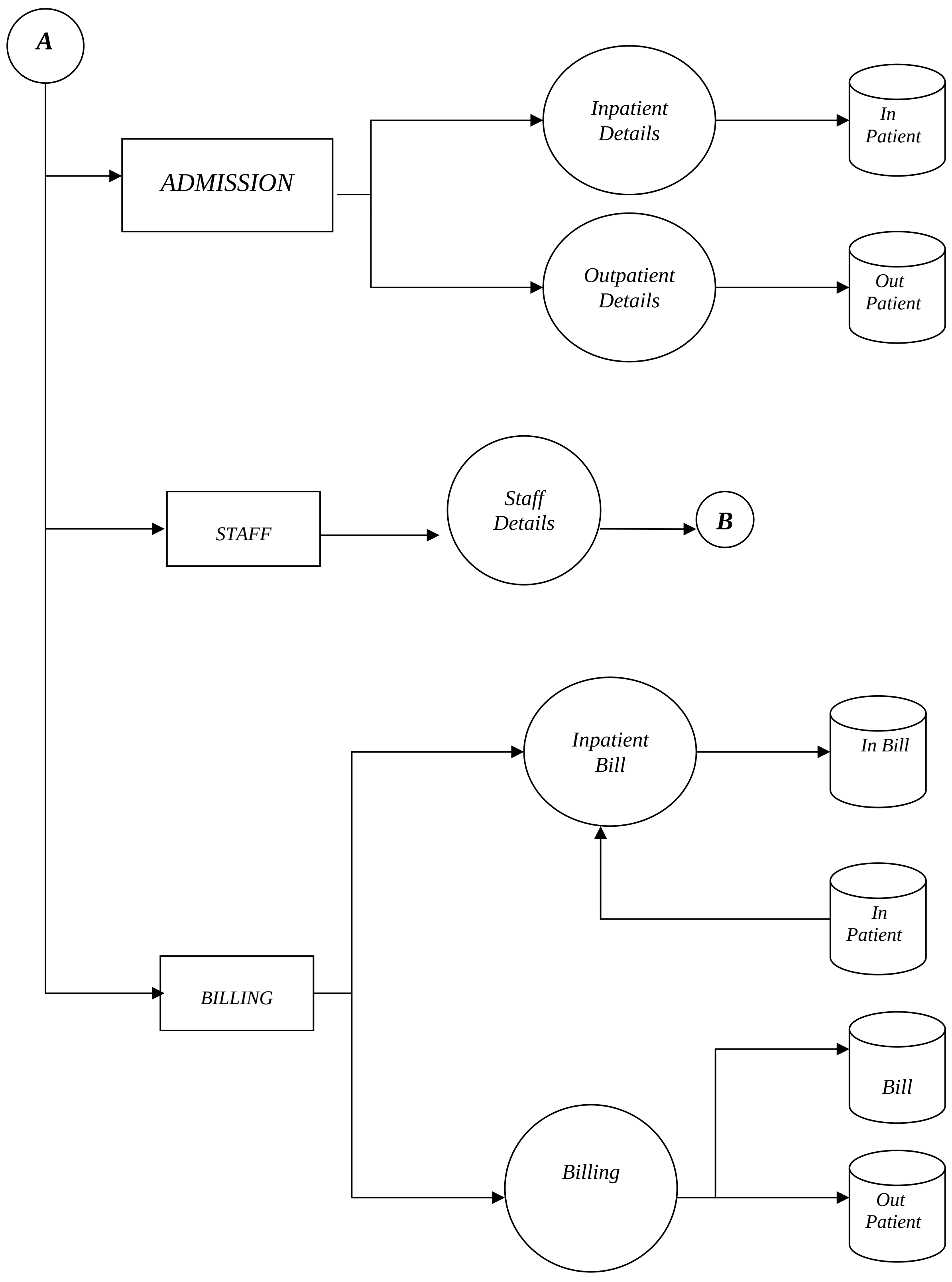


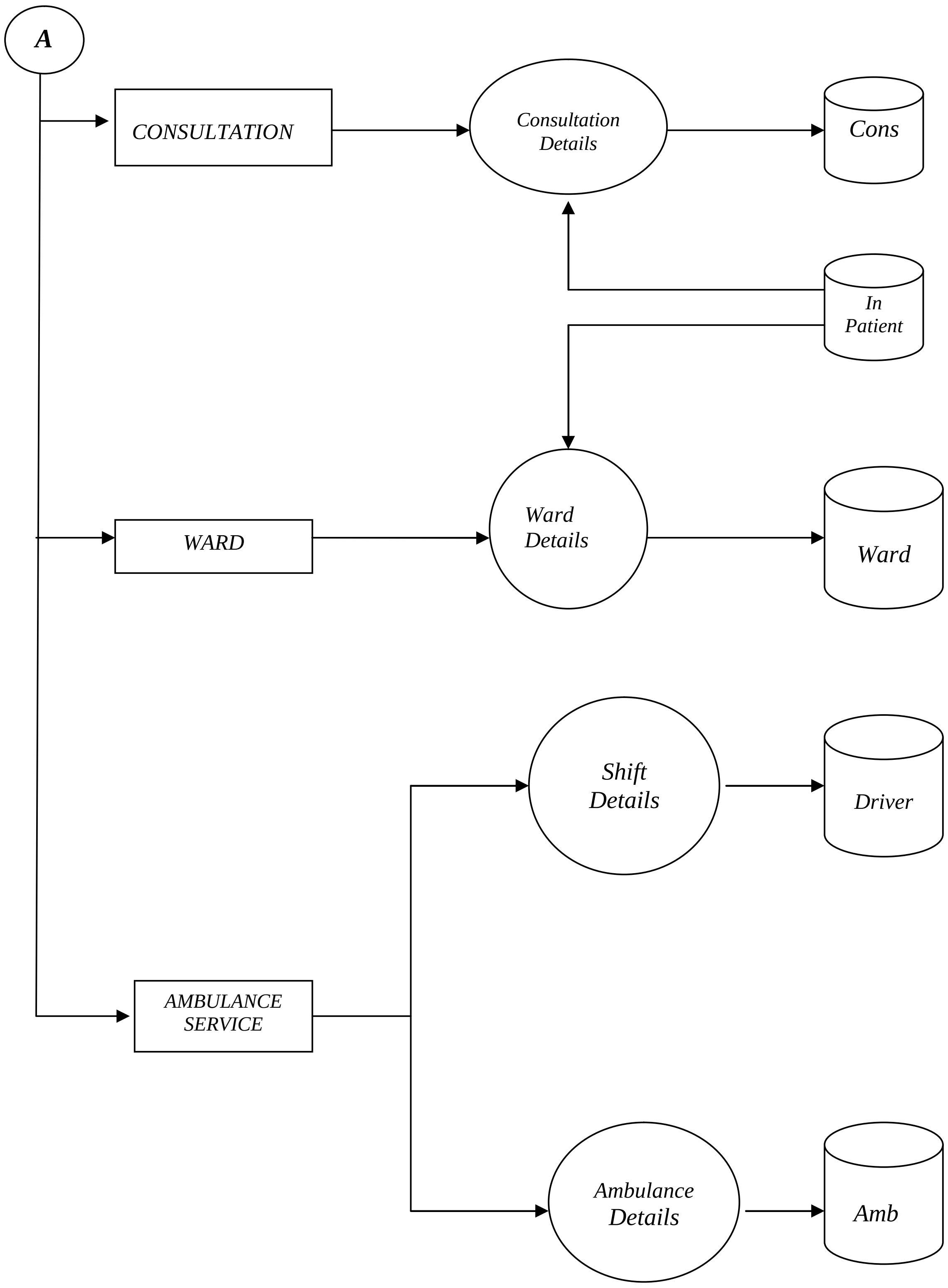
- Data Storage

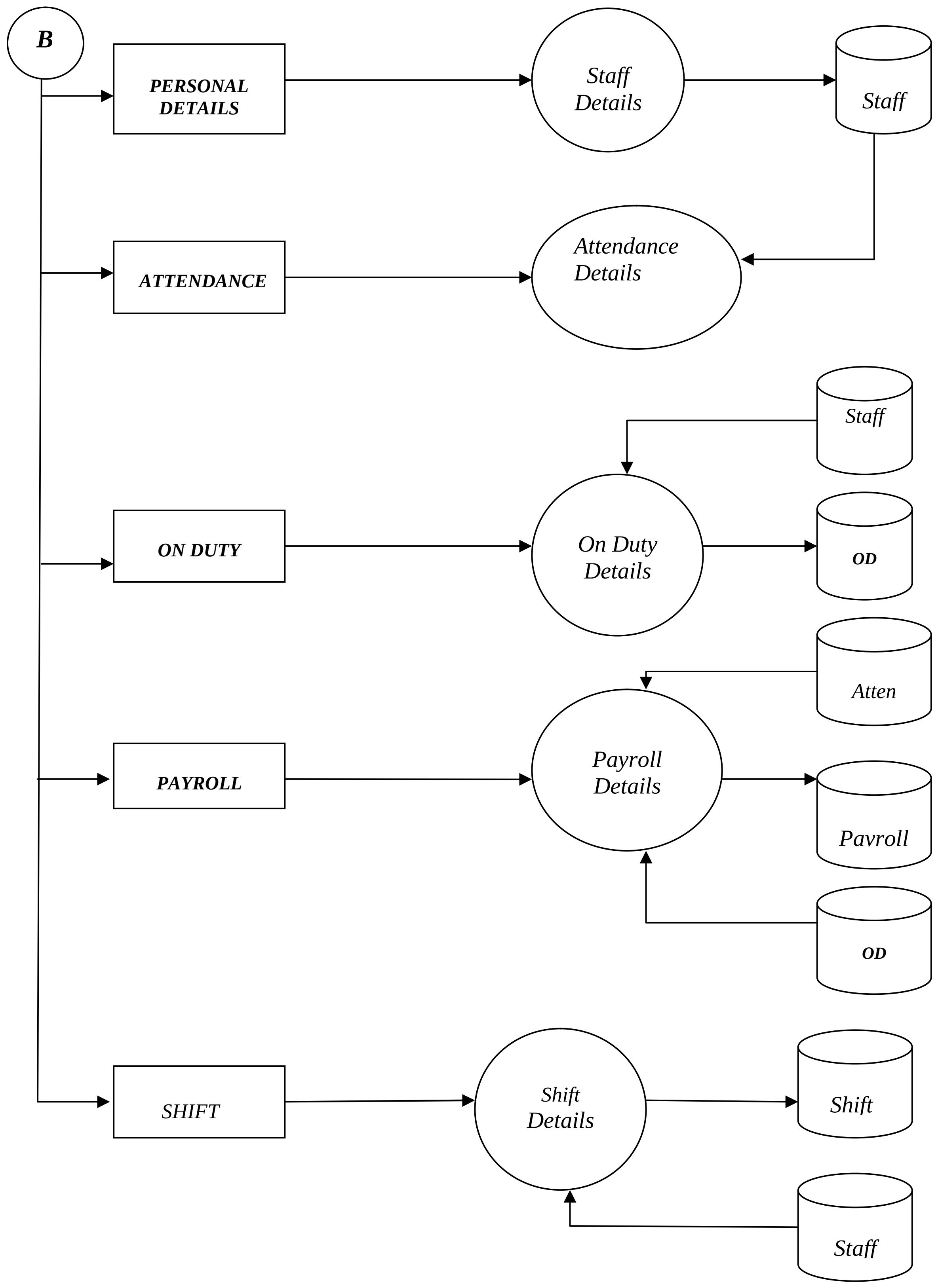


- Control Flow


HOME
Check User Name & Password
A
Auth. Failed


A
ADMISSION
Inpatient Details
In Patient
Outpatient Details
Out Patient
STAFF
Staff Details
B
Inpatient Bill
In Bill
In Patient
BILLING
Bill
Billing
Out Patient


A
CONSULTATION
Consultation Details
Cons
In Patient
WARD
Ward Details
Ward
Shift Details
Driver
AMBULANCE SERVICE
Ambulance Details
Amb


B
PERSONAL DETAILS
Staff Details
Staff
ATTENDANCE
Attendance Details
Staff
ON DUTY
On Duty Details
OD
Atten
PAYROLL
Payroll Details
Pavroll
OD
SHIFT
Shift Details
Shift
Staff

SYSTEM TESTING

System Testing

System Testing is an important stage in any system development life cycle. Testing is a process of executing a program with the intention of finding errors. The importance of software testing and its implications with respect to software quality cannot be overemphasized. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. A good test case is one that has a high probability of finding a yet undiscovered error.

Testing is the set of activities that can be planned in advance and conducted systematically. Different test conditions should be thoroughly checked and the bugs detected should be fixed. The testing strategies formed by the user are performed to prove that the software is free and clear from errors. To do this, there are many ways of testing the system’s reliability, completeness and maintainability.

Unit Testing:

In the unit testing the analyst tests the program making up a system. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function. In a large system, many modules on different levels are needed.

Unit testing can be performed from the bottom up starting with the smallest and lowest level modules and proceeding one at a time. For each module in a bottom-up testing, a short program executes the module and provides the needed data.

Integration Testing:

Integration testing is a systematic technique for constructing the program structure while conducting test to uncover errors associate with interfacing. Objectives are used to take unit test modules and built program structure that has been directed by design.

The integration testing is performed for this Hospital Management System when all the modules where to make it a complete system. After integration the project works successfully.

Validation Testing:

Validation testing can be defined in many ways, but a simple definition is that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists.

- The functions or performance characteristics confirm to specification   and are accepted.
- A deviation from specification is uncovered and a deficiency list is created.

Proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

For example, in this project validation testing is performed against inpatient search module. This module is tested with the following valid and invalid inputs for the field patientname.

White Box Testing

White box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white box testing methods, the software engineer can derive test cases that

- Guarantee that all independent paths with in a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and with in their operational bounds and
- Exercise internal data structure to assure their validity.

For example in this project white box testing is performed against inpatient module. Without entering text if we apply it displays the message "First add record then save it" else it should be saved.

Black Box Testing

This method treats the coded module as a black box. The module runs with inputs that are likely to cause errors. Then the output is checked to see if any error occurred. This method cannot be used to test all errors, because some errors may depend on the code or algorithm used to implement the module.

System Implementation

Implementation is the process of having system personal check out and provides new equipments into use, train the users to install a new application and construct any files of data needed to use it. There are three types of implementation.

- Implementation of computer system to replace a manual system. The problems encountered are covering files, training users, creating accurate files and verifying print outs for integrity.
- Implementation of a new computer system to replace an existing one. This is usually difficult conversion. If not properly planned, there can be many problems. So large computer systems may take as long as a year to convert.
- Implementation of a modified application to replace the existing one using the same computer. This type of conversion is relatively easy to handle, usually there are no major changes in the file.

Our project is yet to be implemented.

SAMPLE CODINGS

SAMPLE CODINGS :

Login form:

```
Public Class Login
    Dim form1 As New mainmenu
    Private Sub OK_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles OK.Click
        If PasswordTextBox.Text = "hospital" Then
            MsgBox("Login Success", MsgBoxStyle.Information,
"Hospital Management")
            PasswordTextBox.Text = ""
            Me.Hide()
            mainmenu.Show()
        Else
            MsgBox("Login Failed", MsgBoxStyle.Critical,
"Hospital Management")
            PasswordTextBox.Text = ""
            Me.Show()
        End If
    End Sub

    Private Sub Cancel_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Cancel.Click
        Me.Close()
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        End
    End Sub
End Class
```

Main Menu:

```
Public Class mainmenu

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        End
    End Sub

    Private Sub InpatientToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
InpatientToolStripMenuItem.Click
        Me.Hide()
        inpatient.Show()
    End Sub
    Private Sub OutpatientToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
OutpatientToolStripMenuItem.Click
        Me.Hide()
        outpatient.Show()
    End Sub

    Private Sub DoctorToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Hide()
        doctper.Show()
    End Sub

    Private Sub DoctorToolStripMenuItem1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Hide()
        docpay.Show()
    End Sub

    Private Sub DoctorToolStripMenuItem2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Hide()
        doctsal.Show()

    End Sub

    Private Sub PersonalDetailsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
PersonalDetailsToolStripMenuItem.Click
        Me.Hide()
        doctper.Show()
    End Sub

    Private Sub mainmenu_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub


    Private Sub DoctorToolStripMenuItem3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Hide()
        doctate.Show()
    End Sub


    Private Sub DoctorToolStripMenuItem4_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        doctsif.Show()

    End Sub
    Private Sub ODFormToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ODFormToolStripMenuItem.Click
        Me.Hide()
        doctsal.Show()
    End Sub

    Private Sub PayrollDetailsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
PayrollDetailsToolStripMenuItem.Click
        Me.Hide()
        docpay.Show()
    End Sub

    Private Sub AttendanceDetailsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
AttendanceDetailsToolStripMenuItem.Click
        Me.Hide()
        doctate.Show()
    End Sub

    Private Sub ShiftDetailsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
ShiftDetailsToolStripMenuItem.Click
        Me.Hide()
        doctsif.Show()
    End Sub

    Private Sub InpatientToolStripMenuItem1_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
InpatientToolStripMenuItem1.Click
        Me.Hide()
        inpatbill.Show()
    End Sub


    Private Sub OutpatientToolStripMenuItem1_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
OutpatientToolStripMenuItem1.Click
        Me.Hide()
        outpatbill.Show()
    End Sub

    Private Sub ConsultationToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
ConsultationToolStripMenuItem.Click
        Me.Hide()
        consul.Show()
    End Sub
    Private Sub WardDetailsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
WardDetailsToolStripMenuItem.Click
        Me.Hide()
        ward.Show()

    End Sub

    Private Sub
BillingAndPharmacyToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
BillingAndPharmacyToolStripMenuItem.Click

    End Sub

    Private Sub AmbulanceServiceToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
AmbulanceServiceToolStripMenuItem.Click
        Me.Hide()
        ambul.Show()

    End Sub

    Private Sub ExitToolStripMenuItem1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Hide()
        Login.Show()
    End Sub

    Private Sub ExitToolStripMenuItem_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ExitToolStripMenuItem.Click
        Me.Hide()
        Login.Show()
    End Sub

    Private Sub InpatientsToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
InpatientsToolStripMenuItem.Click
        iptreport.Show()
    End Sub

    Private Sub PharmacyToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
PharmacyToolStripMenuItem.Click
        iptbillview.Show()
    End Sub

    Private Sub FeesCollectionsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
FeesCollectionsToolStripMenuItem.Click
        outbillview.Show()
    End Sub
End Class
```

Inpatient Admission :

```
Imports System.Data.OleDb
Public Class inpatient
    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        Dim cmd As New OleDbCommand
        conn.open()
        Dim sql As String = "insert into ipt values('" &
TextBox6.Text & "','" & TextBox5.Text & "','" & TextBox4.Text
& "','" & RichTextBox4.Text & "','" & DateTimePicker1.Text &
"','" & ComboBox1.Text & "','" & TextBox2.Text & "','" &
TextBox3.Text & "','" & TextBox7.Text & "','" & TextBox8.Text
& "','" & DateTimePicker2.Text & "')"
        cmd = New OleDbCommand(sql, conn)
        cmd.ExecuteReader()
        MsgBox("Record Inserted", MsgBoxStyle.Information,
"Hospital Management")
        TextBox2.Text = ""
        TextBox3.Text = ""
        TextBox4.Text = ""
        RichTextBox4.Text = ""
        TextBox5.Text = ""
        TextBox6.Text = ""
        TextBox7.Text = ""
        TextBox8.Text = ""
        ComboBox1.Text = ""
        conn.close()
    End Sub

    Private Sub inpatient_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Me.ComboBox1.Items.Add("Male")
        Me.ComboBox1.Items.Add("Female")
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Me.Hide()
        mainmenu.Show()
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
        iptrecord.Show()
    End Sub
End Class
```

Outpatient Admission :

```
Imports System.Data.OleDb
Public Class outpatient
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Me.Hide()
        mainmenu.Show()

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        Dim cmd As New OleDbCommand
        conn.open()
        Dim sql As String = "insert into opt values('" &
TextBox6.Text & "','" & TextBox5.Text & "','" & TextBox4.Text
& "','" & RichTextBox4.Text & "','" & DateTimePicker1.Text &
"','" & ComboBox1.Text & "','" & TextBox2.Text & "','" &
TextBox3.Text & "','" & TextBox7.Text & "','" & TextBox8.Text
& "')"
        cmd = New OleDbCommand(sql, conn)
        cmd.ExecuteReader()
        MsgBox("Record Inserted", MsgBoxStyle.Information,
"Hospital Management")
        TextBox2.Text = ""
        TextBox3.Text = ""
        TextBox4.Text = ""
        RichTextBox4.Text = ""
        TextBox5.Text = ""
        TextBox6.Text = ""
        TextBox7.Text = ""
        TextBox8.Text = ""
        ComboBox1.Text = ""
        conn.close()
    End Sub

    Private Sub outpatient_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Me.ComboBox1.Items.Add("Male")
        Me.ComboBox1.Items.Add("Female")
    End Sub
End Class
```

Inpatient Bill :

```
Imports System.Data
Imports System.Data.OleDb
Public Class inpatbill
    Dim adp As New OleDbDataAdapter
    Dim ds As New DataSet
    Dim x As New Integer
    Dim cmd As New OleDbCommand
    Dim dr As OleDbDataReader
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Me.Hide()
        mainmenu.Show()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        conn.open()
        cmd = New OleDbCommand("insert into inbill values('" &
TextBox1.Text & "','" & ComboBox1.Text & "','" & TextBox3.Text
& "','" & DateTimePicker2.Text & "','" & TextBox8.Text & "','"
& TextBox9.Text & "','" & DateTimePicker1.Text & "','" &
DateTimePicker2.Text & "','" & TextBox11.Text & "')", conn)
        cmd.ExecuteNonQuery()
        MsgBox("Bill Saved to Database", MsgBoxStyle.OkOnly,
"Hospital Management")
        cmd.Dispose()
        TextBox1.Text = " "
        TextBox3.Text = " "
        TextBox8.Text = " "
        TextBox9.Text = " "
        TextBox10.Text = " "
        TextBox11.Text = " "
        ComboBox1.Text = " "
        DateTimePicker1.Value = Now.Date
        DateTimePicker2.Value = Now.Date
        conn.close()
    End Sub

    Private Sub inpatbill_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        ComboBox1.TabIndex = 0
        TextBox9.TabIndex = 1
        TextBox10.TabIndex = 2
        TextBox11.TabIndex = 3
        conn.open()
        cmd = New OleDbCommand("select * from ipt", conn)
        dr = cmd.ExecuteReader
        While (dr.Read)
            ComboBox1.Items.Add(dr(0).ToString)
        End While
        cmd.Dispose()
        cmd = New OleDbCommand("select max(bno) from inbill",
conn)
        dr = cmd.ExecuteReader
        If dr.Read = True Then
            TextBox1.Text = dr(0) + 1.ToString
        Else
            TextBox1.Text = 1
        End If
        cmd.Dispose()
        conn.close()
    End Sub


    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
        conn.open()
        adp = New OleDbDataAdapter("select * from ipt where
mrdno=' " & ComboBox1.Text & "'", adpx)
        adp.Fill(ds, "ipt")
        TextBox3.Text = ds.Tables(0).Rows(0)(1).ToString
        DateTimePicker1.Value =
ds.Tables(0).Rows(0)(10).ToString()
        DateTimePicker2.Value = Now.Date
        x = DateTimePicker2.Value.Day -
DateTimePicker1.Value.Day
        TextBox8.Text = x * 300
        conn.close()
    End Sub

    Private Sub TextBox11_Enter(ByVal sender As Object, ByVal
e As System.EventArgs) Handles TextBox11.Enter
        TextBox11.Text = " "
        Dim x, y, z As New Integer
        x = TextBox8.Text
        y = TextBox9.Text
        z = TextBox10.Text
        TextBox11.Text = x + y + z
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
        iptbillview.Show()
    End Sub
End Class
```

Outpatient Bill :

```
Imports System.Data.OleDb
Imports System.Data
Public Class outpatbill
    Dim adp As New OleDbDataAdapter
    Dim ds As New DataSet
    Dim x As New Integer
    Dim cmd As New OleDbCommand
    Dim dr As OleDbDataReader
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Me.Hide()
        mainmenu.Show()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        conn.open()
        cmd = New OleDbCommand("insert into bill values('" &
TextBox1.Text & "','" & ComboBox1.Text & "','" & TextBox2.Text
& "','" & TextBox3.Text & "','" & DateTimePicker1.Text & "','"
& TextBox7.Text & "')", conn)
        cmd.ExecuteNonQuery()
        MsgBox("Record inserted to Database",
MsgBoxStyle.OkOnly, "Hospital Management")
        cleardata()
        conn.close()
    End Sub
    Sub cleardata()
        TextBox1.Text = Nothing
        TextBox2.Text = Nothing
        TextBox3.Text = Nothing
        TextBox7.Text = Nothing
        ComboBox1.Text = Nothing
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
        conn.open()
        adp = New OleDbDataAdapter("select * from opt where
hno=' " & ComboBox1.Text & "'", adpx)
        adp.Fill(ds, "opt")
        TextBox2.Text = ds.Tables(0).Rows(0)(1).ToString
        TextBox3.Text = ds.Tables(0).Rows(0)(9).ToString
        conn.close()
    End Sub

    Private Sub outpatbill_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        conn.open()
        cmd = New OleDbCommand("select * from opt", conn)
        dr = cmd.ExecuteReader
        While (dr.Read)
            ComboBox1.Items.Add(dr(0).ToString)
        End While
        cmd.Dispose()
        cmd = New OleDbCommand("select max(bno) from bill",
conn)
        dr = cmd.ExecuteReader
        If dr.Read = True Then
            TextBox1.Text = dr(0) + 1.ToString
        Else
            TextBox1.Text = 1
        End If
        conn.close()
        cmd.Dispose()
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
        cleardata()
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
        outbillview.Show()
    End Sub
End Class
```

Inpatient Record Search:

```
Imports System.Data
Imports System.Data.OleDb
Public Class iptrecord
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        conn.open()
        Try
            Dim sql As String = "select * from ipt where name
= '" & TextBox1.Text & "'"
            Dim adp As OleDbDataAdapter
            adp = New OleDbDataAdapter(sql, adpx)
            Dim ds As DataSet = New DataSet
            adp.Fill(ds, "ipt")
            DataGridView1.DataSource = ds.Tables("ipt")
        Catch
            MsgBox("Record not Found", , "Hospital
Management")
        End Try
        conn.close()
    End Sub

    Private Sub iptrecord_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        TextBox1.Focus()
    End Sub
End Class
```

Consultation:

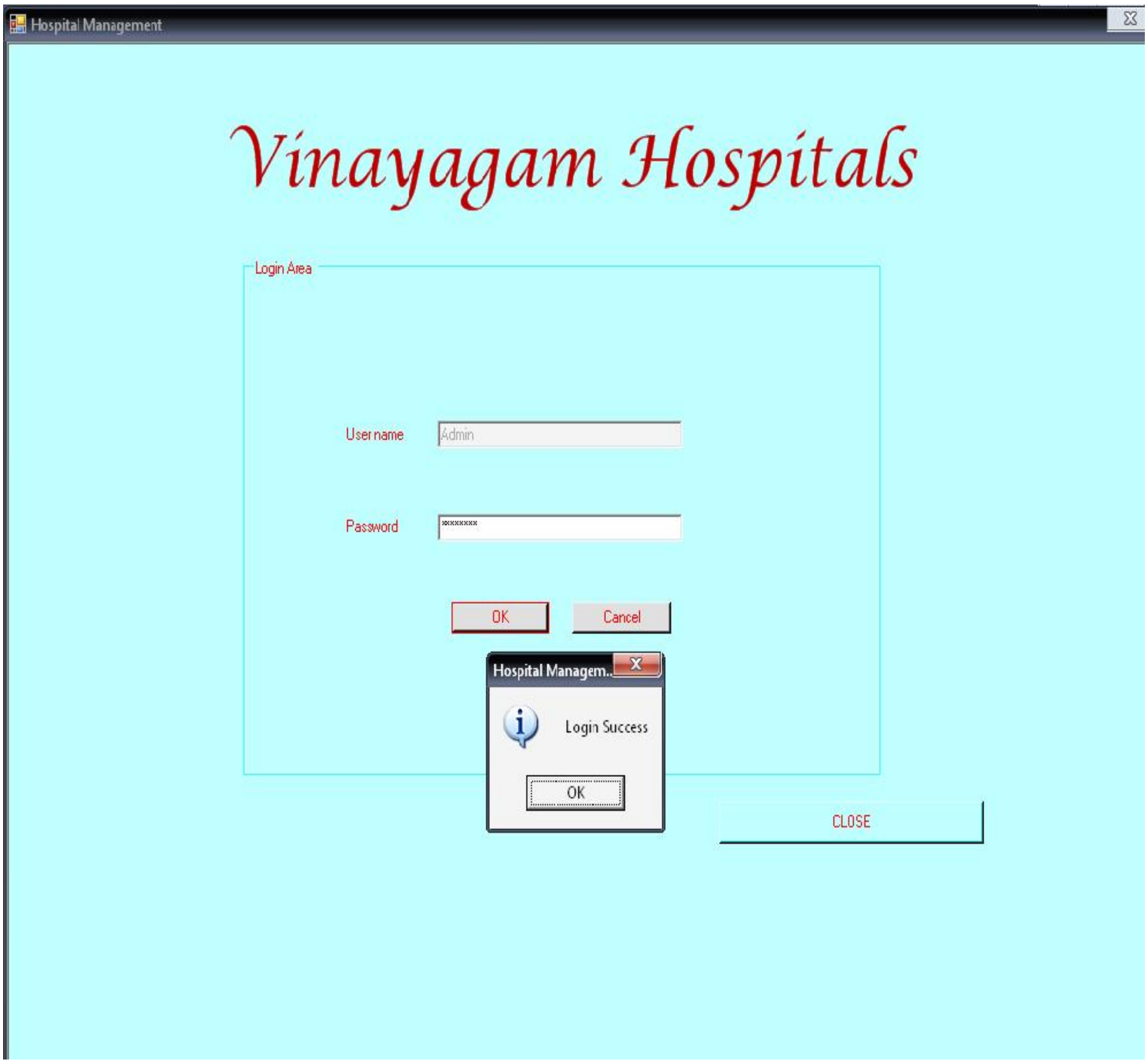
```
Imports System.Data
Imports System.Data.OleDb
Public Class consul
    Dim cmd As New OleDbCommand
    Dim adp As New OleDbDataAdapter
    Dim ds As New DataSet
    Dim dr As OleDbDataReader

    Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
        Me.Hide()
        mainmenu.Show()

    End Sub

    Private Sub consul_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        DataGridView1.Hide()
        conn.open()
        cmd = New OleDbCommand("select * from ipt", conn)
        dr = cmd.ExecuteReader
        While (dr.Read)
            ComboBox1.Items.Add(dr(0).ToString)
        End While
        cmd.Dispose()
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
        adp = New OleDbDataAdapter("select * from ipt where
mrdno=' " & ComboBox1.Text & "'", adpx)
        adp.Fill(ds, "ipt")
        TextBox2.Text = ds.Tables(0).Rows(0)(1).ToString
        TextBox3.Text = ds.Tables(0).Rows(0)(2).ToString
        ComboBox2.Text = ds.Tables(0).Rows(0)(5).ToString
        DateTimePicker1.Value = Now.Date
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        cmd = New OleDbCommand("insert into cons values('" &
ComboBox1.Text & "','" & TextBox2.Text & "','" & TextBox3.Text
& "','" & ComboBox2.Text & "','" & DateTimePicker1.Text &
"','" & TextBox5.Text & "','" & TextBox6.Text & "')", conn)
        cmd.ExecuteNonQuery()
        MsgBox("Record inserted to Database",
MsgBoxStyle.OkOnly, "Hospital Management")
        cleardata()
        cmd.Dispose()
    End Sub
    Sub cleardata()
        ComboBox1.Text = " "
        TextBox2.Text = " "
        TextBox3.Text = " "
        TextBox5.Text = " "
        TextBox6.Text = " "
        ComboBox2.Text = " "
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        cleardata()
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
        DataGridView1.Show()
        Try
            Dim sql As String = "select * from cons where
mrdno = '" & ComboBox1.Text & "'"
            Dim adp As OleDbDataAdapter
            adp = New OleDbDataAdapter(sql, adpx)
            Dim ds As DataSet = New DataSet
            adp.Fill(ds, "cons")
            DataGridView1.DataSource = ds.Tables("cons")
        Catch
            MsgBox("Record not Found", , "Hospital
Management")
        End Try
        conn.close()
    End Sub
End Class
```

Module :

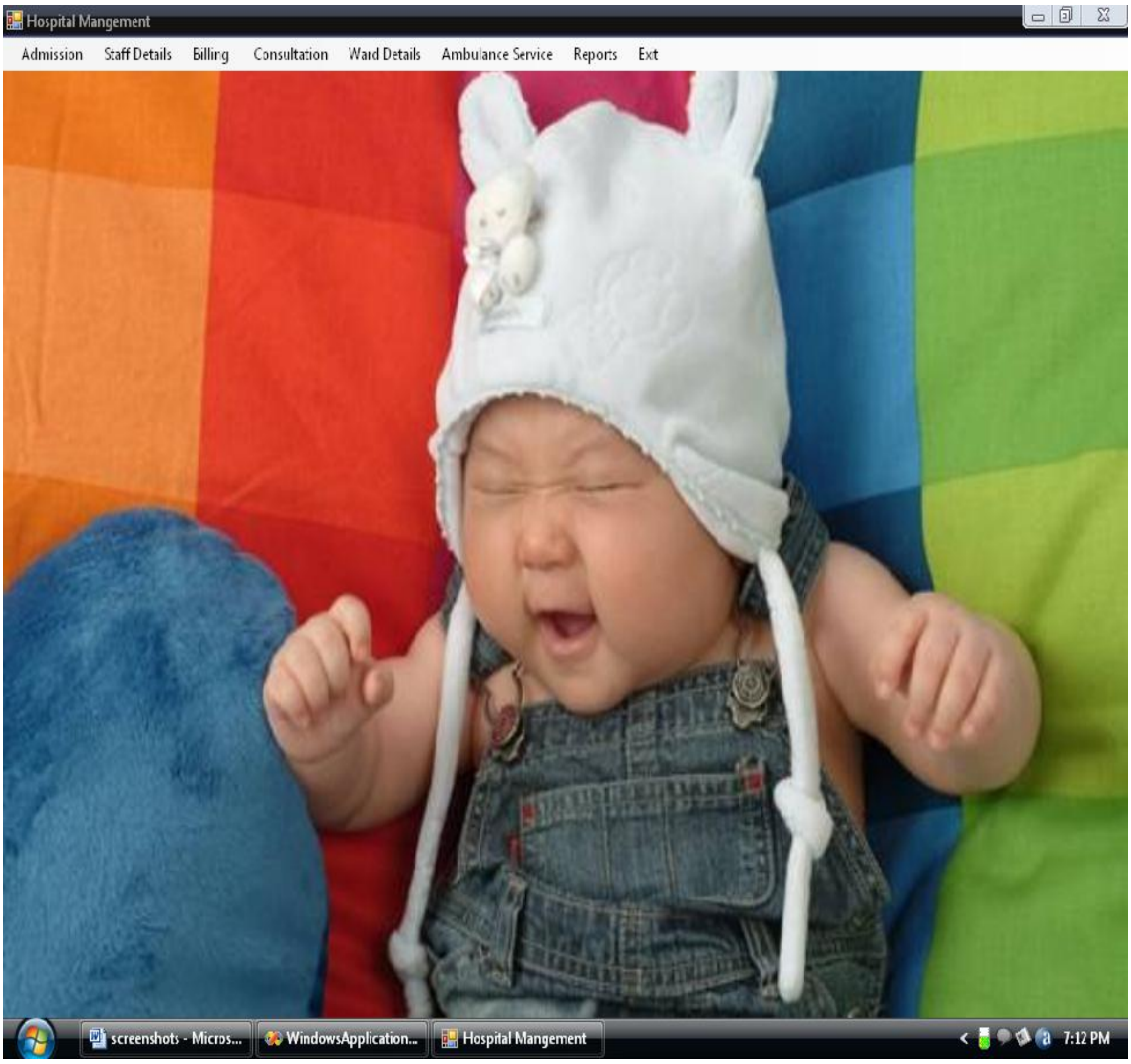
```
Imports System.Data.OleDb
Module Module1
    Public v As New OleDbConnection
    Public adpx As String = "Provider=SQLOLEDB;Data
Source=ARULSOFT;Integrated Security=SSPI;Initial
Catalog=hospital1"
    Public conn = New OleDbConnection("Provider=SQLOLEDB;Data
Source=ARULSOFT;Integrated Security=SSPI;Initial
Catalog=hospital1")
    Public x As Integer = New Integer + 1
    Public Sub opendata()
        v.ConnectionString = "Provider=SQLOLEDB;Data
Source=ARULSOFT;Integrated Security=SSPI;Initial
Catalog=hospital1"
        v.Open()
    End Sub
    Public Sub closedata()
        v.Close()
        v = Nothing
    End Sub
End Module
```

SCREENSHOTS

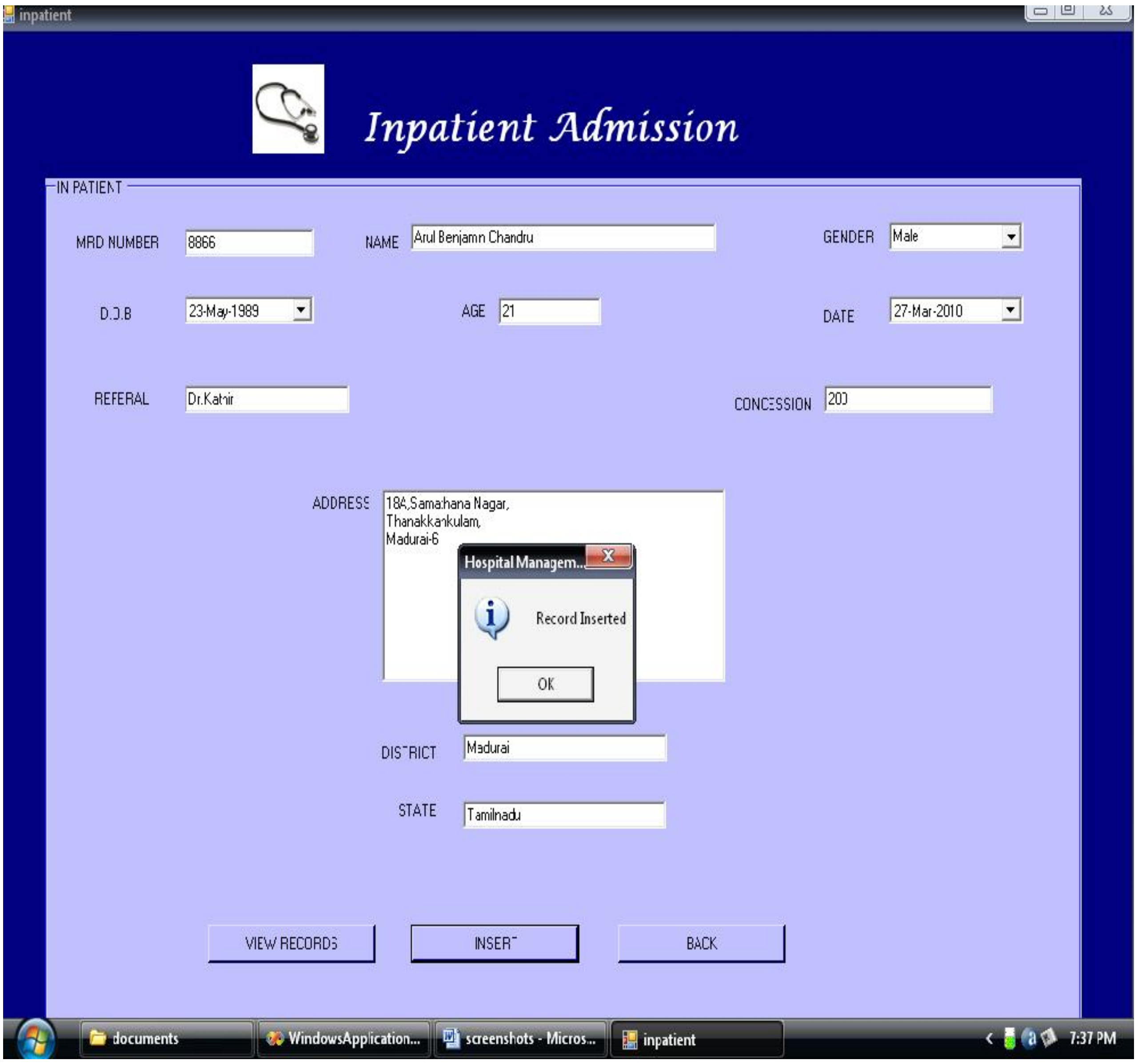
Login Screen:



Main Menu Screen:



Inpatient Admission:



inpatient

Inpatient Admission

IN PATIENT

MRD NUMBER 8866

NAME Arul Benjamn Chandru

GENDER Male

D.O.B 23-May-1989

AGE 21

DATE 27-Mar-2010

REFERAL Dr.Kathir

CONCESSION 200

ADDRESS 18A,Samathana Nagar,
Thanakkankulam,
Madurai-6

Hospital Managem...

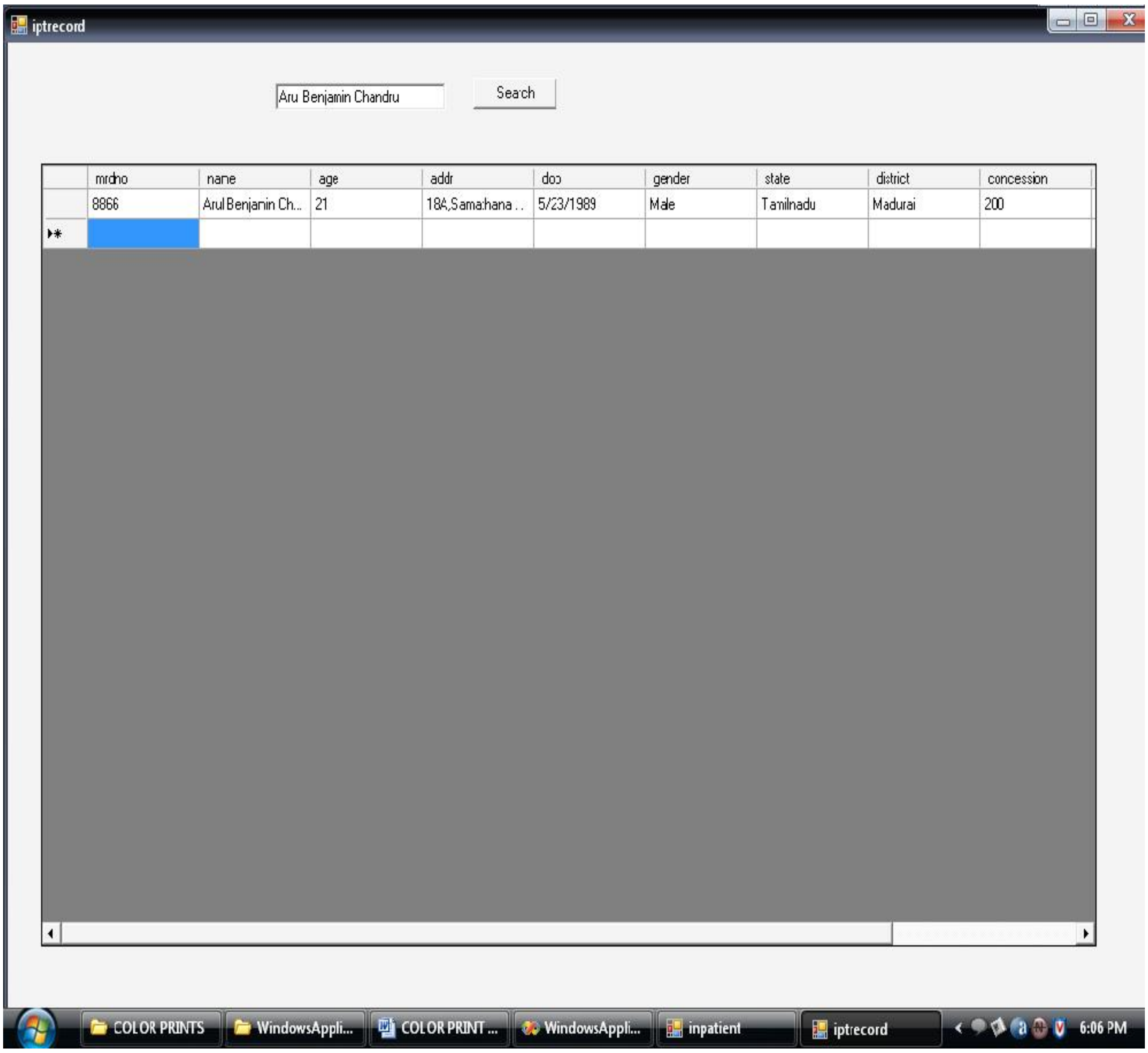
Record Inserted

OK

DISTRICT Madurai

STATE Tamilnadu

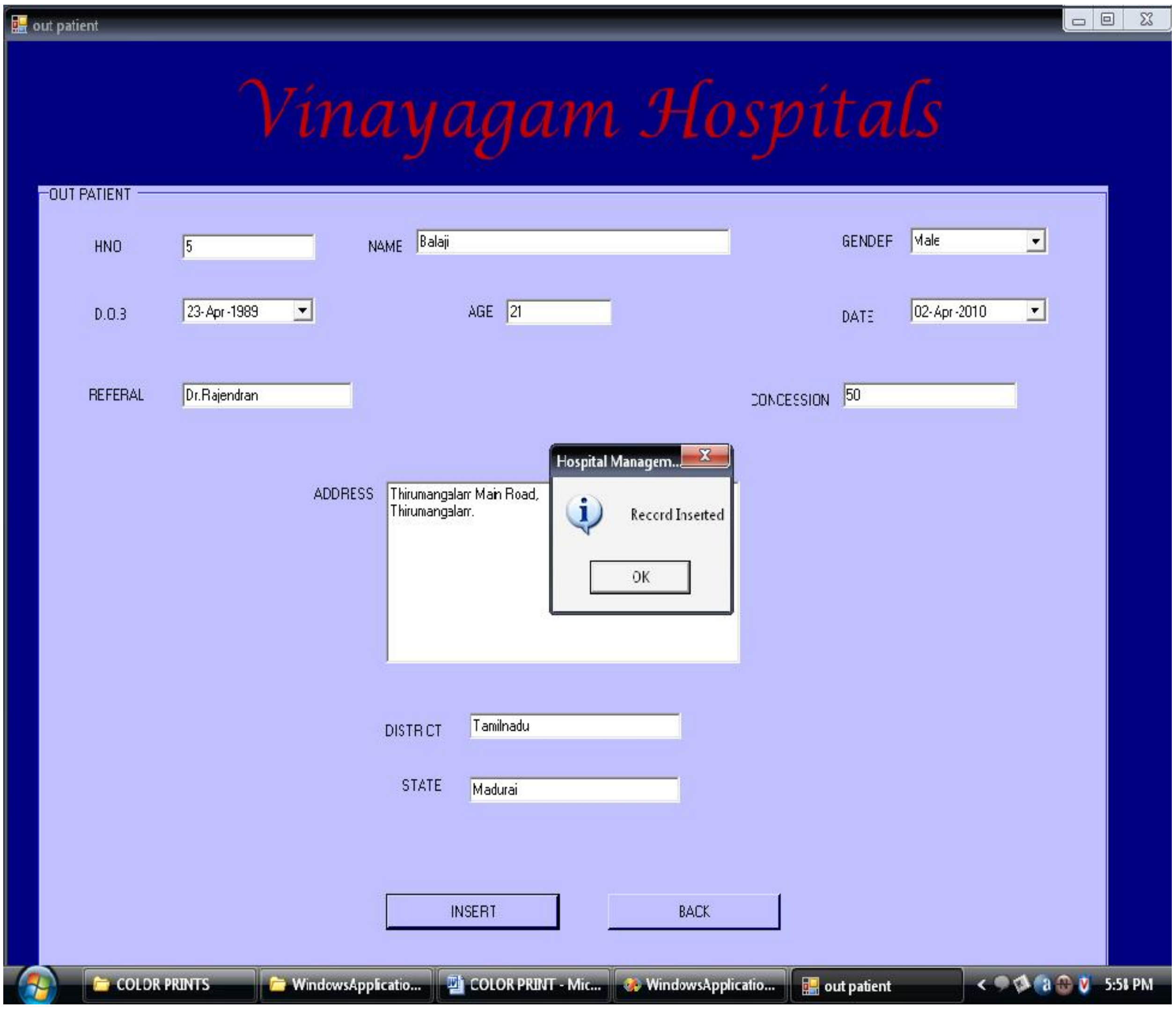
Inpatient Records Search:



Aru Benjamin Chandru

mrdno	name	age	addr	dob	gender	state	district	concession
8866	Arul Benjamin Ch...	21	18A,Samathana ...	5/23/1989	Male	Tamilnadu	Madurai	200

Outpatient Admission:



out patient

Vinayagam Hospitals

OUT PATIENT

HNO	5	NAME	Balaji	GENDER	Male
D.O.B	23-Apr-1989	AGE	21	DATE	02-Apr-2010
REFERAL	Dr.Rajendran			CONCESSION	50

ADDRESS: Thirumangalam Main Road, Thirumangalam.

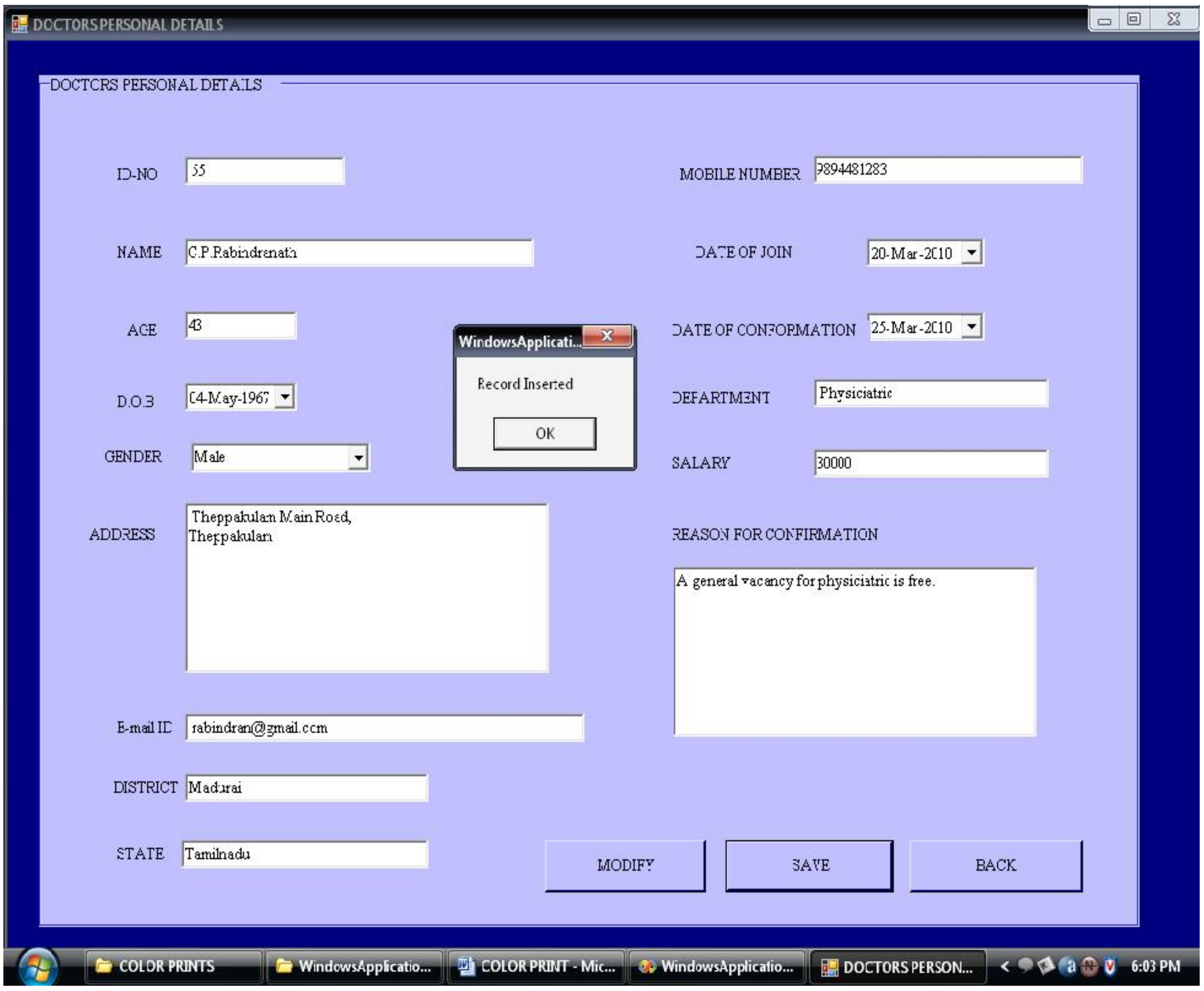
DISTRICT: Tamilnadu

STATE: Madurai

Hospital Managem...

Record Inserted

Doctor Personal Details:



DOCTORS PERSONAL DETAILS

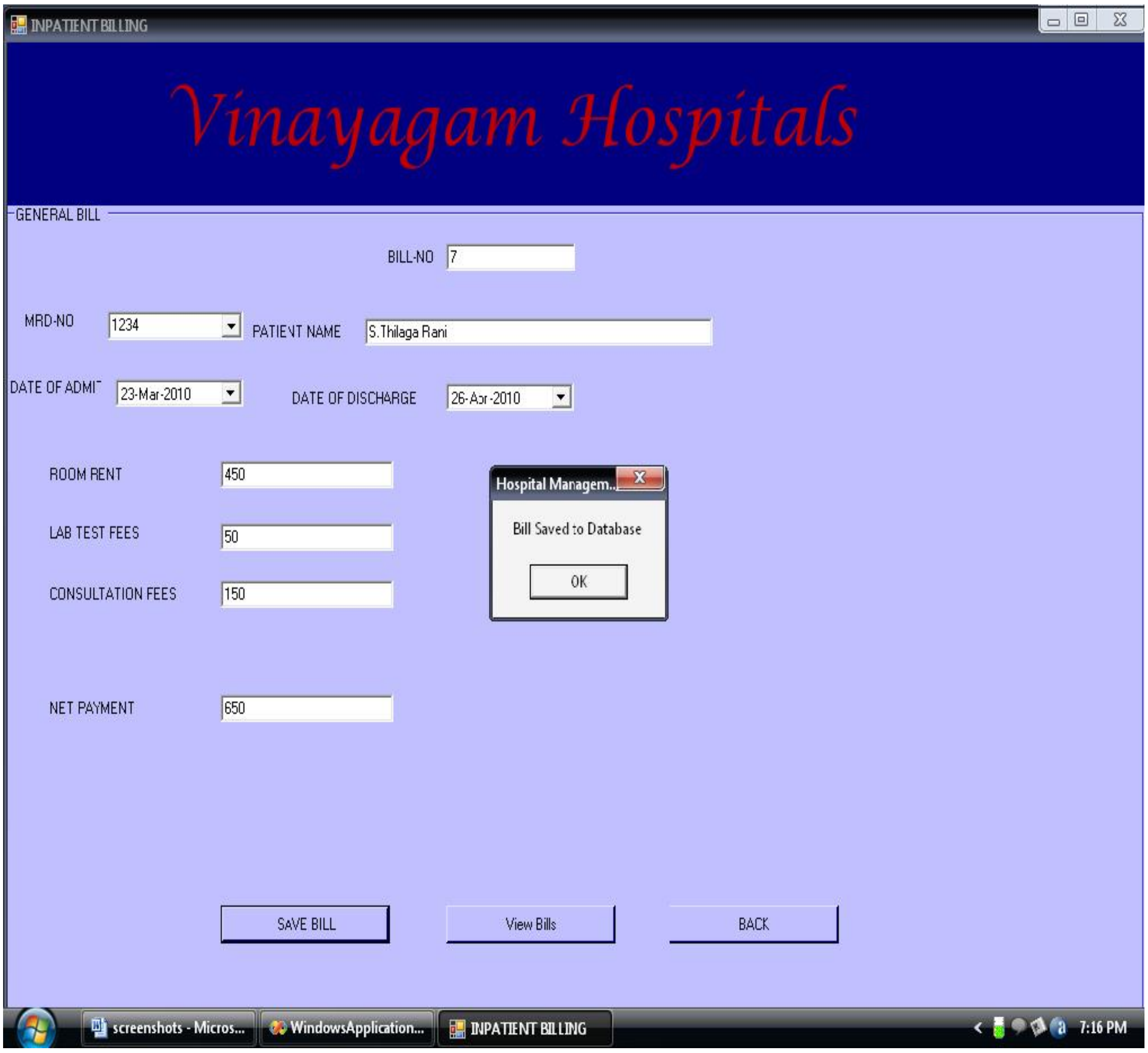
DOCTORS PERSONAL DETAILS

ID-NO	55
NAME	C.P.Rabindranath
AGE	43
D.O.B	04-May-1967
GENDER	Male
ADDRESS	Theppakulam Main Road, Theppakulam
E-mail ID	rabindran@gmail.com
DISTRICT	Madurai
STATE	Tamilnadu
MOBILE NUMBER	9894481283
DATE OF JOIN	20-Mar-2010
DATE OF CONFORMATION	25-Mar-2010
DEPARTMENT	Physiciatric
SALARY	30000
REASON FOR CONFIRMATION	A general vacancy for physiciatric is free.

WindowsApplicati...

Record Inserted

Inpatient Bill:


INPATIENT BILLING

Vinayagam Hospitals

GENERAL BILL

BILL-NO: 7

MRD-NO: 1234

PATIENT NAME: S.Thilaga Rani

DATE OF ADMIT: 23-Mar-2010

DATE OF DISCHARGE: 26-Apr-2010

ROOM RENT: 450

LAB TEST FEES: 50

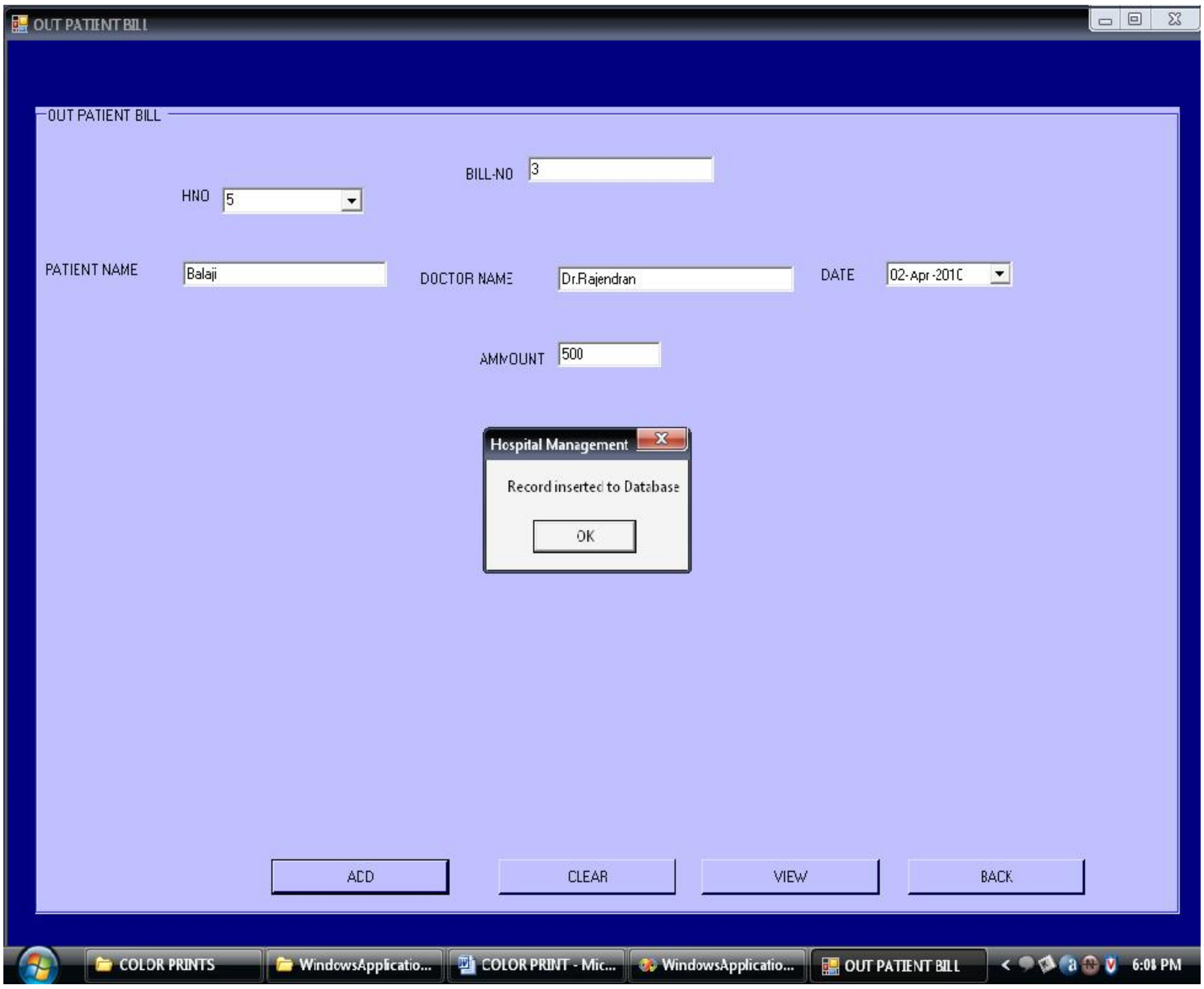
CONSULTATION FEES: 150

NET PAYMENT: 650

Hospital Managem...

Bill Saved to Database

Outpatient Bill:


OUT PATIENT BILL

OUT PATIENT BILL

BILL-NO: 3

HNO: 5

PATIENT NAME: Balaji

DOCTOR NAME: Dr.Rajendran

DATE: 02-Apr-2010

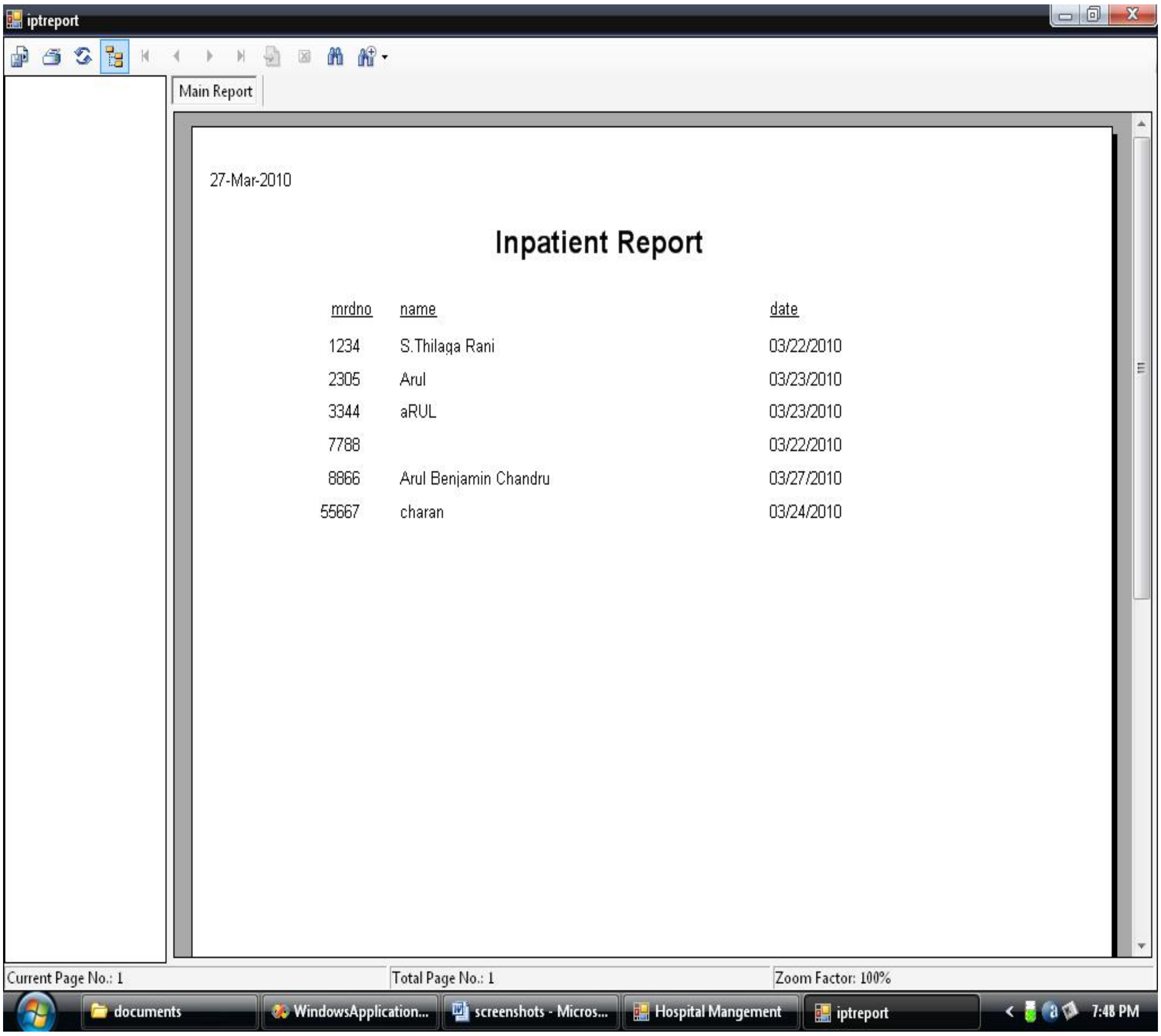
AMMOUNT: 500

Hospital Management

Record inserted to Database

OK

Inpatient Reports:

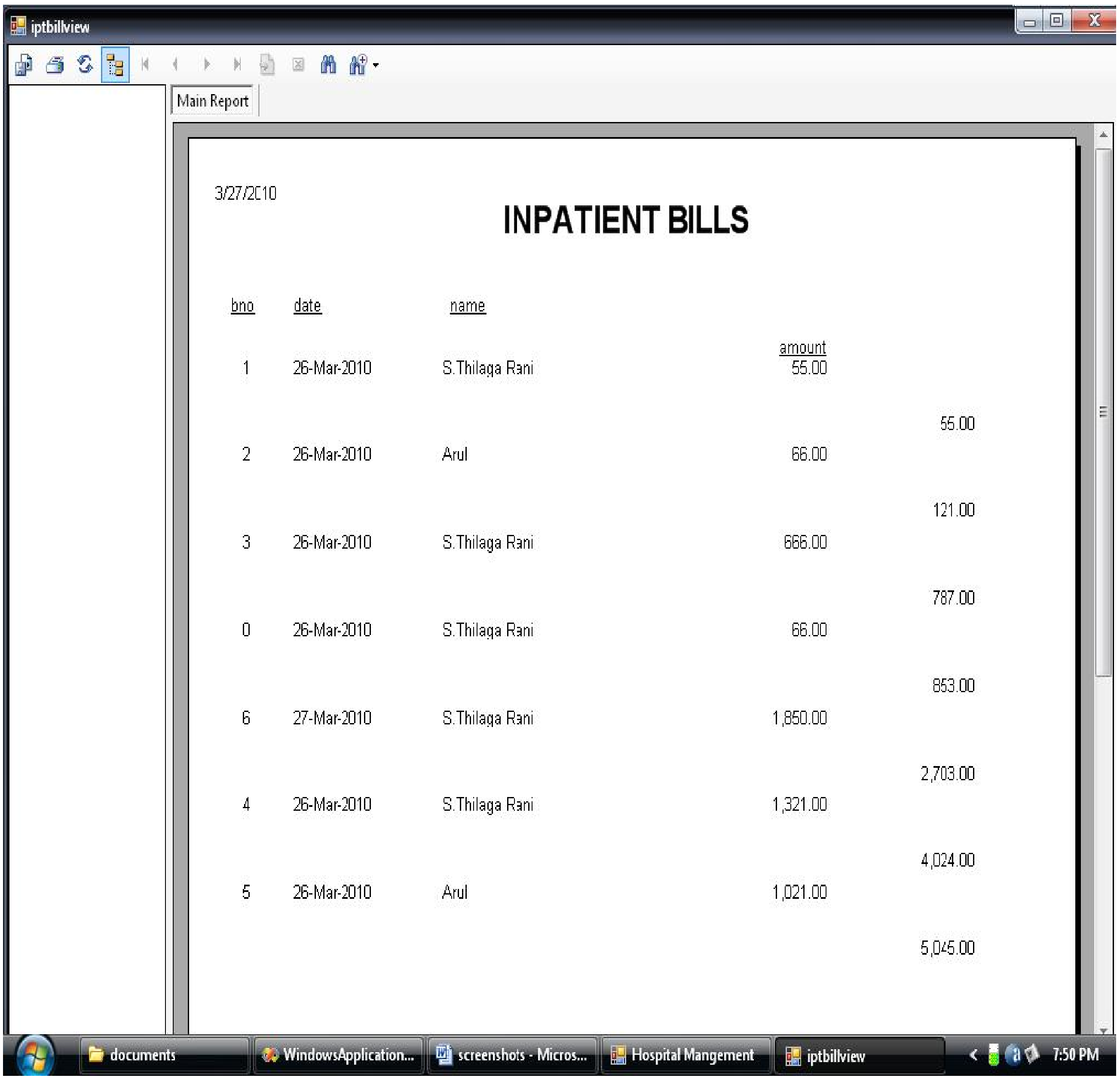


27-Mar-2010

Inpatient Report

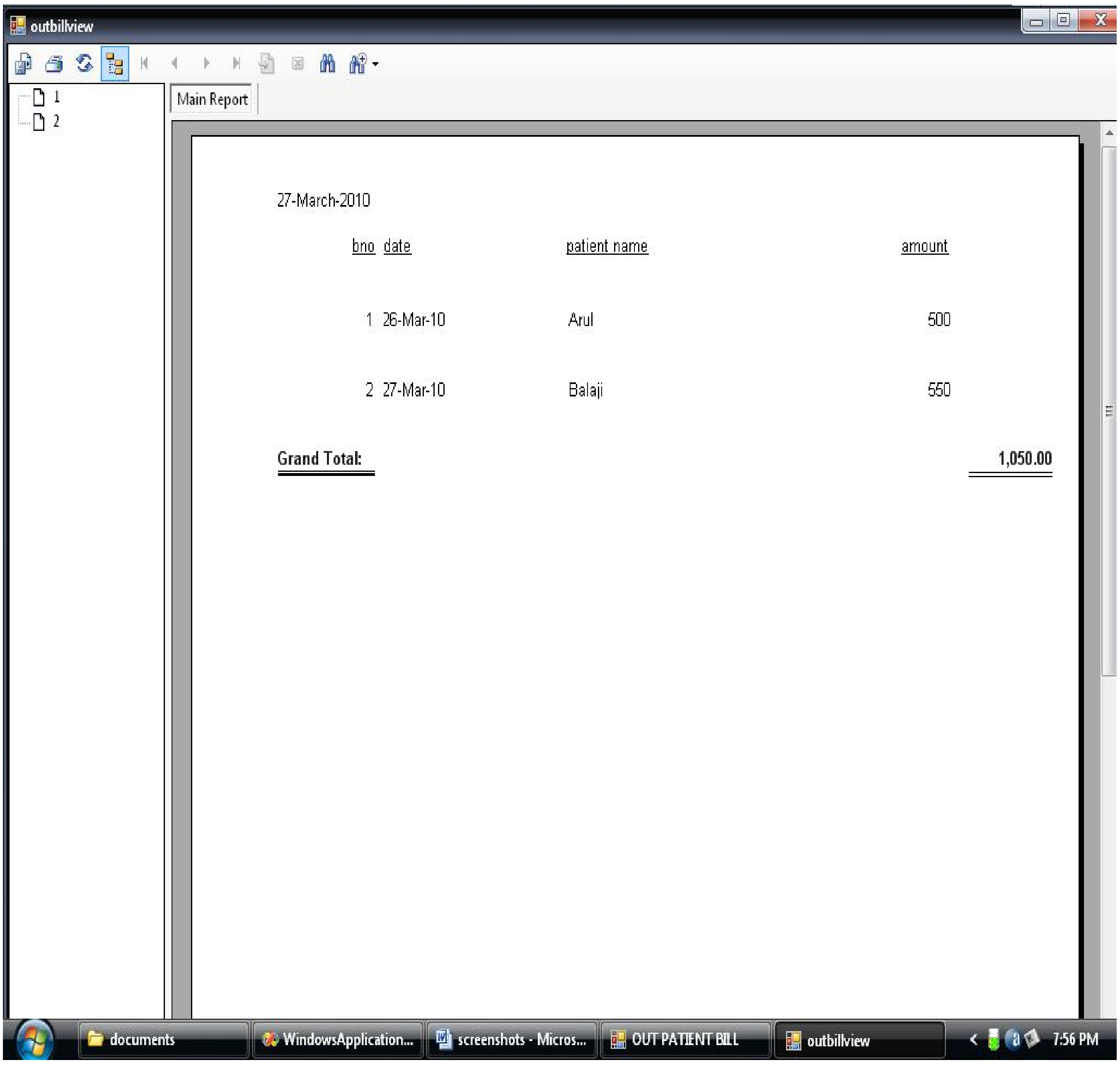
mrdno	name	date
1234	S.Thilaga Rani	03/22/2010
2305	Arul	03/23/2010
3344	aRUL	03/23/2010
7788		03/22/2010
8866	Arul Benjamin Chandru	03/27/2010
55667	charan	03/24/2010

Bills Collection Report:



```
3/27/2010
                         INPATIENT BILLS

bno   date          name                          amount
 1    26-Mar-2010   S.Thilaga Rani                 55.00
                                                                55.00
 2    26-Mar-2010   Arul                           66.00
                                                               121.00
 3    26-Mar-2010   S.Thilaga Rani                666.00
                                                               787.00
 0    26-Mar-2010   S.Thilaga Rani                 66.00
                                                               853.00
 6    27-Mar-2010   S.Thilaga Rani              1,850.00
                                                             2,703.00
 4    26-Mar-2010   S.Thilaga Rani              1,321.00
                                                             4,024.00
 5    26-Mar-2010   Arul                        1,021.00
                                                             5,045.00
```

Outpatient bills Collection Report:



27-March-2010

bno	date	patient name	amount
1	26-Mar-10	Arul	500
2	27-Mar-10	Balaji	550

Grand Total: 1,050.00

BIBILIOGRAPHY

BIBILIOGRAPHY

1. **OOP WITH MICROSOFT VISUAL BASIC.NET AND MICROSOFT VISUAL C#.NET STEP BY STEP** BY **ROBIN.A.REYNOLDS – HAERTLE – Microsoft Press 2002**
2. **THE BOOK OF VISUAL BASIC 2005.** by **Matthew MacDonald – NO STARCH PRESS**
3. **VB.NET Developer's Guide** by **Syngress Publishing, Inc**.
4. **Visual Basic .NET How to Program Second Edition** by **Deitel & Associates, Inc.,**
5. **VB.Net Language in a Nutshell** by **Steven Roman, Ron Petrusha – O'Reilly 2001**
6. **Microsoft ADO.NET Step By Step – Microsoft Press 2005**
7. **Programming Microsoft Windows with Microsoft Visual Basic .NET** by **Charles Petsold** 2002
8. **PROGRAMMING MICROSOFT VISUAL BASIC.NET** By **Francesco Baleno – Microsoft Press 2004**
9. **Microsoft Visual Basic .NET Projects for the Classroom** By **Alfred C Thompson** Distributed by **Mainfunction.com**
10. **Software Engineering , A PRACTITIONER'S APPROACH** by **Roger.S.Pressman,Ph.D. Fifth Edition**
11. **Software Engineering Concepts,** By **Richard E.Fairley – Tata McGraw – Hill Publishing Company Limited**

WEBILIOGRAPHY

WEBILIOGRAPHY

1. http://www.vbdotnetheaven.com/
2. http://www.connectionstrings.com/