International Conference on Recent Advances in Management, Architecture, Technology and Engineering
(ICRAMATE'16)
Organized by
International Journal of Advanced Research in Management, Architecture, Technology and Engineering
(IJARMATE)
21st March 2016

IDENTIFICATION OF HIGH RISK GROUP PRONE AND CORRESPONDING PREVENTIONS RECOMMENDATIONS

D.Infee[1], A.Sunitha[2], Arul Marcel Moshi.A[3]

[1]PG Scholar,Department of MCA, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur -628215,India
[2]Assistant Professor, Department of MCA, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur - 628215,India
[3] Assistant Professor, Department of Mechanical Engineering, National Engineering college, kovilpatti -628503, India

Abstract— **The health advisor used to analyse all the collected information and to provide health advice to minimize the disease. Traditionally, the administrator should maintain all the information; and the health advisor will visit nearby areas. It is felt as such a difficult task while we are in need of the relevant data immediately. Hence, this work has been planned to be a remedy and a best alternative approach. It will be very helpful to cure where the poor people affected by much disease. In the early days the hospital details had to be maintained in the manual format. It was consumed more time and retrieval the data is the toughest one. After introducing our proposed approach, all the records will be maintained by systematic format. It will consume less time and maintain records easily. Here, the health advisor will analyze all the problems in particular area and rectify those problems. It will be more helpful to maintain all the medical details in the surrounding areas too. Also, to maintain the records of sale and service manually, is a time-consuming job. With the increase in database, it will become a massive job to maintain the database. The retrieval of records of previously registered patients will be a tedious job. Lack of security for the records, anyone disarrange the records of our system.**

Index Terms— **High Risk Group Prone**

I. INTRODUCTION

A **hospital information system** (**HIS**) is an element of health informatics that focuses mainly on the administrational needs of hospitals. In many implementations, a HIS is a comprehensive, integrated information system designed to manage all the aspects of a hospital's operation, such as medical, administrative, financial, and legal issues and the corresponding processing of services. The **Hospital Records Database** is a database provided by the Wellcome Trust and UK National Archives which provides information on the existence and location of the records of UK hospitals. This includes the location and dates of administrative and clinical records, the existence of catalogues, and links to some online hospital catalogues. The website was proposed as a resource of the month by the Royal Society of Medicine in 2009. **Patient Administration Systems** (often abbreviated to PAS) developed out of the automation of administrative paperwork in healthcare organizations, particularly hospitals, and are one of the core components of a hospital's IT infrastructure. The PAS records the patient's demographics (e.g. name, home address, date of birth) and details all patient contact with the hospital, both outpatient and inpatient. PAS systems are often criticized for providing only administrative functionality to hospitals, however attempts to provide more clinical and operational functionality have often been expensive failures.

In the early days the hospital details had to be maintained in the manual format. It consumed more time and retrieval of the data was the toughest one. Now all the records can be maintained by following a systematic format. This approach will consume less time only and maintain all the required records easily and precisely. Here, the health advisor will analyze all the problems in his particular area and rectify those problems within time. It will be surely more helpful to maintain all the medical details in the surrounding areas also.

II. SYSTEM ANALYSIS

2.1 EXISTING LAYOUT

- To maintain the records of sale and service manually, is a Time-consuming job.
- With the increase in database, it will become a massive job to maintain the database.
- The retrieval of records of previously registered patients will be a tedious job.
- Lack of security for the records, anyone disarrange the records of our system.
- If someone want to check the details of the available doctors the previous system does not provide any necessary detail of this type.
- So, we can motivated all the records should maintained in a systematic format.
- It can be used to maintained all the records in very easiest format.

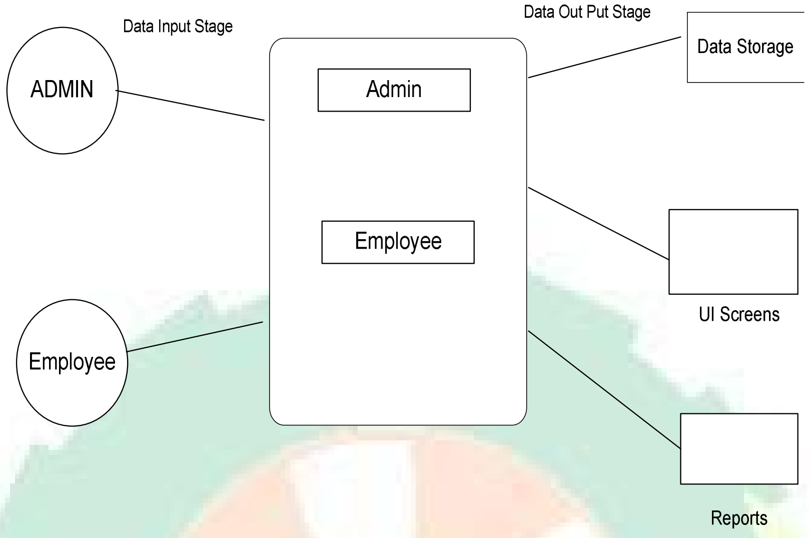


Fig 2.1: planned layout of the proposed work

2.2 DISADVANTAGES OF EXISTING SYSTEM

- Difficult to handle large number of manual records.
- Appropriate information is not gathered.
- More cost is needed.

2.3 PROPOSED SYSTEM

- The health advisor should collect all the information in a single point view.
- And the details maintained in a systematic format.
- So we can easily maintained all the patient records in short time.
- First, the health advisor analysis which is the more problematic area and what are the diseases affected by those areas.
- So we can easily maintained all the patient records in short time.

2.4 ADVANTAGES

- Complete information is gathered
- Time and cost is much reduced

2.5 REQUIREMENT SPECIFICATION

Requirement specification in system engineering is the direct result of a requirements analysis and can refer to

- Software Requirement Specification
- Hardware Requirements Specification

A software requirements specification (SRS) is a description of a software system to be developed, laying out functional and non-functional requirements, and may include a set of use cases that describe interactions the users will have with the software.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

The software requirements specification document enlists enough and necessary requirements that are required for the project development.

To derive the requirements we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

Figure 2.1 shows the overall layout of the proposed system, in which the central rectangular block will act as the Central Processing Unit (CPU). From the data input stage, the initially collected information of patients / sales and services will be fed in the created programmed systematic system. The overall control of the system will be done by the administrator. Some delegation of authorities will be there to have some controls over the core processes of the programmed system, they will be appointed by the administrator.

III. HARDWARE AND SOFTWARE REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

• PROCESSOR	:	PENTIUM III 766 MHz
• RAM	:	128 MD SD RAM
• MONITOR	:	15" COLOR
• HARD DISK	:	20 GB
• FLOPPY DRIVE	:	1.44 MB
• CDDRIVE	:	LG 52X
• KEYBOARD	:	STANDARD 102 KEYS
• MOUSE	:	3 BUTTONS

3.2 HARDWARE REQUIREMENTS

• OPERATING SYSTEM	:	WINDOWS 7
• ENVIRONMENT	:	VISUAL STUDIO .NET 2008
• .NET FRAMEWORK	:	VERSION 2.0
• LANGUAGE	:	C#.NET

- WEB SERVER :INTERNET INFORMATION SERVER 5.0
- BACK END : SQL SERVER 2005

4.1 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 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.

4.2 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

IV.SOFTWARE DESCRIPTION

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.

4.3 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.

4.4 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.

Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships

V. SYSTEM DESIGN

5.1 Data Flow Diagram

Data-flow diagrams (DFDs) were introduced and popularized for structured analysis and design. DFDs show the flow of data from external entities into the system, showed how the data moved from one process to another, as well as its logical storage.

5.2 UML Diagrams

Unified Modelling Language (UML) is a standardized general-purpose modelling language in the field of object-oriented software engineering. The Unified Modelling Language includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems.

5.3 Use Case Diagram

Use case diagrams overview the usage requirements for a system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe "the meat" of the actual requirements.

Use cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse

5.4 Class Diagram

Class diagrams are the mainstay of object-oriented analysis and design. Class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed design modeling.

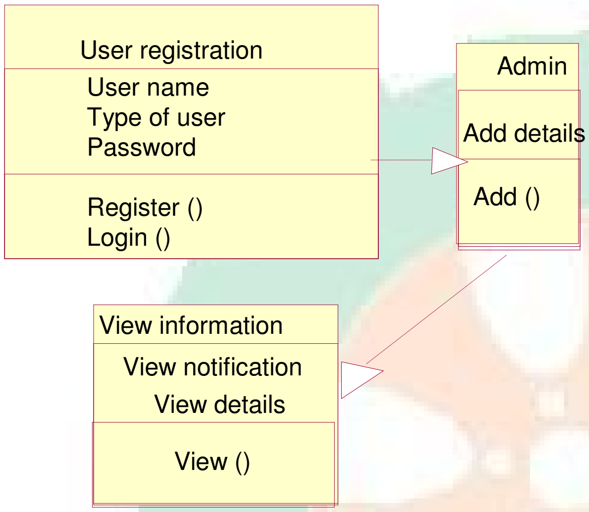


Fig 5.1: process layout

VI. MODULES DESCRIPTION

6.1 Modules

- Admin module
- Health advisor module
- Corporation module

6.2 Admin module

- An administrator should maintained the hospital details in systematic format.
- And also maintained collected the disease information.
- The administrator should allow only who can accessed in any details.
- The administrator share their details to the health advisor.

6.3 Health advisor module

- The Health advisor collect all the diseases information and patients information in near by Hospitals.
- The details should maintain only in the systematic format.
- And also the health advisor analysis which is the most diseases affected area. The health advisor
- provide health advice to people to minimize the diseases

6.4 Corporation Module

- In this Module the User can view the complete anlyzed data.
- In this module, generates the reports as input and gives details of patient on that date along with total information.

VII. TEST CASES AND RESULTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 MAINTENANCE

Correction

Correct defects in the software: defects relating to incorrect requirements, or incorrectly specifications; defects from any of the construction phases - `bugs'.

Adaption

Adapt to changes in the original software and hardware platform. E.g. simpler: MS-DOS to Windows. Complex: stand-alone to client-server.

Enhancement

Customer identifies additional requirements.

Prevention

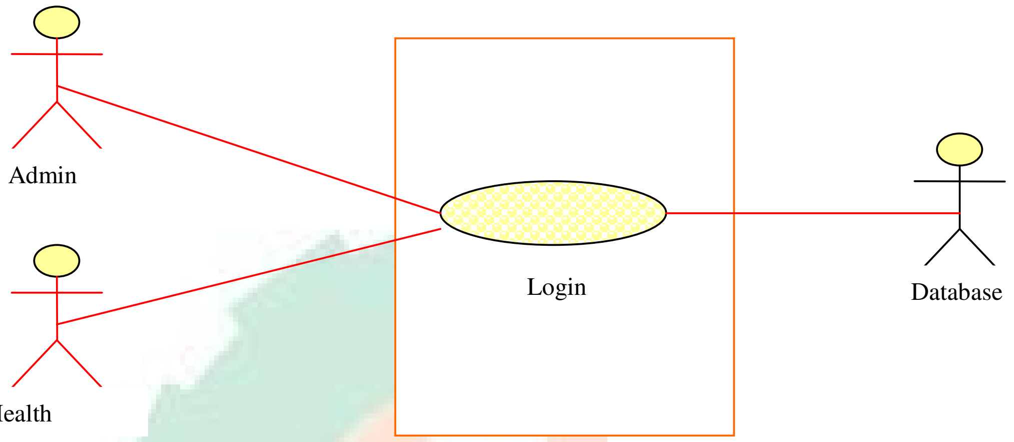
After many sets of changes, the software `deteriorates', or otherwise becomes difficult to maintain. See **Reengineering**, **Legacy Systems**.

Problems:

- Programmers do not like doing maintenance.
- Maintenance needs a project of its own. It is very uncommon to include maintenance in a (development) contract.
- Cost estimation. Normal software cost estimation is difficult enough; most estimation models do not include maintenance.
- Design deficiencies make system impossible to extend.
- Deterioration of legacy systems.

7.2 UNIT TESTING

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented



specifications and contains clearly defined inputs and expected results.

7.3 INTEGRATION TESTING

Testing in which software components, hardware components, or both together are combined and tested to evaluate interactions between them. Integration testing takes as its input modules that have been checked out by unit testing, groups them in larger aggregates, applies tests defined in an Integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

7.4 FUNCTIONAL TESTING

It is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional testing usually describes what the system does.

Functional testing does not imply that you are testing a function (method) of your module or class. Functional testing tests a slice of functionality of the whole system.

Functional testing differs from system testing in that functional testing "verifies a program by checking it against ... design document(s) or specification(s)", while system testing "validate[s] a program by checking it against the published user or system requirements".

Functional testing has many types:

- Smoke testing
- Sanity testing
- Regression testing
- Usability testing

7.5 SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

7.6 WHITEBOX TESTING

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

7.7 BLACKBOX TESTING

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

7.8 REGRESSION TESTING

Regression testing is the process of testing changes to computer programs to make sure that the older programming still works with the new changes. Regression testing is a normal part of the program development process. Test department coders develop code test scenarios and exercises that will test new units of code after they have been written. Before a new version of a software product is released, the old test cases are run against the new version to make sure that all the old capabilities still work. The reason they might not work because changing or adding new code to a program can easily introduce errors into code that is not intended to be changed. It is a quality control measure to ensure that the newly modified code still complies with its specified requirements and that unmodified code has not been affected by the maintenance activity.

7.9 ACCEPTANCE TESTING

Acceptance testing is formal testing conducted to determine whether a system satisfies its acceptance criteria and thus whether the customer should accept the system.

The main types of software testing are:

- Component.
- Interface.
- System.
- Acceptance.
- Release.

Acceptance Testing checks the system against the "Requirements". It is similar to systems testing in that the whole system is checked but the important difference is the change in focus:

Systems testing checks that the system that was specified has been delivered.

Acceptance Testing checks that the system delivers what was requested. The customer, and not the developer should always do acceptance testing. The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgment. The User Acceptance Test Plan will vary from system to system but, in general, the testing should be planned in order to provide a realistic and adequate exposure of the system to all reasonably expected events. The testing can be based upon the User Requirements Specification to which the system should conform.

7.10 TEST RESULTS

All the test cases mentioned above are passed successfully. No defects encountered.

7.11 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,

7.12 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

VII. 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 **"PROJECT NAME".** 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.

REFERENCES

1.Sheryl Bai M.L.S and Pamela Kelly "Development of a Web-based faculty publications database" (2000) Bull Med Libr Assoc. 2000 Apr; 88(2): 189-192.

2.Asabe, S. A, Oye, N. D, Monday Goji "Hospital Patient Database Managementsystem - ACase Study of General Hospital North-Bank Makurdi-Nigeria"Compusoft, An international journal of advanced computer technology, 2(3), March-2013 (Volume-II, Issue-III)

3.Mark McMurtrey "A Case study of the application of the systems Development Life cycle (SDLC) in 21st Century Health care: Something old, Something New?" Journal of the southern Association for Information Systems Volume I, Issue I, Winter 2013.