European Journal of Applied Sciences 7 (1): 32-49, 2015
ISSN 2079-2077

DOI: 10.5829/idosi.ejas.2015.7.1.1129

Design and Implementation of a Software Based Medical Diagnostic System

Umezinwa Chukwuebuka Ben, Nnaji Genevive Ada and Edokpolor Harrison Osasogie

Department of Electrical/Electronic Engineering,
Imo State Polytechnics, Umuagwo Ohaji, Imo State, Nigeria

Abstract: The importance of developing a support system for medical decision-making in the clinical field has been pointed out for improving the accuracy and objectivity of the judgment of clinicians and for early detection of disease. In this paper, we notice that signs, symptoms and clinical laboratory data are essential information which show the functional depression and failure of the ecosystem and a practical software system is presented that provides physicians with clues to clinical diagnosis. First, by analysing the medical task and the structuralization of patient information, a medical modelling method is provided, where diagnosis can be considered as the identification of a patient who corresponds to the controlled object. Secondly, by using the concept of a patient model and a disease model, a system identification algorithm is proposed and an actual system constructed for medical diagnosis is described in order to confirm the usefulness of the proposed method.

Key words: Software · Medical diagnosis · Modelling method and laboratory data

INTRODUCTION

Medical diagnosis (often simply termed diagnosis) refers both to the process of attempting to determine or identify a possible disease or disorder (and diagnosis in this sense can also be termed (medical) diagnostic procedure and to the opinion reached by this process (also being termed (medical) diagnostic opinion. From the point of view of statistics the diagnostic procedure involves classification tests.

Medical diagnosis or the actual process of making a diagnosis is a cognitive process. A clinician uses several sources of data and puts the pieces of the puzzle together to make a diagnostic impression. The initial diagnostic impression can be a broad term describing a category of diseases instead of a specific disease or condition. After the initial diagnostic impression, the clinician obtains follow up tests and procedures to get more data to support or reject the original diagnosis and will attempt to narrow it down to a more specific level. Diagnostic procedures are the specific tools that the clinicians use to narrow the diagnostic possibilities [1-4].

A diagnostic procedure (as well as the opinion reached thereby) does not necessarily involve elucidation of the etiology of the diseases or conditions of interest, that is, what caused the disease or condition. Such elucidation can be useful to optimize treatment, further specify the prognosis or prevent recurrence of the disease or condition in the future [5-8].

However, a diagnosis can take many forms. It might be a matter of naming the disease, lesion, dysfunction or disability. It might be a management-naming or prognosis-naming exercise. It may indicate either degree of abnormality on a continuum or kind of abnormality in a classification [9-10]. It's influenced by non-medical factors such as power, ethics and financial incentives for patient or doctor. It can be a brief summation or an extensive formulation, even taking the form of a story or metaphor. It might be a means of communication such as a computer code through which it triggers payment, prescription, notification, information or advice. It might be pathogenic or salutogenic. It's generally uncertain and provisional [11-13].

It should be noted that medical diagnosis in psychology or psychiatry is problematic. There are differing theoretical views toward mental conditions and few objective tests available for various major disorders (e.g., clinical depression), so a causal analysis with respect to symptomatology and disorder/disease is not always possible. As a result, most if not all mental

Corresponding Author: Umezinwa Chukwuebuka Ben, Department of Electrical/Electronic Engineering,
Imo State Polytechnics, Umuagwo Ohaji, Imo State, Nigeria.

conditions function as both symptoms and disorders. There are often functional descriptions provided for psychological disorders and these are vulnerable to circular reasoning due to the etiological fuzziness inherent of these diagnostic categories [14-16].

This project uses computer software to diagonize diseases. In computers, medical software is a significant branch of software engineering. Many medical devices that monitor or control patients are predominantly controlled by software. Medical devices are frequently regulated and must comply with local and regional laws. In the European Union, these includes the Medical Devices Directive. In the United States, the Food and Drug Administration has increased its involvement in reviewing the development of medical device software starting in the mid-1980s, where coding errors in a radiation therapy device (Therac-25) resulted in the overdose of patients. FDA is now focused on regulatory oversight on medical device software development process and system-level testing. In the recent years, 1995-2005 IEC 62304 has become the benchmark standard for the development of medical software [17-18].

Statement of the Program: Diagnosis of tropical diseases in Africa has been a long time problem, because of poverty, bad government, lack of good health facility. Most people could not even afford the available ones. But this software will go along way to eliminate these problems which includes lack of qualified manpower in the health sector.

Objective of the Study: The main objective of this study is to develop an expert system (computer software) that will mimick a domain expert (medical doctor), asks questions to d patients, diagonize and prescribe drugs to the patient.

Limitations: This project covers only the diagnosis of tropical diseases, which includes: malaria, typhoid, headache etc. It interacts with the patients, diagonizes and prescribes drugs to them based on the symptoms they experience on their body.

Significance of the Study: The importance of this project cannot be over- emphasized. This will go along way to help patients.

It reduces/ eliminates the cost of medical services. It brings good health care closer to the unreached, if deployed well. It solves the problem of lack of qualified health workers, since it mimicks the medical doctor (expert system). Accuracy and reliability in diagnosis is achieved. It will also help the government to save cost of building health centers and employing medical doctors.

Therefore, this research is so important and essential especially in regions with limited medical facilities.

Methodology and System Analysis

Methodologies

Top down Design: Top-down and bottom-up are strategies of information processing and knowledge ordering, mostly involving software, but also other humanistic and scientific theories (see systemic). In practice, they can be seen as a style of thinking and teaching. In many cases *top-down* is used as a synonym of *analysis* or *decomposition* and *bottom-up* of *synthesis*.

A top-down approach is essentially the breaking down of a system to gain insight into its compositional sub-systems. In a top-down approach an overview of the system is first formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. A top-down model is often specified with the assistance of "black boxes", these make it easier to manipulate. However, black boxes may fail to elucidate elementary mechanisms or be detailed enough to realistically validate the model.

Top-down approaches emphasize planning and a complete understanding of the system. It is inherent that no coding can begin until a sufficient level of detail has been reached in the design of at least some part of the system. The Top-Down Approach is done by attaching the stubs in place of the module. This, however, delays testing of the ultimate functional units of a system until significant design is complete [19-20].

Top-Down Approach: Practicing top-down programming has several advantages:

- Separating the low level work from the higher level abstractions leads to a modular design.
- Modular design means development can be self contained.
- Having "skeleton" code illustrates clearly how low level modules integrate.
- Fewer operations errors (to reduce errors, because each module has to be processed separately, so programmers get large amount of time for processing).

Bottom up Design: A bottom-up approach is the piecing together of systems to give rise to grander systems, thus making the original systems sub-systems of the emergent system. In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, whereby the beginnings are small but eventually grow in complexity and completeness. However, "organic strategies" may result in a tangle of elements and subsystems, developed in isolation and subject to local optimization as opposed to meeting a global purpose.

Bottom-up emphasizes coding and early testing, which can begin as soon as the first module has been specified. This approach, however, runs the risk that modules may be coded without having a clear idea of how they link to other parts of the system and that such linking may not be as easy as first thought. Re-usability of code is one of the main benefits of the bottom-up approach.

In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, whereby the beginnings are small, but eventually grow in complexity and completeness.

Object-oriented programming (OOP) is a programming paradigm that uses "objects" to design applications and computer programs.

In Mechanical Engineering with software programs such as Pro/ENGINEER, Solid works and Autodesk Inventor users can design products as pieces not part of the whole and later add those pieces together to form assemblies like building LEGO. Engineers call this piece part design.

This bottom-up approach has one weakness. We need to use a lot of intuition to decide the functionality that is to be provided by the module. If a system is to be built from existing system, this approach is more suitable as it starts from some existing modules [21-25].

Mehodology Adopted: Due to the complexity of this software project, I was careful to choose the kind of design approach to use in other to present my modules simple enough for the user to use.

Therefore I chose to use the *top down design method* which is mostly used in software development because of its simplicity and because an overview of the system is first formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. The graphical diagram of a Top down Design method is shown below. From the diagram it shows that the system starts from the supra system down to the program module.

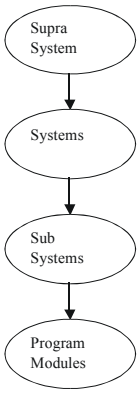

Fig. 1: A Diagram Representing the Top down design method

Organigram of Present System

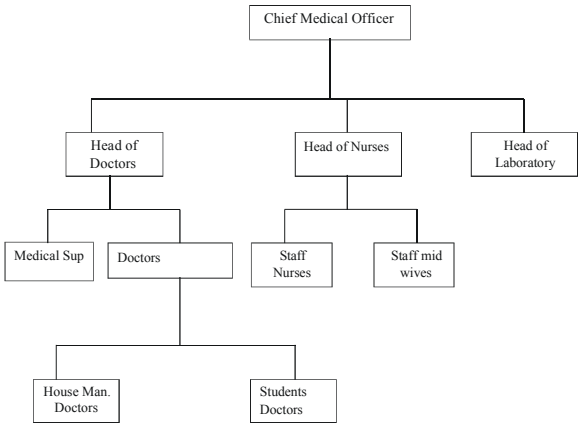

Fig. 2: over All Data Flow Diagram

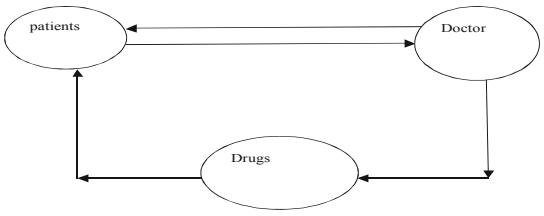

Fig. 3: Weakness of the Present System

In most hospitals in Nigeria especially Government ones, to see the doctor or other health worker for a problem is so difficult. Most times patients are made to sit down for hours in other to see a doctor. Sometimes when the patient gets to see the doctor, after examining him or her it will take another long waiting to get the result of the diagnosis and subsequent drug administration.

The current medical diagnosis system is a manual one. This makes the system so tedious and time consuming. Here, patients have to sit and wait for hours for attention.

In the manual system, human beings interact with the patient. And these humans are prone to error, fraudulent activities and loss of important data due to emotional states.

In the manual system proper documentation of patient's information and medical results are not granted, due to the vulnerability and untrusted nature of human beings who might be in duress or emotional stress when diagnosing or prescribing drugs for a patient.

Method of Data Collection: During the research work, data needed for the project was gathered from various sources. In gathering and collecting necessary data and information needed for system analysis, two major fact-finding techniques were used in this work and they are:

Primary Source: This refers to the sources of collecting original data in which the researcher made use of empirical approach such as personal interview and questionnaires with health workers and patients.

Secondary Source: The secondary data were obtained by the researcher from magazines, Journal, Newspapers, Library source and Internet downloads. The data collected from this means have been covered in literature review in the chapter two of the project.

Oral Interview: This was done between the researchers, many medical doctors from different hospitals and health center and patients. Also various departmental heads in a hospital were interviewed. Reliable facts were got based on the questions posed to the medical staffs and patients by the researcher.

Study of Manuals: Manuals and report based on different information frequently asked by doctors when diagnosing a patient and the patient responses were studied and a lot of information concerning the system in question was obtained. This information was gathered and information relating to them and other requirements were also obtained.

Evaluation of Forms: Some forms that are necessary and available were assed. These include patient's registration form, Drug prescription form, payment receipts, etc. These forms help in the design of the new system.

Objectives of the Existing System: The main objective of this project software is to develop a system (software program) that will interact with patients, retrieve relevant information from the patient; which includes: symptoms, health record and other body conditions. This system analyses the inputs from the patients and prescribes drugs for the patients and also keeps accurate record of the patients health status.

Input Analysis: The data and information inputted by the patients which include symptoms, body conditions and other health conditions form the input system that has to be analysed to get solutions to diseases through drug prescription. The symptoms options were implemented with radio buttons.

Process Analysis: The information input made by the patients are collected and analysed, hence the desired information is outputted for the patient. The input includes: disease symptoms, body conditions and other relevant health inputs.

Output Analysis: The output from the system is the result of the diagnosis of the patient, the drugs prescribed by the system and the health status of the patients after the diagnosis by the system. The result will be sent to:

- Pharmacy
- Patients medical Record.

Weakness of the Present System: Due to the manual means being used by the hospitals and medical centers to diagonize and prescribe drugs, lots of problems has been experience for decades. Some of them include

- Delay in processing patients lab result
- Unavailability of some key staff while processing lab results, which leads to patients repeatedly visiting the health center in other to see his/her's lab result or to collect drugs.

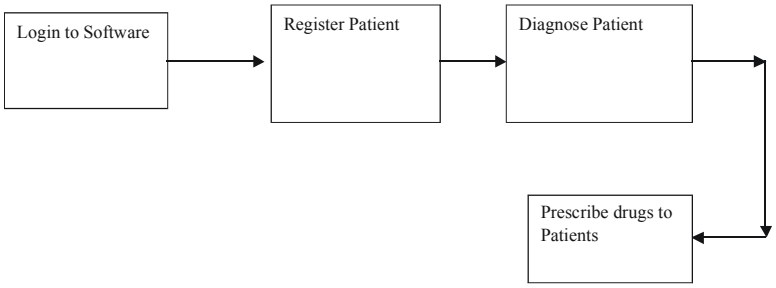

Fig. 4: Systems Design

- Loss of vital documents as the filing system is manual
- Damage of documents due to fire incident.
- Illegal alteration of health record or lab results by fraudulent staff leading to insecurity.
- Takes a lot of time to retrieve diagnosis result or to prescribe drugs.

High Level Model of the Proposed System: The new system is designed to solve problems affecting the manual system in use. It is designed to be so efficient thereby relieving both the patients and the medical personnel from much stress as experienced in the manual system.

This system will do the analyzing and storing of medical and drug information either automatically or interactively.

The proposed system will also have some other feature like:

- Accuracy in the handling of data.
- Fast rate of operation and excellent responses time
- Flexibility (i.e.) it can be accessed at any time.
- Easy way of back up or duplicating data in diskettes in case of data loss.
- Better storage of medical record and faster retrieval system.
- Accessibility from any part of the world if the software is placed on the internet

Objectives of Design: The Objective of the software project include:

- Develop an expert system for patients diagnosis
- Develop and MIS for patients data and information
- Develop a database for drugs and different disease.
- Keep track of patients health record and drugs prescribed.

Database Specification: Due to the size of the data handled by this software, I decided to use the Microsoft Access as my back- end database software, Microsoft Office Access, previously known as Microsoft Access, is a pseudo-relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately. As of May 2010 the current version is Microsoft Office Access 2007; Microsoft Access 2010 is in beta as of May 10, 2010.

Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in:

- Other Access databases
- Excel
- SharePoint lists
- Text
- XML
- Outlook
- HTML
- dBase
- Paradox
- Lotus 1-2-3
- ODBC-compliant data containers, including:
- Microsoft SQL Server
- Oracle
- MySQL
- PostgreSQL

Software developers and data architects can use Microsoft Access to develop application software and "power users" can use it to build simple applications. Like other Office applications, Access is supported by Visual Basic for Applications, an object-oriented programming language that can reference a variety of objects including

DAO (Data Access Objects), ActiveX Data Objects and many other ActiveX components. Visual objects used in forms and reports expose their methods and properties in the VBA programming environment and VBA code modules may declare and call Windows operating-system functions.

Microsoft released Access version 1.0 on 13 November 1992 and an Access 1.1 release in May 1993 to improve compatibility with other Microsoft products and include the Access Basic programming language.

Microsoft specified the minimum hardware requirements for Access v2.0 as: Microsoft Windows v3.1 with 4 MB of RAM required, 6 MB RAM recommended; 8 MB of available hard disk space required, 14 MB hard disk space recommended. The product shipped on seven 1.44 MB diskettes. The manual shows a 1993 copyright date.

Originally, the software worked well with relatively small databases but testing showed that some circumstances caused data corruption. For example, file sizes over 10 MB proved problematic (note that most hard disks held less than 500 MB at the time this was in wide use) and the *Getting Started* manual warns about a number of circumstances where obsolete device drivers or incorrect configurations can cause data loss. With the phasing out of Windows 95, 98 and ME, improved network reliability and Microsoft having released 8 service packs for the Jet.

Database Engine, the reliability of Access databases has been improved and it supports both more data and a larger number of users.

With Office 95, Microsoft Access 7.0 (a.k.a "Access 95") became part of the Microsoft Office Professional Suite, joining Microsoft Excel, Word and PowerPoint and transitioning from Access Basic to Visual Basic for Applications (VBA). Since then, Microsoft has released new versions of Microsoft Access with each release of Microsoft Office. This includes Access 97 (version 8.0), Access 2000 (version 9.0), Access 2002 (version 10.0), Access 2003 (version 11.5) and Access 2007 (version 12.0).

The native Access database format (the Jet MDB Database) has also evolved over the years. Formats include Access 1.0, 1.1, 2.0, 7.0, 97, 2000, 2002 and 2007. The most significant transition was from the Access 97 to the Access 2000 format; which is not backward compatible with earlier versions of Access. At the time of this writing, all newer versions of Access support the Access 2000 format. New features were added to the Access 2002 format which can be used by Access 2002, 2003, 2007 and 2010.

In Access 2007, a new database format was introduced: ACCDB. The ACCDB supports complex data types such as multivalue and attachment fields. These new field types are essentially recordsets in fields and allow the storage of multiple values in one field.

Prior to the introduction of Access, the desktop database market was dominated by Borland with their Paradox and dBase programs and FoxPro. Microsoft Access was the first mass market database program for Windows. With the purchase of FoxPro and incorporating its Rushmore query optimization routines into Access, Microsoft Access quickly became the dominant database for Windows effectively eliminating the competition which failed to transition from the MS-DOS world.

Access's initial codename was Cirrus; the forms engine was called Ruby. This was before Visual Basic - Bill Gates saw the prototypes and decided that the BASIC language component should be co-developed as a separate expandable application, a project called Thunder. The two projects were developed separately as the underlying forms engines were incompatible with each other; however, these were merged together again after VBA.

Access was also the name of a communications program from Microsoft, meant to compete with ProComm and other programs. This proved a failure and was dropped.[2] Years later, Microsoft reused the name for its database software.

Uses: Microsoft Access is used to create simple database solutions. Access tables support a variety of standard field types, indices and referential integrity. Access also includes a query interface, forms to display and enter data and reports for printing. The underlying Jet database, which contains these objects, is multiuser-aware and handles record-locking and referential integrity including cascading updates and deletes.

Simple tasks can be automated through macros with point-and-click options. Microsoft Access is very popular among non-programmers who can create visually pleasing and relatively advanced solutions on their own. It is also easy to place a database on a network and have multiple users share and update data without overwriting each other’s work. Data is locked at the record level which is significantly different from Excel which locks the entire spreadsheet.

Microsoft offers a wide range of template databases within the program and for download from their website. These options are available upon starting Access and allow users to quickly use and enhance a database with pre-defined tables, queries, forms, reports and macros. Popular templates include tracking contacts, assets, issues, events, projects and tasks. Templates do not include VBA code.

Microsoft Access also offers the ability for programmers to create solutions using the programming language Visual Basic for Applications (VBA), which is similar to Visual Basic 6.0 (VB6) and used throughout the Microsoft Office programs such as Excel, Word, Outlook and

PowerPoint. Most VB6 code including the use of Windows API calls, can be used in VBA. Power users and developers can extend basic end-user solutions to a professional solution with advanced automation, data validation, error trapping and multi-user support.

Database solutions created entirely in Microsoft Access are well suited for individual and workgroup use across a network. The number of simultaneous users that can be supported depends on the amount of data, the tasks being performed, level of use and application design. Generally accepted limits are solutions with 1 GB or less of data (Access supports up to 2 GB) and 50 or fewer simultaneous users. This is appropriate for workgroup and department solutions where the total number of users number a few hundred.

Applications that simply view data or have simple data entry can support considerably more users. Applications that run complex queries or analysis across large datasets would naturally require greater bandwidth and memory. Microsoft Access is designed to scale to support more data and users by linking to multiple Access databases or using a back-end database like Microsoft SQL Server. With the latter design, the amount of data and users can scale to enterprise-level solutions.

Microsoft Access' role in web development prior to version 2010 is limited. User interface features of Access, such as forms and reports, only work in Windows. The Microsoft Jet Database Engine, core to Access, can be accessed through technologies such as ODBC or OLE DB.

The data (i.e., tables and queries) can be accessed by web-based applications developed in ASP.NET, PHP, or Java. Many ISPs offer Microsoft Access as a data storage option.

Access 2010 will allow forms and reports to be published to web sites using what is called "access web services" that runs on Sharepoint software. These web based forms and reports run in any standard browser. The resulting web forms and reports when run in the browser don't require any ActiveX or add-ins like Silverlight. Thus, the resulting application can be used by running Firefox on Ubuntu Linux, for example.

In enterprise environments, Microsoft Access is particularly appropriate for meeting end-user database needs and for rapid application development. Microsoft Access is easy enough for end users to create their own queries, forms and reports, laying out fields and groupings, setting formats, etc. This frees up the professional developers to focus on more complex portions of the application.

A compiled MDE or ACCDE version of an Access database can be created to prevent users from getting to the design surfaces to modify module code, forms and reports. This is often used in environments where end-user modifications are discouraged or the application's code should be kept private.

Microsoft offers a runtime version of Microsoft Access 2007 for download. This allows people to create Access solutions and distribute it for use by non-Microsoft Access owners (similar to the way DLLs or EXEs are distributed). Unlike the regular version of Access, the runtime version allows users to use the Access application but they cannot use its design surfaces.

Microsoft also offers developer extensions for download to help distribute Access applications, create database templates and integrate source code control with Microsoft Visual sourcesave.

Features: Users can create tables, queries, forms and reports and connect them together with macros. Advanced users can use VBA to write rich solutions with advanced data manipulation and user control.

The original concept of Access was for end users to be able to “access” data from any source. Other uses include: the import and export of data to many formats including Excel, Outlook, ASCII, dBase, Paradox, FoxPro, SQL Server, Oracle, ODBC, etc. It also has the ability to link to data in its existing location and use it for viewing, querying, editing and reporting. This allows the existing data to change and the Access platform to always use the latest data. It can perform heterogeneous joins between

data sets stored across different platforms. Access is often used by people downloading data from enterprise level databases for manipulation, analysis and reporting locally.

There is also the Jet Database format (MDB or ACCDB in Access 2007) which can contain the application and data in one file. This makes it very convenient to distribute the entire application to another user, who can run it in disconnected environments.

One of the benefits of Access from a programmer's perspective is its relative compatibility with SQL (structured query language) — queries can be viewed graphically or edited as SQL statements and SQL statements can be used directly in Macros and VBA Modules to manipulate Access tables. Users can mix and use both VBA and "Macros" for programming forms and logic and offers object-oriented possibilities. VBA can also be included in queries.

Microsoft Access offers parameterized queries. These queries and Access tables can be referenced from other programs like VB6 and. NET through DAO or ADO. From Microsoft Access, VBA can reference parameterized stored procedures via ADO.

The desktop editions of Microsoft SQL Server can be used with Access as an alternative to the Jet Database Engine. This support started with MSDE (Microsoft SQL Server Desktop Engine), a scaled down version of Microsoft SQL Server 2000 and continues with the SQL Server Express versions of SQL Server 2005 and 2008.

Microsoft Access is a file server-based database. Unlike client–server relational database management systems (RDBMS), Microsoft Access does not implement database triggers, stored procedures, or transaction logging. Access 2010 does now include table-level triggers and stored procedures built into the ACE data engine. Thus a Client-server database system is not a requirement for using stored procedures or table triggers with Access 2010. Tables, queries, Forms, reports and Macros can now be developed specifically for web base application in Access 2010. Integration with Microsoft SharePoint 2010 is also highly improved.

Development: All database tables, queries, forms, reports, macros and modules are stored in the Access Jet database as a single file.

For query development, Access offers a Query Designer, a graphical user interface that allows users to create queries without knowledge of the SQL programming language. In the Query Designer, users can "show" the data sources of the query (which can be tables or queries) and select the fields they want returned by clicking and dragging them into the grid. Joins can be created by clicking and dragging fields in tables to fields in other tables. Access allows users to view and manipulate the SQL code if desired. Any Access table, including linked tables from different data sources, can be used in a query.

Access also supports the creation of Pass-Through queries. These are queries that can be linked to external data sources through the use of ODBC connections on the local machine. This enables users to interact with data stored outside the Access programme without using linked Tables. The Pass-Through queries are written using the SQL syntax supported by the external data source.

When developing Reports that are linked to Queries placing or moving items in the design view of the Report, Access runs the linked query in the background on any placement or movement of an item in that Report. If the Report is linked to a Query that takes a long time to return records this forces you to wait until the query has run before you can add/edit or move the next item in the Report (this feature cannot be turned off).

Non-programmers can use the macro feature to automate simple tasks through a series of drop down selections. Macros allow users to easily chain commands together such as running queries, importing or exporting data, opening and closing forms, previewing and printing reports, etc. Macros support basic logic (IF conditions) and the ability to call other macros. Macros can also contain sub-macros which are similar to subroutines. In Access 2007, macros are significantly enhanced with the inclusion of error handling and temporary variable support. Access 2007 also introduces embedded macros that are essentially properties of an object's event. This eliminates the need to store macros as individual objects. Macros however, are limited in their functionality by a lack of programming loops and advanced coding logic. Most professional Access developers use the VBA programming language for a richer and more powerful development environment. The programming language available in Access is, as in other products of the Microsoft Office suite, Microsoft Visual Basic for Applications, which is nearly identical to Visual Basic 6.0 (VB6). VBA code can be stored in modules and code behind forms and reports. Modules can also be classes.

To manipulate data in tables and queries in VBA, two database access libraries of COM components are provided: the Data Access Objects (DAO), which is included in Access and Windows and evolved to ACE in Microsoft Access 2007 for the ACCDE database format and ActiveX Data Objects ActiveX Data Objects (ADO). Beside DAO and ADO, developers can also use OLE DB and ODBC for developing native C/C++ programs for Access. For ADPs and the direct manipulation of SQL Server data, ADO is required. DAO is most appropriate for managing data in Access/Jet databases and the only way to manipulate the complex field types in ACCDB tables.

In the database container or navigation pane of Access 2007, the system automatically categorizes each object by type. Many Access developers use the Leszynski naming convention, though this is not universal; it is a programming convention, not a DBMS-enforced rule. It is particularly helpful in VBA where references to object names may not indicate its data type (e.g. tbl for tables, qry for queries).

Microsoft Access is most often used for individual and workgroup projects (the Access 97 speed characterization was done for 32 users). Since Access 97 and with Access 2003 and 2007, Microsoft Access and hardware have evolved significantly. Databases under 1 GB in size (which can now fit entirely in RAM) and 50 simultaneous users are well within the capabilities of Microsoft Access. Of course, performance depends on the database design and tasks. Disk intensive work such as complex searching and querying are the most time consuming.

As data from a Microsoft Access database can be cached in RAM, processing speed may be substantially better when there is only a single user or if the data is not changing. In the past, the effect of packet latency on the record locking system caused Access databases to be too slow on Virtual Private Network (VPN) or Wide Area Network (WAN) against a Jet database. This is less of an issue now with broadband connections. Performance can also be enhanced if a continuous connection is maintained to the back end database throughout the session rather than opening and closing it for each table access. If Access Database performance over VPN or WAN suffers, then Microsoft Terminal Services is an effective solution. Access databases linked to SQL Server or Access Data Projects work great over VPN and WAN.

Split Database Architecture: Microsoft Access applications can adopt a split-database architecture. The database can be divided into a front end database that contains the application objects (queries, forms, reports, macros and modules) and is linked to tables stored in a back end shared database containing the data. The "back-end" database can be stored in a location shared by many users, such as a file server. The 'front-end' database is distributed to each user's desktop and linked to the shared database. Using this design, each user has a copy of Microsoft Access installed on their machine along with their application database. This reduces network traffic since the application is not retrieved for each use and allows the front end database to contain tables with data that is private to each user for storing settings or temporary data. This split database design also allows development of the application independent of the data. When a new version is ready, the front end database is replaced without impacting the data database. Microsoft Access has two built-in utilities, Database Splitter and Linked Table Manager, to facilitate this architecture.

Linked tables in Access use absolute paths rather than relative paths, so the development environment either has to have the same path as the production environment or a "dynamic-linker" routine can be written in VBA.

This is not an economical setup across slow networks, or large organizations separated by great distances, as this will result in excessive lag to database users. SQL backend should be considered in these circumstances.

Access to SQL Server Upsizing (SQL as a Backend): To scale Access applications to enterprise or web solutions, a recommended technique is to migrate to Microsoft SQL Server or equivalent server database. A client–server design significantly reduces maintenance and increases security, availability, stability and transaction logging.

Access includes an Upsizing Wizard that allows users to upsize their databases to Microsoft SQL Server an ODBC client–server database. An additional solution, the SQL Server Migration Assistant for Access (SSMA), is also available for download from Microsoft.

A variety of upsizing options are available. After migrating the data and queries to SQL Server, the MDB/ACCDB Access database can be linked to the database. This is the easiest migration and most appropriate if the user does not have rights to create objects such as stored procedures on SQL Server. Retrieving data from linked tables is optimized to just the records needed, but are not as efficient for multi-table joins that may require copying the whole table across the network.

Access databases can also be converted to an Access Data Projects (ADP) which is tied directly to one SQL Server database. ADPs support the ability to directly create and modify SQL Server objects such as tables, views, stored procedures, views and SQL Server constraints. The views and stored procedures can significantly reduce the network traffic for multi-table joins. Fortunately, SQL Server supports temporary tables and links to other data sources beyond the single SQL Server database.

Finally, some Access databases are completely replaced by another solution such as ASP.NET or Java once the data is converted.

In many cases, hybrid solutions are created where web interfaces are built by developers using ASP.NET, while administrative or reporting features that don't need to be distributed to everyone and/or change often, are kept in Access for information workers to maintain.

While all Access data can migrate to SQL Server, some queries cannot migrate successfully. In some situations, you may need to translate VBA functions and user defined functions into TSQL or. NET functions / procedures. Crosstab queries can be migrated to SQL Server using the PIVOT command.

Microsoft Access has a reputation among IT professionals as not being as economical on server resources when running large query transactions, especially if users force-terminate the application on the client side. Transactions that were running might still be running on the SQL server unbeknownst to the end user.

Protection: Microsoft Access offers several ways to secure the application while allowing users to remain productive.

The most basic is a database password. Once entered, the user has full control of all the database objects. This is a relatively weak form of protection which can be easily cracked.

A higher level of protection is the use of workgroup security requiring a user name and password. Users and groups can be specified along with their rights at the object type or individual object level. This can be used to specify people with readonly or data entry rights but may be challenging to specify. A separate workgroup security file contains the settings which can be used to manage multiple databases. Workgroup security is not supported in the Access 2007 ACCDB database format, although Access 2007 still supports it for MDB databases.

Databases can also be encrypted. The ACCDB format offers significantly advanced encryption from previous versions.

Additionally, if the database design needs to be secured to prevent changes, Access databases can be locked/protected (and the source code compiled) by converting the database to a. MDE file. All changes to the VBA project (modules, forms, or reports) need to be made to the original MDB and then reconverted to MDE. In Access 2007, the ACCDB database is converted to an

ACCDE file. Some tools are available for unlocking and "decompiling", although certain elements including original VBA comments and formatting are normally irretrievable.

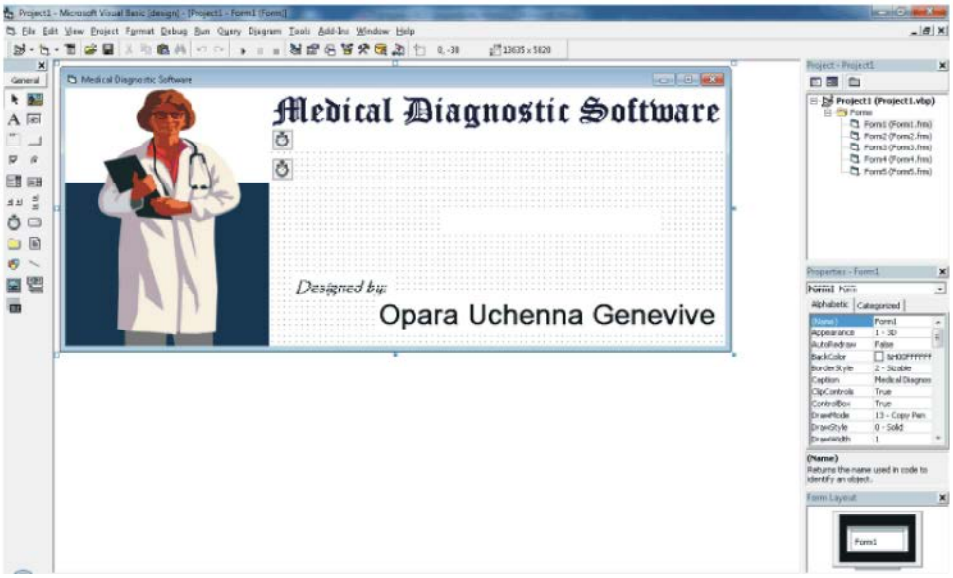

Fig. 5:

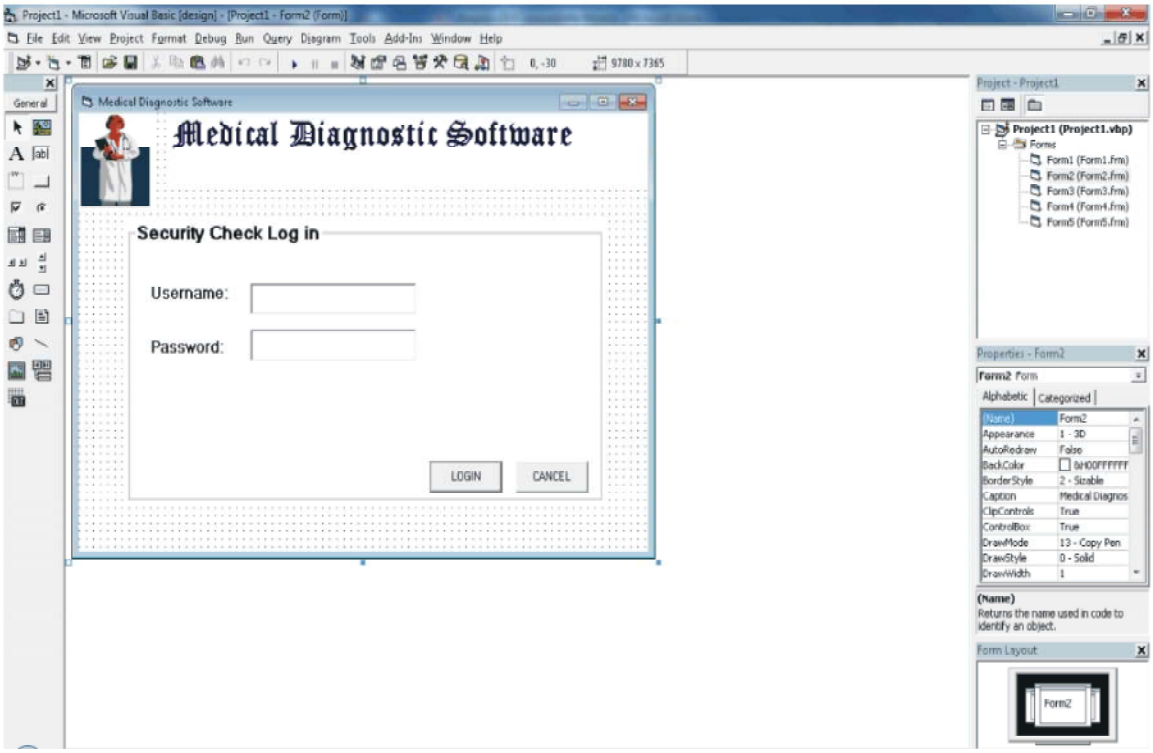

Fig. 6:

Input/output Format Specification

Output Design: The output design of this software was developed as a web page, that is to say that, it represents a webpage that displays information to users and accepts input from them.

Design output - Design outputs are the results of the design and engineering efforts. These are normally the final specifications for the device. Including the manufacturing process and the in-coming, in-process and finished device inspection, measurement or test methods and criteria. The outputs are normally documented in models, drawings, engineering analysis and other documents. The output needs to be directly traceable to the input requirements. Design verification and validation should demonstrate that the final output specifications conform to the input requirements and meet user needs and intended uses.

Input Design: Input facilities the entry of data into the computer system. Input design involves the selection of the best strategy for getting data into the computer system at the right time and as accurately as possible. This is because the most difficult aspect of input design in accuracy. The use of well-defined documents can encourage users to record data accurately without omission.

For example, if a customer's telephone number is a needed input data, the sales order form should have a specific line that is clearly labeled "customer telephone number". Having several lines labeled "customer information" would be less effective. This is because sometimes only the name and address would be filled in leaving out the telephone number.

Input design must capture all the data that the system needs, without introducing any errors. Input errors can be greatly reduced when inputting directly by using appropriate forms for data capture and well designed computer screen layout.

Algorithm Specifications

Program Pseudocode:

Login into the software program
Register Patient
Select type of sickness
Diagnose patient
Display lab result
Prescribe drugs for the patient

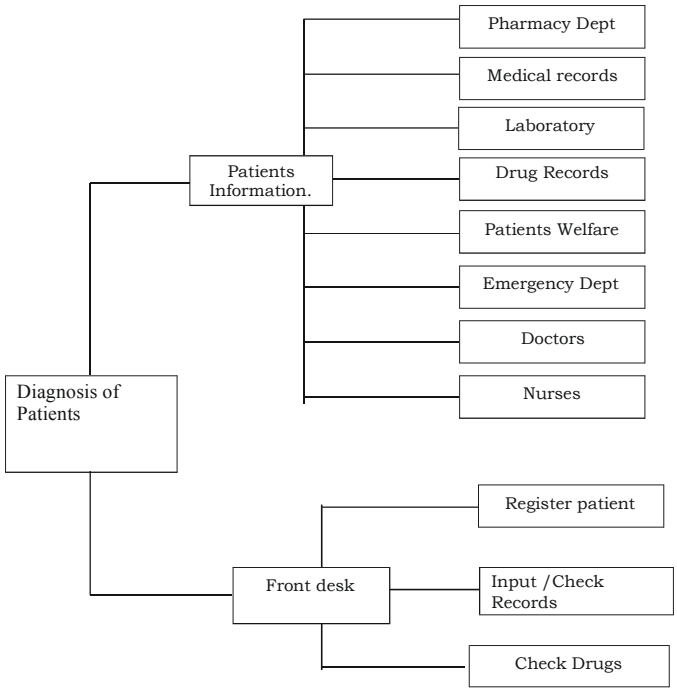

Fig. 7: Over All Dataflow Diagram for the New System

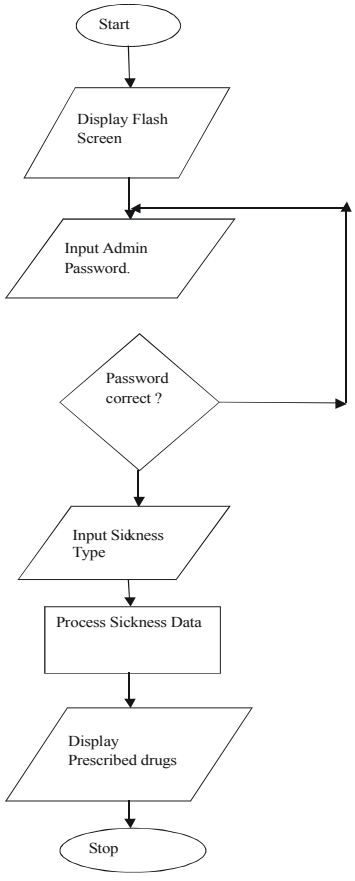

Fig. 8: Program Flow Chart Data Dictionary

A data dictionary, or metadata repository, as defined in the *IBM Dictionary of Computing*, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage and format. The term may have one of several closely related meanings pertaining to databases and database management systems (DBMS):

- A document describing a database or collection of databases
- An integral component of a DBMS that is required to determine its structure a piece of middleware that extends or supplants the native data dictionary of a DBMS

The term Data Dictionary and Data Repository are used to indicate a more general software utility than a catalogue. A Catalogue is closely coupled with the DBMS Software; it provides the information stored in it to user and the DBA, but it is mainly accessed by the various software modules of the DBMS itself, such as DDL and DML compilers, the query optimiser, the transaction processor, report generators and the constraint enforcer. On the other hand, a Data Dictionary is a data structure that stores meta-data, i.e., data about data. The Software package for a stand-alone Data Dictionary or Data Repository may interact with the software modules of the

DBMS, but it is mainly used by the Designers, Users and Administrators of a computer system for information resource management. These systems are used to maintain information on system hardware and software configuration, documentation, application and users as well as other information relevant to system administration.

- If a data dictionary system is used only by the designers, users and administrators and not by the DBMS Software, it is called a Passive Data Dictionary; otherwise, it is called an Active Data Dictionary or Data Dictionary. An *Active Data Dictionary* is automatically updated as changes occur in the database. A *Passive Data Dictionary* must be manually updated.
- The data Dictionary consists of record types (tables) created in the database by systems generated command files, tailored for each supported back-end DBMS. Command files contain SQL Statements for CREATE TABLE, CREATE UNIQUE INDEX, ALTER TABLE (for referential integrity), etc., using the specific statement required by that type of database.
- Database users and application developers can benefit from an authoritative data dictionary document that catalogs the organization, contents and conventions of one or more databases [2]. This typically includes the names and descriptions of various tables and fields in each database, plus additional details, like the type and length of each data element. There is no universal standard as to the level of detail in such a document, but it is primarily a weak kind of data.

Hardware Specification: For the effective operation of the newly designed system, the following minimum hardware specifications are recommended.

- The computer system in used should be IBM compatible since they are considered clone systems.
- The Random access memory (RAM) should be at least 128KB.
- The system should have a hard disk of at least 50GB and at least a diskette drive of high density of 1.44MB (3.5 inches)
- The system should be equipped with an E.G.A/V.G.A, a coloured monitor.
- An uninterruptible power supply (UPS) units
- It should be internet ready.

These listed configurations are the minimum requirements but if the configurations are higher the reports derived will definitely be better and the program will run much faster.

Software Specification: The software specifications required on the computer system are:

- A window 98 or higher version for faster processing
- Web browser
- Text Editor

How to Install the Software: To install this program software, follow the following steps:

- Insert the CD ROM inside the CD ROM drive
- Open the CD ROM Disc
- Double click on the Setup File
- Follow the on screen instruction

How to Use the Software: Double click on the program icon, log in into the software with the password and username. Register the patient and select the type of sickness for diagnose and prescription.

Program 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 successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limiting type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing is an *investigatory* testing phase, where the focus is to have almost a destructive attitude and tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

Types of Tests Done on the Project Software: The following examples are different types of testing that I performed during System testing:

- GUI software testing
- Usability testing
- Performance testing
- Compatibility testing
- Error handling testing
- Reliability Testing.

GUI Software Testing: In computer science, GUI software testing is the process of testing a product that uses a graphical user interface, to ensure it meets its written specifications. This is normally done through the use of a variety of test cases. To generate a 'good' set of test cases, the test designer must be certain that their suite covers all the functionality of the system and also has to be sure that the suite fully exercises the GUI itself. The difficulty in accomplishing this task is twofold: one has to deal with domain size and then one has to deal with sequences. In addition, the tester faces more difficulty when they have to do regression testing.

The size problem can be easily illustrated. Unlike a CLI (command line interface) system, a GUI has many operations that need to be tested. A very small program such as Microsoft WordPad has 325 possible GUI operations[1]. In a large program, the number of operations can easily be an order of magnitude larger.

The second problem is the sequencing problem. Some functionality of the system may only be accomplishable by following some complex sequence of GUI events. For example, to open a file a user may have to click on the File Menu and then select the Open operation and then use a dialog box to specify the file name and then focus the application on the newly opened window.

Obviously, increasing the number of possible operations increases the sequencing problem exponentially. This can become a serious issue when the tester is creating test cases manually.

Usability Testing: This is a technique used to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system.[1] This is in contrast with usability inspection methods where experts use different methods to evaluate a user interface without involving users.

Usability testing focuses on measuring a human-made product's capacity to meet its intended purpose. Examples of products that commonly benefit from usability testing are foods, consumer products, web sites or web applications, computer interfaces, documents and devices. Usability testing measures the usability, or ease of use, of a specific object or set of objects, whereas general human-computer interaction studies attempt to formulate universal principles

Usability testing is a black-box testing technique. The aim is to observe people using the product to discover errors and areas of improvement. Usability testing generally involves measuring how well test subjects respond in four areas: efficiency, accuracy, recall and emotional response. The results of the first test can be treated as a baseline or control measurement; all subsequent tests can then be compared to the baseline to indicate improvement.

- *Performance* -- How much time and how many steps, are required for people to complete basic tasks? (For example, find something to buy, create a new account and order the item.)
- *Accuracy* -- How many mistakes did people make? (And were they fatal or recoverable with the right information?)
- *Recall* -- How much does the person remember afterwards or after periods of non-use?
- *Emotional response* -- How does the person feel about the tasks completed? Is the person confident, stressed? Would the user recommend this system to a friend?

Methods of Usability Test: Setting up a usability test involves carefully creating a scenario, or realistic situation, wherein the person performs a list of tasks using the product being tested while observers watch and take notes. Several other test instruments such as scripted instructions, paper prototypes and pre- and post-test questionnaires are also used to gather feedback on the product being tested. For example, to test the attachment function of an e-mail program, a scenario would describe a situation where a person needs to send an e-mail attachment and ask him or her to undertake this task. The aim is to observe how people function in a realistic manner, so that developers can see problem areas and what people like. Techniques popularly used to gather data during a usability test include think aloud protocol and eye tracking.

Hallway Testing: Hallway testing (or Hall Intercept Testing) is a general methodology of usability testing. Rather than using an in-house, trained group of testers, just five to six random people, indicative of a cross-

section of end users, are brought in to test the product, or service. The name of the technique refers to the fact that the testers should be random people who pass by in the hallway

Remote Testing: Remote usability testing (also known as unmoderated or asynchronous usability testing) involves the use of a specially modified online survey, allowing the quantification of user testing studies by providing the ability to generate large sample sizes. Similar to an in-lab study, a remote usability test is task-based and the platforms allow you to capture clicks and task times. Hence, for many large companies this allows you to understand the WHY behind the visitors intents when visiting a website or mobile site. Additionally, this style of user testing also provides an opportunity to segment feedback by demographic, attitudinal and behavioural type.

The tests are carried out in the user's own environment (rather than labs) helping further simulate real-life scenario testing. This approach also provides a vehicle to easily solicit feedback from users in remote areas.

Performance Testing: This covers a broad range of engineering or functional evaluations where a material, product, system, or person is not specified by detailed material or component specifications: rather, emphasis is on the final measurable performance characteristics.

Performance testing can refer to the assessment of the performance of a human examinee. For example, a behind-the-wheel driving test is a performance test of whether a person is able to perform the functions of a competent driver of an automobile.

In the computer industry, software performance testing is used to determine the speed or effectiveness of a computer, network, software program or device. This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions. Qualitative attributes such as reliability, scalability and interoperability may also be evaluated. Performance testing is often done in conjunction with stress testing.

Compatibility Testing: This is a part of software non-functional tests, is testing conducted on the application to evaluate the application's compatibility with the computing environment. Computing environment may contain some or all of the below mentioned elements:

- Computing capacity of Hardware Platform (IBM 360, HP 9000, etc.)..
- Bandwidth handling capacity of networking hardware
- Compatibility of peripherals (Printer, DVD drive, etc.)
- Operating systems (MVS, UNIX, Windows, etc.)
- Database (Oracle, Sybase, DB2, etc.)
- Other System Software (Web server, networking/ messaging tool, etc.)
- Browser compatibility (Firefox, Netscape, Internet Explorer, Safari, etc.)

Browser compatibility testing, can be more appropriately referred to as user experience testing. This requires that the web applications are tested on different web browsers, to ensure the following:

- Users have the same visual experience irrespective of the browsers through which they view the web application.
- In terms of functionality, the application must behave and respond the same way across different browsers.

Exception Handling: This is a programming language construct or computer hardware mechanism designed to handle the occurrence of exceptions, special conditions that change the normal flow of program execution.

Programming languages differ considerably in their support for exception handling (as distinct from error checking, which is normal program flow that codes for responses to adverse contingencies such as invalid state changes or the unsuccessful termination of invoked operations.) In some programming languages there are functions which cannot be safely called on invalid input data... or functions which return values which cannot be distinguished from exceptions. For example in C, the *atoi* (ASCII to integer conversion) function may return 0 (zero) for any input that cannot be parsed into a valid value. In such languages the programmer must either perform error checking (possibly through some auxiliary global variable such as C's errno) or input validation (perhaps using regular expressions).

The degree to which such explicit validation and error checking is necessary is in contrast to exception handling support provided by any given programming environment. Hardware exception handling differs somewhat from the support provided by software tools, but similar concepts and terminology are prevalent.

In general, an exception is *handled* (resolved) by saving the current state of execution in a predefined place and switching the execution to a specific subroutine known as an *exception handler*. Depending on the

situation, the handler may later resume the execution at the original location using the saved information. For example, a page fault will usually allow the program to be resumed, while a division by zero might not be resolvable transparently.

From the processing point of view, hardware interrupts are similar to resume-able exceptions, though they are typically unrelated to the user's program flow.

From the point of view of the author of a routine, raising an exception is a useful way to signal that a routine could not execute normally. For example, when an input argument is invalid (e.g. a zero denominator in division) or when a resource it relies on is unavailable (like a missing file, or a hard disk error). In systems without exceptions, routines would need to return some special error code. However, this is sometimes complicated by the semipredicate problem, in which users of the routine need to write extra code to distinguish normal return values from erroneous ones.

In runtime engine environments such as Java or. NET, there exist tools that attach to the runtime engine and every time that an exception of interest occurs, they record debugging information that existed in memory at the time the exception was thrown (call stack and heap values). These tools are called automated exception handling or error interception tools and provide 'root-cause' information for exceptions.

Contemporary applications face many design challenges when considering exception handling strategies. Particularly in modern enterprise level applications, exceptions must often cross process boundaries and machine boundaries. Part of designing a solid exception handling strategy is recognizing when a process has failed to the point where it cannot be economically handled by the software portion of the process.

Reliability Testing: This was developed apart from the mainstream of probability and statistics. It was originally a tool to help nineteenth century maritime insurance and life insurance companies compute profitable rates to charge their customers. Even today, the terms "failure rate" and "hazard rate" are often used interchangeably.

The failure of mechanical devices such as ships, trains and cars, is similar in many ways to the life or death of biological organisms. Statistical models appropriate for any of these topics are generically called "time-to-event" models. Death or failure is called an "event" and the goal is to project or forecast the rate of events for a given population or the probability of an event for an individual.

When reliability is considered from the perspective of the consumer of a technology or service, actual reliability measures may differ dramatically from perceived reliability. One bad experience can be magnified in the mind of the customer, inflating the perceived unreliability of the product. One plane crash where hundreds of passengers die will immediately instill fear in a large percentage of the flying consumer population, regardless of actual reliability data about the safety of air travel.

Reliability period of any object is measured within the durability period of that object.

Test Plan: Software testing can be approached in different ways for example one may decide to proceed in a top-down fashion in which case.

- The main system driver is tested first using dommies to replace sub-systems.
- The sub-system driver for each of the sub-system is tested.
- Finally, for each sub-system the software modules are tested one after the other to complete then top down test plan.

An alternative to top-down test plan is the bottom-up test plan. In this:

- The program module comprising each sub-system are tested one after the other
- The sub-system driver for each of the sub-system is tested to ensure that control is transferred to the appropriate program modules.
- Finally the main system driver is tested.

Because the lower levels are available before any top level is tested. There are other test approaches but the two mentioned above lie within the scope of this project.

Software Sub-System Testing: In the software sub-system testing, i used the bottom-up test plan, which includes the types of test I listed above on the system testing module.

I generated adequate test data for each level of testing namely:

- System Driver
- Sub-systems
- Program modules

Change over Procedure: These are methods to change from an existing system to a new one. I used the direct change over procedure.

Direct Change over: This is the simplest methods of changeover. Imagine a new checkout system for a supermarket. If the implementation is done by direct changeover, then the supermarket will be closed for one night, all the old checkouts will be taken out, the new ones will be installed and the supermarket will be ready to open the next day. Although this is a simple method, it takes a lot of planning. All the files have to be ready to load, all the workers need to have been trained in advance and the system must have been properly tested. If the checkouts do not work when they are switched on, the store will have to close because they cannot go back to the old system.

This is how the UK stock market was computerized in the 1980s. The old market closed on Friday night and the computers were all switched on the following Monday morning. Unfortunately, there was an error in the software, which made all the computers sell shares and carry on selling them. By the time human beings had intervened to stop the computers, half of the value of companies had been wiped off the stock market and many people had been ruined.

Parallel Change over: During changeover, a new system and an existing system run side by side. To input the same data and perform the same processes, compare their output and prove the reliability of the new system. If the new system is accepted, the existing system will stop running and will be replaced by the new one [3].

The practical example of parallel running in human resource management is job placement. A new staff and an old staff work for the same job. If the new staff's performance is OK, the existing staff may not be needed any more and will be replaced. This is different from direct changeover and phased implementation because parallel running requires two systems working at the same time.

Pilot Change over: Pilot change over implementation is a method of changing from an existing system to a new one.

Phased implementation is a changeover process that takes place in stages.

As an example, think of a supermarket. In this supermarket the checkout system is being upgraded to a newer version. Imagine that only the checkout counters of the vegetable section are changed over to the new system, while the other counters carry on with the old system. If the new system does not work properly, it won't matter because only a small part of the supermarket has been computerised. If it does work, staff can take turns working on the vegetable counters to get some practice using the new system. After the vegetables section is working perfectly, the meat section might be next, then the confectionery section and so on. Eventually all the various counters in the supermarket system would have been phased in and everything would be running. This takes a long time as there are two systems working until the changeover is completed. However, the supermarket is never in danger of having to shut and the staff are all able to get plenty of training in operating the new system, so it is a much friendlier method.

This Project has specifically improved access to medical care, reducing the cost of visiting the doctor for reatment. Due to the simulation nature of this project, it has reduced the risk, over head cost and logistics required in one to one interaction with a doctor. This software can be used by individuals, Government and organizations that wants to have free, easy and quality access to health care and these are the achievements of this project after the classroom simulation.

To Interact with patients,

- Diagnose a patient
- Prescribes drugs for patients.
- Keep track of patients health information and data
- Collect data which doctors could use to treat and remediate patients sickness.

Problems Encountered and Solutions: The problems I encountered while undertaking this project includes:

- Limited power supply to finish the software coding and typing of report.
- Scarcity of original Microsoft Visual Basic Cd-rom.
- Limited Funds

Recommendations/ Suggestions for Further Improvement: Due to the importance and significance of this project, I recommend the following for future researchers:

- The program software should be design to run on the internet for wider use
- Multimedia (videos and sounds) should be included in the training materials for more interesting learning experience.
- The interactive aspects of this software should be increased.
- For more over all completion, hardware should be interfaced to this project.

CONCLUSION

Human doctors provide a highly efficient health solutions to patients and have been estimated to reduce death rate, achievement outcomes by as much as two standard deviations. The goal of medical Expert system would be to engage the patients in sustained reasoning activity and to interact with the patient based on health symptoms and problems. If such systems realize even half the impact of human doctors, the payoff for society promised to be substantial.

REFERENCES

1. Anderson, J.R., 1983. The Architecture of Cognition. Cambridge, Massachusetts: Harvard University Press, pp: 4.
2. Mayer, R.E., 1988. Teaching and Learning Computer Programming: Multiple Research Perspectives. Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp: 7-9.
3. Olson, G.M., S. Sheppard and E. Soloway, 1987. Empirical Studies of Programmers: Second Workshop. Norwood, New Jersey: Ablex Publishing Corporation, pp: 9-13.
4. Anderson, J.R. and G.H. Bower, 1973. Human associative memory. Washington, DC: V.H Winston and Sons., pp: 12.
5. Anderson, J.R., 1983. The architecture of cognition. Cambridge, MA: Harvard University Press, pp: 3-4.
6. Bloom, B.S., 1984. The 2 sigma problem: The search for methods of group instruction as effective as one-to- one tutoring. Educational Researcher, 13: 4-16.
7. Carbonell, 1970. Al in CA!: An artificial intelligence approach to computer-assisted instruction. Washington, DC: V.H Winston and Sons, pp: 12.
8. IEEE Transactions on Man-Machine Systems, 11,190-202.
9. Cohen, P.A., J.A. Kulik and C.C. Kulik, 1982. Educational outcomes of tutoring: A meta analysis of findings. American Educational Research Journal, 19: 237-248.
10. Collins, A.M. and E.F. Loftus, 1975. A spreading-activation theory of semantic processing. Psychology cal Review, 82: 407-428.
11. Frasson, C., G. Gauthier and A. Lesgold, 1996. Intelligent tutoring systems: Third International Conference, ITS'96. New York: Springer-Verlag.
12. Lesgold, A., S. Lajoie, M. Bunzo and G. Eggan, 1992. SHERLOCK: A coached practice environment for electronics troubleshooting job. In J. Larkin, pp: 6.
13. Weinshank, D.J., M. Urban-Lurain and T. Danieli, 1988. Introduction to Computing: Telecourse with Structured BASIC. (2nd ed.). Dubuque, Iowa: Kendall/Hunt Publishing Company, pp: 5.
14. Weinshank, D.J., M. Urban-Lurain, T. Danieli and G. McCuaig, 1995. Integrated Introduction to Computing. (updated first edition, revised and enlarged ed.). Dubuque, Iowa: Kendall/Hunt Publishing Company, pp: 2-5.
15. Larkin and Chabay, 1992. Computer Assisted Learning Program guidelines, Washington, DC: V.H Winston and Sons., pp: 12.
16. Steven and Collins, 1977. Artificial learning a practical guide, Washington, DC: V.H Winston and Sons, pp: 1-5.
17. Collins and Steven, 1991. Natural Learning System, Washington, DC: V.H Winston and Sons, pp: 9-12.
18. Wolf and Mc Donald, 1984. Research Program Tutoring system, V.H Winston and Sons, pp: 3-7.
19. Mayer, 1988. Computer aided learning schemes, Washington, DC: V.H Winston and Sons, pp: 8.
20. Olson, Sheppard and Soloway, 1987. The Computer teaches, California Press: V.H Winston and Sons, pp: 8-12.
21. Anderson and Corbett, 1993. Artificial learning systems, using Robots, V.H Winston and Sons, pp: 12.
22. Levitt, 2000. Combinational Digital Systems, New York Times Press, NY, pp: 2-6.
23. Douglas, T. and Soft Tech, 2001. Artificial learning Technologies, New York Times Press, NY, pp: 8-16.
24. Harlan, Mills and Nikklaus, 1970. Parallel Structured Programming Concepts, New York Times Press, NY, pp: 2-6.
25. Boolos and Jeffrey, 1974, 1999. Introduction to final reading, New York Times Press, NY, pp: 2-6.