**1. INTRODUCTION**

**1.1 ABSTRACT**

 This project is entitles as **“Bio informatics lab data analysis system”** is developed using ASP.NET as front end C# as coding language and SQL Server as back end. Data details will be furnished using Data grid and MS charts. Data validation will be done through java script.

 The main objective of this project is to develop a Client Server based environment, for transferring the medical report details from the bio medical lab. Also to improve data analysis technique, using previous data comparison methods.

 This system works on accurate recording of all user report transactions lead to better data management and increases in various comparisons. This project consists of two types of interactions like admin and patient. Admin will be providing with a well secured login and patient details will be registered using admin. A user name and password will be provided to the user. The initially process is to enter the medical test requirement of the patient. The details contain prescribed doctor details, patient details and lab test details. Each disease will contain different CTA (Complete test analysis) for their test result. For example a general blood sugar test contains HBL cholesterol, LDL cholesterol, Glucose level and etc. These all said to be CTA test.

 Once patient has been register successfully their blood will be collected and a temporary reference number will be provided. This temporary number is for admin and employee reference. Once their blood test has been done, the test result will be uploaded by the admin by selected the corresponding user name. The uploading details will be stored in a centralized server for data utility purpose.

 Patient can login with their user and password. Through selecting the date, the patient can view their current test result from their home itself. A test history option will be provided to the patient. So that patient can compare their test with their previous test results. This option makes the patient to manage and know about their health condition and their treatment. An interactive grid and chart will be generated for furnishing the result to the patient. Admin can maintain their patient detail, patient count, test results and etc. This makes more comfortable communication between admin and patients. Also entire treatment record will be computerized for future references.

**1.2 MODULES**

1. Admin login and patient creation
2. Upload Test Reports
3. User View
4. Upload contents
5. Chart and grid

**1.3 MODULES DESCRIPTION:**

**Admin Login**

 The initial module of this project is admin module. Here admin will be provided with a user name and password. There is also possibility for password changing option inside the admin login. The admin will be fully authorized person for this entire project. In added with he will be fully operational authority of this project. The admin can able to access all the option in this project and he can able to do all types of updating. And also admin can able to create patient and provide their username and password.

**Upload Test Reports**

 This module is under the control of admin. A temporary reference number will be generated for patients. After the completion of lab test, the test report details will be uploaded in the patient zone. The uploading data will be centralized in server. The uploading details will be categorized into date wise, doctor wise, Patient wise, disease wise and etc. Each and every test contains sub test details. Full payment will be collected from the user side while patient giving their blood to the lab advisor.

**User View**

Using the user name and password, the patient can login anywhere at any time. All the uploaded contents can be viewed in the user’s login. A change password option will be provided to the user to make their treatment history more secured.

**Upload Content**

 A premium option will be provide to the patient in this module. In case the patient has been taking treatment in other place means, after their treatment, patient can upload their new lab report with the existing treatment report. Data merging technique has been implemented for merging the existing data with the current data. Not even other lab details, they can upload various medical records, which can be used in their future.

**Chart and Grid**

This is the data display module, various grid and charts can be generated from the user side. This gives the data more clarity to access. Patient can select from date and to date to view their treatment history. All data will be compared in charts and graphs for clarity results. Using this option patient can monitor their health. And they will get more confidence on their treatment.

**2. SYSTEM SPECFICATION**

**2.1 HARDWARE SPECIFICATION**

PROCESSOR : Intel Pentium Dual Core 1.8 GHz

MOTHERBOARD : Intel 915GVSR chipset board

RAM : 4 GB DDR3 RAM

HARD DISK DRIVE : 750 GB

DVD/CD DRIVE : Sony 52 x Dual layer drive

MONITOR : 17” Color TFT Monitor

KEYBOARD : Multimedia Keyboard 108 Keys

MOUSE : Logitech Optical Mouse

CABINET : ATX iball.

**2.2 SOFTWARE CONFIGURATION**

FRONTEND : ASP.NET 2012

CODING LANDUAGE : C#

BACK END : SQL SERVER 2010

CLIENT SERVER TOOL : AJAX 2.0

OPERATING SYSTEMS : Microsoft Windows 7

DOCUMENTATION : Microsoft word 2007.

SCRIPTING LANGUAGE : Java Script

**3. SYSTEM STUDY**

**3.1 EXISTING SYSTEM**

 In the system both bio lab admin and patients are facing more problems due to manual works. In case the blood sample form the patient is received means, they need to wait for the medical result or they need to come again to receive their medical reports. This makes the patient more inconvenient to travel and wait for the result. In some times the patient may miss their report, at that time they need to visit again to the lab for getting their report again. Also some times the patient may miss their previous report , which may need to track their medical history. In case of verifying the patient history with the manual paper reports means, they need to compare their reports manually. Admin also facing more problems in the existing system like updating the test details and managing the customer details. Still doing more paper works with problems

 Some of the bio labs are computerized like maintaining their customer details and proving the reports in a computer printout. But they still typing their result in a word file and proving printouts to the patients. They are not maintaining proper patient details with their previous medical reports. This is the most important problems facing by the bio lab now a day.

**3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM**

* Only Manual work
* No computerized patient report
* No customized login for patients to receive their medical report
* No comparison for the records with the previous records
* Patients may miss their reports
* Admin facing more problem in adding new disease to the database
* Problem in maintain customers

**3.2 PROPOSED SYSTEM**

 All the drawbacks in the existing system have been over came in the proposed system. The important of the proposed system in computerization of the manual work to an automated process. In the proposed system the patient need not visit the lab frequently. Once the blood sample has been collected from the patient, the lab will start their process. The lab admin will be directly entering the reports details into the patient corresponding username. This data will be transfer to the patient’s login directly. Also all the patients will be provided with a user name and a password. Patient can directly login in the link and they can view their test report immediately.

 For the admin part, no paper works or manual print outs will be done. All the details will be computerized in this application. Lab can provide both software copy and hardcopy to the patients. What even data has been provided will be computerized in the system. Even after some year a particular patient’s details can be fetched out.

**3.2.1 ADVANTAGES OF THE PROPOSED SYSTEM**

* Fully automated work
* All patient details and report details will be computer
* Individual login will be provided to patients with user name and password
* Reports can be compared by the patient in their login itself. Patient will get aware about their treatment while comparing their records.
* No possible way to miss their records, everything will be computerized.
* In case of adding new disease details, admin can add or remove the disease details with admin login
* Any customers can be tracked easily and can easily maintain their details also

**4. SYSTEM DESIGN**

**4.1 DATA FLOW DIAGRAM**

**Level 0:**

Request for username & password

Database

Admin / customer

Response

**Level 1:**

 Username , password

adlogintbl

Testname , range

newtesttbl

Edit , Delete

Name , age , username,password

updateresulttbl

customertbl

Admin / user

User id , name , labtest

Name , view test

Old password , new password

**Level 2:**

Username , password

customertbl

Id , data , testname , result

id , name

Customer

updateresulttbl

Old password, new password

**4.2 ER DIAGRAM**

**4.3 TABLE DESIGN**

**Table Name :** adlogintbl

**Primary key :** username

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| Username | Varchar | 50 | Primary key | Admin username |
| Password | Varchar | 20 | Not Null | Admin password |

**Table Name :** customertbl

**Primary key :** username

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| name | Varchar | 50 | Primary key | Customer name |
| gender | Varchar | 20 | Not Null | Customer gender |
| age | Int | 10 | Not Null | Customer age |
| bloodgroup | Varchar | 50 | Not Null | Customer blood group |
| street1 | Varchar | 100 | Not Null | Customer street1 |
| street2 | Varchar | 100 | Not Null | Customer street2 |
| city | Varchar | 50 | Not Null | Customer city |
| state | Varchar | 50 | Not Null | Customer state |
| pin | Int | 10 | Not Null | Customer pinno |
| phone | Int | 10 | Not Null | Customer phoneno |
| email | Varchar | 50 | Not Null | Customer email |
| username | Varchar | 50 | Not Null | Customer username |
| password | Varchar | 20 | Not Null | Customer password |

**Table Name :** newtesttbl

**Primary key :** testname

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description**  |
| testname | varchar | 50 | Primary key | Create testname |
| subtestname | varchar | 50 | Not Null | Create subtestname |
| frmrange | int | 10 | Not Null | From range |
| fmeasurement | varchar | 50 | Not Null | From measurement |
| torange | int | 10 | Not Null | To range |
| tmeasurement | varchar | 50 | Not Null | To measurement |

**Table Name :** updateresulttbl

**Primary key :** id

**Foreign key :** username

**Foreign key :** username

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| Id | int | 10 | Primary key | Identification number |
| username | varchar | 50 | Foreign key | Customer username |
| name | varchar | 50 | Not Null | Customer name |
| testname | varchar | 50 | Not Null | Customer testname |
| subtestname | int | 50 | Not Null | Customer subtestname |
| labtest | varchar | 50 | Not Null | Customer labtest |
| labtestvalue | int | 10 | Not Null | Customer labtestvalue |
| frmrange | int | 10 | Not Null | From range |
| fmeasurement | varchar | 50 | Not Null | From measurement |
| torange | int | 10 | Not Null | To range |
| tmeasurement | varchar | 50 | Not Null | To measurement |
| result | varchar | 50 | Not Null | Test result |
| update | Datetime | - | Not Null | Update date |
| uptime | Datetime | - | Not Null | Update time |

**5. ABOUT SOFTWARE**

**5.1 ABOUT FRONT END**

Asp.net is a server-side web application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services. It was first released in January 2002 with version 1.0 of the.net framework, and is the successor to Microsoft’s active server pages (asp) technology. Asp.net is built on the common language runtime (clr), allowing programmers to write asp.net code using any supported .net language. The asp.net soap extension framework allows asp.net components to process soap messages.

After four years of development, and a series of beta releases in 2000 and 2001, asp.net 1.0 was released on January 5, 2002 as part of version 1.0 of the .net framework. Even prior to the release, dozens of books had been written about asp.net, and Microsoft promoted it heavily as part of its platform for web services. Scott Guthrie became the product unit manager for asp.net, and development continued apace, with version 1.1 being released on April 24, 2003 as a part of windows server 2003. This release focused on improving asp. Net’s support for mobile devices.

**Characteristics**

Asp.net web pages, known officially as web forms, are the main building blocks for application development. web forms are contained in files with a ".aspx" extension; these files typically contain static (x)html markup, as well as markup defining server-side web controls and user controls where the developers place all the rc content[further explanation needed] for the web page. Additionally, dynamic code which runs on the server can be placed in a page within a block <% -- dynamic code -- %>, which is similar to other web development technologies such as php, jsp, and asp. With asp.net framework 2.0, Microsoft introduced a new code-behind model which allows static text to remain on the .aspx page, while dynamic code remains in an .aspx.vb or .aspx.cs or .aspx.fs file (depending on the programming language used).

**Directives**

A directive is a special instruction on how asp.net should process the page.the most common directive is <%@ page %> which can specify many attributes used by the asp.net page parser and compiler.

**Examples**

Inline code[edit source | editbeta]

<%@ page language="c#" %>

<!Doctype html public "---//w3c//dtd xhtml 1.0 //en"

"http://www.w3.org/tr/xhtml1/dtd/xhtml1-transitional.dtd">

<script runat="server">

 protected void page\_load(object sender, eventargs e)

 { // assign the datetime to label control

 lbl1.text = datetime.now.tolongtimestring();

 }

</script>

 <html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

 <title>sample page</title>

</head>

<body>

 <form id="form1" runat="server">

Code-behind solutions[edit source | editbeta]

<%@ page language="c#" codefile="samplecodebehind.aspx.cs" inherits="website.samplecodebehind"

Autoeventwireup="true" %>

The above tag is placed at the beginning of the aspx file. The code file property of the @ page directive specifies the file (.cs or .vb or .fs) acting as the code-behind while the inherits property specifies the class from which the page is derived. In this example, the @ page directive is included in samplecodebehind.aspx, then (samplecodebehind.aspx.cs) acts as the code-behind for this page:

**Source language c#:**

Using system;

Namespace website

{

 public partial class samplecodebehind : system.web.ui.page

 {

 protected void page\_load(object sender, eventargs e)

 {

 response.write("hello, world");

 }

 }

}

Source language visual basic.net:

Imports system

Namespace website

 public partial class samplecodebehind

 inherits system.web.ui.page

 protected sub page\_load(byval sender as object, byval e as eventargs)

 response.write("hello, world")

 end sub

 end class

**End namespace**

In this case, the page\_load() method is called every time the aspx page is requested. The programmer can implement event handlers at several stages of the page execution process to perform processing.

**User controls**

User controls are encapsulations of sections of pages which are registered and used as controls in asp.net, etc.

**Custom controls**

Programmers can also build custom controls for asp.net applications. Unlike user controls, these controls do not have an ascx markup file, having all their code compiled into a dynamic link library (dll) file. Such custom controls can be used across multiple web applications and visual studio projects.

**Rendering technique**

Asp.net uses a visited composites rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template. Literal text goes into instances of the literal control class, and server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page. The page doubles as the root of the control tree.

Actual requests for the page are processed through a number of steps. First, during the initialization steps, an instance of the page class is created and the initialization code is executed. This produces the initial control tree which is now typically manipulated by the methods of the page in the following steps. As each node in the tree is a control represented as an instance of a class, the code may change the tree structure as well as manipulate the properties/methods of the individual nodes. Finally, during the rendering step a visitor is used to visit every node in the tree, asking each node to render itself using the methods of the visitor. The resulting html output is sent to the client.

After the request has been processed, the instance of the page class is discarded and with it the entire control tree. This is a source of confusion among novice asp.net programmers who rely on the class instance members that are lost with every page request/response cycle.

**State management**

Asp.net applications are hosted by a web server and are accessed using the stateless http protocol. As such, if an application uses state full interaction, it has to implement state management on its own. Asp.net provides various functions for state management. Conceptually, Microsoft treats "state" as gui state. Problems may arise if an application needs to keep track of "data state"; for example, a finite-state machines which may be in a transient state between requests (lazy evaluation) or which takes a long time to initialize. State management in asp.net pages with authentication can make web scraping difficult or impossible.

**Application**

Application state is held by a collection of shared user-defined variables. These are set and initialized when the application\_onstart event fires on the loading of the first instance of the application and are available until the last instance exits. Application state variables are accessed using the applications collection, which provides a wrapper for the application state. Application state variables are identified by name.

**Session state**

Server-side session state is held by a collection of user-defined session variables that are persistent during a user session. These variables, accessed using the session collection, are unique to each session instance. The variables can be set to be automatically destroyed after a defined time of inactivity even if the session does not end. Client-side user session is maintained by either a cookie or by encoding the session id in the url itself.

Asp.net supports three modes of persistence for server-side session variables:

**In-process mode**

The session variables are maintained within the asp.net process. This is the fastest way; however, in this mode the variables are destroyed when the asp.net process is recycled or shut down.

**Asp state mode**

Asp.net runs a separate windows service that maintains the state variables. Because state management happens outside, the asp.net process, and because the asp.net engine accesses data using .net removing, asp state is slower than in-process. This mode allows an asp.net application to be load-balanced and scaled across multiple servers. Because the state management service runs independently of asp.net, the session variables can persist across asp.net process shutdowns. However, since session state server runs as one instance, it is still one point of failure for session state. The session-state service cannot be load-balanced, and there are restrictions on types that can be stored in a session variable.

**5.2 ABOUT BACK END**

 SQL Server is Microsoft's relational database management system (RDBMS). It is a full-featured database primarily designed to compete against competitors Oracle Database (DB) and MySQL.

Like all major RBDMS, SQL Server supports ANSI SQL, the standard SQL language. However, SQL Server also contains T-SQL, its own SQL implemention.SQL Server Management Studio (SSMS) (previously known as Enterprise Manager) is SQL Server’s main interface tool, and it supports 32-bit and 64-bit environments.

**SQL Server is sometimes referred to as MSSQL and Microsoft SQL Server.**

Originally released in 1989 as version 1.0 by Microsoft, in conjunction with Sybase, SQL Server and its early versions were very similar to Sybase. However, the Microsoft-Sybase partnership dissolved in the early 1990s, and Microsoft retained the rights to the SQL Server trade name. Since then, Microsoft has released 2000, 2005 and 2008 versions, which feature more advanced options and better security.

Examples of some features include: XML data type support, dynamic management views (DMVs), full-text search capability and database mirroring.SQL Server is offered in several editions with different feature set and pricing options to meet a variety of user needs, including the following:

Enterprise: Designed for large enterprises with complex data requirements, data warehousing and Web-enabled databases. Has all the features of SQL Server, and its license pricing is the most expensive.

Standard: Targeted toward small and medium organizations. Also supports e-commerce and data warehousing.

Workgroup: For small organizations. No size or user limits and may be used as the backend database for small Web servers or branch offices.

Express: Free for distribution. Has the fewest number of features and limits database size and users. May be used as a replacement for an Access database.

**Mainstream editions**

**Datacenter**

SQL Server 2008 R2 Datacenter is the full-featured edition of SQL Server and is designed for datacenters that need the high levels of application support and scalability. It supports 256 logical processors and virtually unlimited memory. Comes with StreamInsight Premium edition. The Datacenter edition has been retired in SQL Server 2012, all its features are available in SQL Server 2012 Enterprise Edition.

**Enterprise**

SQL Server Enterprise Edition includes both the core database engine and add-on services, with a range of tools for creating and managing a SQL Server cluster. It can manage databases as large as 524 megabytes and address 2 terabytes of memory and supports 8 physical processors. SQL 2012 Enterprise Edition supports 160 Physical Processors

**Standard**

SQL Server Standard edition includes the core database engine, along with the stand-alone services. It differs from Enterprise edition in that it supports fewer active instances (number of nodes in a cluster) and does not include some high-availability functions such as hot-add memory (allowing memory to be added while the server is still running), and parallel indexes.

**Web**

SQL Server Web Edition is a low-TCO option for Web hosting.

**Business Intelligence**

Introduced in SQL Server 2012 and focusing on Self Service and Corporate Business Intelligence. It includes the Standard Edition capabilities and Business Intelligence tools: PowerPivot, Power View, the BI Semantic Model, Master Data Services, Data Quality Services and xVelocity in-memory analytics.

**Workgroup**

SQL Server Workgroup Edition includes the core database functionality but does not include the additional services. Note that this edition has been retired in SQL Server 2012.

**Express**

SQL Server Express Edition is a scaled down, free edition of SQL Server, which includes the core database engine. While there are no limitations on the number of databases or users supported, it is limited to using one processor, 1 GB memory and 4 GB database files (10 GB database files from SQL Server Express 2008 R2). It is intended as a replacement for MSDE. Two additional editions provide a superset of features not in the original Express Edition. The first is SQL Server Express with Tools, which includes SQL Server Management Studio Basic. SQL Server Express with Advanced Services adds full-text search capability and reporting services.

**Specialized editions**

Azure

Microsoft SQL Azure Database is the cloud-based version of Microsoft SQL Server, presented as software as a service on Azure Services Platform.

Compact (SQL CE)

The compact edition is an embedded database engine. Unlike the other editions of SQL Server, the SQL CE engine is based on SQL Mobile (initially designed for use with hand-held devices) and does not share the same binaries. Due to its small size (1 MB DLL footprint), it has a markedly reduced feature set compared to the other editions. For example, it supports a subset of the standard data types, does not support stored procedures or Views or multiple-statement batches (among other limitations). It is limited to 4 GB maximum database size and cannot be run as a Windows service, Compact Edition must be hosted by the application using it. The 3.5 version includes support for ADO.NET Synchronization Services. SQL CE does not support ODBC connectivity, unlike SQL Server proper.

Developer

SQL Server Developer Edition includes the same features as SQL Server 2012 Enterprise Edition, but is limited by the license to be only used as a development and test system, and not as production server. This edition is available to download by students free of charge as a part of Microsoft's DreamSpark program

Evaluation

SQL Server Evaluation Edition, also known as the Trial Edition, has all the features of the Enterprise Edition, but is limited to 180 days, after which the tools will continue to run, but the server services will stop.

Fast Track

SQL Server Fast Track is specifically for enterprise-scale data warehousing storage and business intelligence processing, and runs on reference-architecture hardware that is optimized for Fast Track.

LocalDB

Introduced in SQL Server Express 2012, LocalDB is a minimal, on-demand, version of SQL Server that is designed for application developers. It can also be used as an embedded database.

Parallel Data Warehouse (PDW)

**Data warehouse Appliance Edition**

Pre-installed and configured as part of an appliance in partnership with Dell & HP base on the Fast Track architecture. This edition does not include SQL Server Integration Services, Analysis Services, or Reporting Services.

**Architecture**

The protocol layer implements the external interface to SQL Server. All operations that can be invoked on SQL Server are communicated to it via a Microsoft-defined format, called Tabular Data Stream (TDS). TDS is an application layer protocol, used to transfer data between a database server and a client. Initially designed and developed by Sybase Inc. for their Sybase SQL Server relational database engine in 1984, and later by Microsoft in Microsoft SQL Server, TDS packets can be encased in other physical transport dependent protocols, including TCP/IP, Named pipes, and Shared memory. Consequently, access to SQL Server is available over these protocols. In addition, the SQL Server API is also exposed over web services.

**Data storage**

 Data storage is a database, which is a collection of tables with typed columns. SQL Server supports different data types, including primary types such as Integer, Float, Decimal, Char (including character strings), Varchar (variable length character strings), binary (for unstructured blobs of data), Text (for textual data) among others. The rounding of floats to integers uses either Symmetric Arithmetic Rounding or Symmetric Round Down (Fix) depending on arguments: SELECT Round(2.5, 0) gives 3.

Microsoft SQL Server also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log. A SQL Server database can contain a maximum of 231 objects, and can span multiple OS-level files with a maximum file size of 260 bytes. The data in the database are stored in primary data files with an extension .mdf. Secondary data files, identified with a .ndf extension, are used to store optional metadata. Log files are identified with the .ldf extension.

Storage space allocated to a database is divided into sequentially numbered pages, each 8 KB in size. A page is the basic unit of I/O for SQL Server operations. A page is marked with a 96-byte header which stores metadata about the page including the page number, page type, free space on the page and the ID of the object that owns it. Page type defines the data contained in the page - data stored in the database, index, allocation map which holds information about how pages are allocated to tables and indexes, change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While page is the basic unit of an I/O operation, space is actually managed in terms of anextent which consists of 8 pages. A database object can either span all 8 pages in an extent ("uniform extent") or share an extent with up to 7 more objects ("mixed extent"). A row in a database table cannot span more than one page, so is limited to 8 KB in size. However, if the data exceeds 8 KB and the row contains Varchar or Varbinary data, the data in those columns are moved to a new page (or possibly a sequence of pages, called an Allocation unit) and replaced with a pointer to the data.

For physical storage of a table, its rows are divided into a series of partitions (numbered 1 to n). The partition size is user defined; by default all rows are in a single partition. A table is split into multiple partitions in order to spread a database over a cluster. Rows in each partition are stored in either B-tree or heap structure. If the table has an associated index to allow fast retrieval of rows, the rows are stored in-order according to their index values, with a B-tree providing the index. The data is in the leaf node of the leaves, and other nodes storing the index values for the leaf data reachable from the respective nodes. If the index is non-clustered, the rows are not sorted according to the index keys. An indexed view has the same storage structure as an indexed table. A table without an index is stored in an unordered heap structure. Both heaps and B-trees can span multiple allocation units.

**Buffer management**

SQL Server buffers pages in RAM to minimize disc I/O. Any 8 KB page can be buffered in-memory, and the set of all pages currently buffered is called the buffer cache. The amount of memory available to SQL Server decides how many pages will be cached in memory. The buffer cache is managed by the Buffer Manager. Either reading from or writing to any page copies it to the buffer cache. Subsequent reads or writes are redirected to the in-memory copy, rather than the on-disc version. The page is updated on the disc by the Buffer Manager only if the in-memory cache has not been referenced for some time. While writing pages back to disc, asynchronous I/O is used whereby the I/O operation is done in a background thread so that other operations do not have to wait for the I/O operation to complete. Each page is written along with its checksum when it is written. When reading the page back, its checksum is computed again and matched with the stored version to ensure the page has not been damaged or tampered with in the meantime.

**6. SYSTEM TESTING**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding [software bugs](http://en.wikipedia.org/wiki/Software_bug) (errors or other defects).

Development Testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Development Testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process. Software testing can be stated as the process of validating and verifying that a computer program/application/product:

* meets the requirements that guided its design and development,
* works as expected,
* can be implemented with the same characteristics,
* and satisfies the needs of stakeholders.
* White-box testing,
* Black-box testing,
* Unit testing,
* Integration testing,
* System testing,
* Acceptance testing,
* Validation testing

**WHITE-BOX TESTING**

**White-box 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).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

**BLACK-BOX TESTING**

Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides through test coverage. Incorrect and missing functions, interface errors, errors in data structures, error in functional logic are the errors falling in this category.

**UNIT TESTING**

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming a unit could be an entire module but is more commonly an individual function or procedure. In object-oriented programming a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process.

Ideally, each test case is independent from the others: substitutes like method stubs, mock objects, fakes and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.

**INTEGRATION TESTING:**

 Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment.

 For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

**SYSTEM TESTING:**

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(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 limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

**ACCEPTANCE TESTING:**

Acceptance test cards are ideally created during sprint planning or iteration planning meeting, before development begins so that the developers have a clear idea of what to develop. Sometimes acceptance tests may span multiple stories (that are not implemented in the same sprint) and there are different ways to test them out during actual sprints. One popular technique is to mock external interfaces or data to mimic other stories which might not be played out during iteration (as those stories may have been relatively lower business priority). A user story is not considered complete until the acceptance tests have passed.

**VALIDATION TESTING:**

Verification is intended to check that a product, service, or system (or portion thereof, or set thereof) meets a set of initial design specifications. In the development phase, verification procedures involve performing special tests to model or simulate a portion, or the entirety, of a product, service or system, then performing a review or analysis of the modeling results. In the post-development phase, verification procedures involve regularly repeating tests devised specifically to ensure that the product, service, or system continues to meet the initial design requirements, specifications, and regulations as time progresses. It is a process that is used to evaluate whether a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process.

**6.2 Test case**

A [test case](http://en.wikipedia.org/wiki/Test_case) normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output, expected result, and actual result. Clinically defined a test case is an input and an expected result. This can be as pragmatic as 'for condition x your derived result is y', whereas other test cases described in more detail the input scenario and what results might be expected. It can occasionally be a series of steps (but often steps are contained in a separate test procedure that can be exercised against multiple test cases, as a matter of economy) but with one expected result or expected outcome.

 The optional fields are a test case ID, test step, or order of execution number, related requirement(s), depth, test category, author, and check boxes for whether the test is automatable and has been automated. Larger test cases may also contain prerequisite states or steps, and descriptions. A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database, or other common repository. In a database system, you may also be able to see past test results, who generated the results, and what system configuration was used to generate those results. These past results would usually be stored in a separate table.

**7. IMPLEMENTATION**

 System implementation generally benefits from high levels of user involvement and management support. User participation in the design and operation of information systems has several positive results. First, if users are heavily involved in systems design, they move opportunities to mold the system according to their priorities and business requirements, and more opportunities to control the outcome. Second, they are more likely to react positively to the change process. Incorporating user knowledge and expertise leads to better solutions.

The relationship between users and information systems specialists has traditionally been a problem area for information systems implementation efforts. Users and information systems specialists tend to have different backgrounds, interests, and priorities. This is referred to as the user-designer communications gap. These differences lead to divergent organizational loyalties, approaches to problem solving, and vocabularies

Conversions to new systems often get off track because companies fail to plan the project realistically or they don’t execute or manage the project by the plan. Remember that major systems conversions are not just IT projects. Companies should maintain joint responsibility with the vendor in the project-planning process, maintenance of the project-plan status, as well as some degree of control over the implementation.

All key user departments should have representation on the project team, including the call center, website, fulfillment, management, merchandising, inventory control, marketing and finance. Team members should share responsibilities for conversion, training and successful completion of the project tasks.

The software vendor should have a time-tested project methodology and provide a high-level general plan. As the merchant client, your job is to develop the detailed plan with the vendor, backed up with detail tasks and estimates.

For example, a generalized plan may have a list of system modifications, but lack the details that need to be itemized. These may include research, specifications, sign-offs, program specs, programming, testing and sign-off, and the various levels of testing and program integration back into the base system.

Plan for contingencies, and try to keep disruptions to the business to a minimum. We have seen systems go live and with management initially unable to get their most frequently used reports this can be a big problem.

The systems project should have a senior manager who acts as the project sponsor. The project should be reviewed periodically by the steering committee to track its progress. This ensures that senior management on down to the department managers are committed to success.

Once you have a plan that makes sense, make sure you manage by the plan. This sounds elementary, but many companies and vendors stumble on it.

Early in the project publish a biweekly status report. Once you get within a few months, you may want to have weekly conference call meetings and status updates. Within 30 days of “go live,” hold daily meetings and list what needs to be achieved.

Project management is the discipline of planning, organizing, motivating, and controlling resources to achieve specific goals. A project is a temporary endeavor with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverables), undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value. The temporary nature of projects stands in contrast with business as usual (or operations), which are repetitive, permanent, or semi-permanent functional activities to produce products or services. In practice, the management of these two systems is often quite different, and as such requires the development of distinct technical skills and management strategies.

The primary challenge of project management is to achieve all of the project goals and objectives while honoring the preconceived constraints. The primary constraints are [scope](http://en.wikipedia.org/wiki/Scope_%28project_management%29), time, quality and budget. The secondary —and more ambitious— challenge is to optimize the allocation of necessary inputs and integrate them to meet pre-defined objectives.

**Documentation:**

Requirements come in a variety of styles, notations and formality. Requirements can be goal-like (e.g., distributed work environment), close to design (e.g., builds can be started by right-clicking a configuration file and select the 'build' function), and anything in between. They can be specified as statements in natural language, as drawn figures, as detailed mathematical formulas, and as a combination of them all.

A good architecture document is short on details but thick on explanation. It may suggest approaches for lower level design, but leave the actual exploration trade studies to other documents.

It is very important for user documents to not be confusing, and for them to be up to date. User documents need not be organized in any particular way, but it is very important for them to have a thorough [index](http://en.wikipedia.org/wiki/Index_%28publishing%29).

Change Management within ITSM (as opposed to software engineering or project management) is often associated with ITIL, but the origins of change as an IT management process predate ITIL considerably, at least according to the IBM publication A Management System for the Information Business.

In the ITIL framework, Change Management is a part of "Service Transition" - transitioning something newly developed (i.e. an update to an existing production environment or deploying something entirely new) from the Service Design phase into Service Operation (AKA Business As Usual) and aims to ensure that standardized methods and procedures are used for efficient handling of all changes.

**System Installation:**

Installation performed without using a [computer monitor](http://en.wikipedia.org/wiki/Computer_monitor) connected. In attended forms of headless installation, another machine connects to the target machine (for instance, via a [local area network](http://en.wikipedia.org/wiki/Local_area_network)) and takes over the display output. Since a headless installation does not need a user at the location of the target computer, unattended headless installers may be used to install a program on multiple machines at the same time.

An installation process that runs on a preset time or when a predefined condition transpires, as opposed to an installation process that starts explicitly on a user's command. For instance, a system willing to install a later version of a computer program that is being used can schedule that installation to occur when that program is not running. An operating system may automatically install a device driver for a device that the user connects. (See [plug and play](http://en.wikipedia.org/wiki/Plug_and_play).) Malware may also be installed automatically. For example, the infamous [Conficker](http://en.wikipedia.org/wiki/Conficker) was installed when the user plugged an infected device to his computer.

**7.1 USER TRAININGS:**

It forms the core of apprenticeships and provides the backbone of content at institutes of technology (also known as technical colleges or polytechnics). In addition to the basic training required for a trade, occupation or profession, observers of the labor-marketrecognize as of 2008 the need to continue training beyond initial qualifications: to maintain, upgrade and update skills throughout working life. People within many professions and occupations may refer to this sort of training as professional development.

* Training Tips
	+ Train people in groups, with separate training programs for distinct groups
	+ Select the most effective place to conduct the training
	+ Provide for learning by hearing, seeing, and doing
	+ Prepare effective training materials, including interactive tutorials

Rely on previous trainees

Convert a subset of the database.  Apply incremental updates on the new system or the old system to synchronize the updates taking place.  Use this method when the system is deployed in releases.  For example, updates are applied to the old system if a pilot project is applying updates against the new system so that areas outside the pilot have access to the data.

Some commentators use a similar term for workplace learning to [improve performance](http://en.wikipedia.org/wiki/Performance_improvement): "[training and development](http://en.wikipedia.org/wiki/Training_and_development)". There are also additional services available online for those who wish to receive training above and beyond that which is offered by their employers. Some examples of these services include career counselling, skill assessment, and supportive services. One can generally categorize such training as on-the-job or off-the-job:

On-the-job training method takes place in a normal working situation, using the actual [tools](http://en.wikipedia.org/wiki/Tool), equipment, documents or materials that trainees will use when fully trained. On-the-job training has a general reputation as most effective for vocational work. It involves Employee training at the place of work while he or she is doing the actual job. Usually a professional trainer (or sometimes an experienced employee) serves as the course instructor using hands-on training often supported by formal classroom training.

Off-the-job training method takes place away from normal work situations — implying that the employee does not count as a directly productive worker while such training takes place. Off-the-job training method also involves employee training at a site away from the actual work environment. It often utilizes lectures, case studies, role playing and simulation, having the advantage of allowing people to get away from work and concentrate more thoroughly on the training itself. This type of training has proven more effective in inculcating concepts and ideas.

* A more recent development in job training is the On the Job Training Plan, or OJT Plan. According to the United States Department of the Interior, a proper OJT plan should include: An overview of the subjects to be covered, the number of hours the training is expected to take, an estimated completion date, and a method by which the training will be evaluated.

**8. CONCLUSION**

 This project has been implemented successfully and the output has been verified. All the outputs are generating according to the given input. Data validations are done according to the user and admin input data. The patient’s user name and password are generated in admin login. All the generated username and passwords are getting login successfully. While creating the test details all the data are responding properly according to the input details.

 Secured login provided for both admin and patient. An individual master page has been created for both admin and patient. Data integrity has been verified wile uploading the test details. SQL data base has been successfully configured to storing the data in the database. Patients are provided with charts to verify the test history. Thus the project has been implemented successfully according to the commitment.

**9. FUTURE ENHANCEMENT**

 Even thou the system has been enhanced and planned well; still there are some more future enhancement to be done. Due to time constrain, the project has been stopped at this phase.

This application is developed under client server technology, but this application is not hosted in the web server. Hosting this application in web server makes real time.

This application can able to host in a cloud server, this makes improvements in the performance.

A dedicated mobile application can be implement for Android, IOS and windows.

This application can be centralized and can able to tied with hospitals also. This makes the lab assistant to send the test report to the concern doctor directly.

Patients can makes tests in their free insurance schemes

More comparison chars can be created

Data backups can be done from the admin side. Admin take back up for every month.

**10. BIBLIOGRAPHY**

Books reffered

1. Alistair McMonnies, “Object-oriented programming in ASP.NET”, Pearson Education, and ISBN: 81-297-0649-0, First Indian Reprint 2004.
2. Jittery R.Shapiro, “The Complete Reference ASP.NET” Edition 2002, Tata McGraw-Hill, Publishing Company Limited, New Delhi.
3. Robert D.Schneider, Jettey R.Garbus, “Optimizing SQL Server”, Second Edition, Pearson Education Asia, ISBN: 981-4035-20-3

Websites reffered :

1. http://www.microsoft.com/dotnet/visual basic
2. http://www.dotnetheaven.com
3. http://www.codeproject.com
4. http://www.Planetcode.com
5. http://www.stackoverflow.com

**11.SAMPLE SCREENS**

**Screen shots should taken by candidates only**

**Reason:**

**You can enter your name and project related information while taking Screen shots**

**12.SAMPLE CODE**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Web;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.Configuration;**

**using System.Data.SqlClient;**

**public partial class adcreatecustomer : System.Web.UI.Page**

**{**

 **SqlConnection con;**

 **SqlCommand cmd;**

 **string query, gen;**

 **public void data()**

 **{**

 **string connstring = WebConfigurationManager.ConnectionStrings["connection"].ConnectionString;**

 **con = new SqlConnection(connstring);**

 **con.Open();**

 **}**

 **private int random(int min, int max)**

 **{**

 **Random random = new Random();**

 **return random.Next(min, max);**

 **}**

 **protected void Page\_Load(object sender, EventArgs e)**

 **{**

 **if (rdmale.Checked == true)**

 **{**

 **gen = "Male";**

 **}**

 **else**

 **{**

 **gen = "Female";**

 **}**

 **}**

 **protected void txtname\_TextChanged(object sender, EventArgs e)**

 **{**

 **lblmes.Visible = false;**

 **txtage.Focus();**

 **}**

 **protected void btngenerate\_Click(object sender, EventArgs e)**

 **{**

 **lblusername.Visible = true;**

 **lblpassword.Visible = true;**

 **int a = random(100, 999);**

 **int b = random(10000, 99999);**

 **lblusername.Text = txtname.Text.Substring(0, 3) + a.ToString();**

 **lblpassword.Text = b.ToString();**

 **}**

 **protected void btncreate\_Click(object sender, EventArgs e)**

 **{**

 **data();**

 **query = "select username from customertbl where username='" + lblusername.Text + "'";**

 **cmd = new SqlCommand(query, con);**

 **SqlDataReader rd = cmd.ExecuteReader();**

 **if (rd.Read())**

 **{**

 **lblmes2.Visible = true;**

 **lblmes2.Text = "Already Exists";**

 **}**

 **else**

 **{**

 **lblmes2.Visible = false;**

 **data();**

 **query = "insert into customertbl(name,gender,age,bloodgroup,street1,street2,city,state,pin,phone,email,username,password)values('" + txtname.Text + "','" + gen + "','" + txtage.Text + "','" + ddbloodgroup.SelectedItem + "','" + txtstreet1.Text + "','" + txtstreet2.Text + "','" + txtcity.Text + "','" + txtstate.Text + "','" + txtpinno.Text + "','" + txtphoneno.Text + "','" + txtemail.Text + "','" + lblusername.Text + "','" + lblpassword.Text + "')";**

 **cmd = new SqlCommand(query, con);**

 **cmd.ExecuteNonQuery();**

 **con.Close();**

 **lblmes.Visible = true;**

 **lblmes.Text = "Customer Created";**

 **rdmale.Checked = true;**

 **rdfemale.Checked = false;**

 **txtname.Text = "";**

 **txtage.Text = "";**

 **txtstreet1.Text = "";**

 **txtstreet2.Text = "";**

 **txtcity.Text = "";**

 **txtstate.Text = "";**

 **txtpinno.Text = "";**

 **txtphoneno.Text = "";**

 **txtemail.Text = "";**

 **txtname.Focus();**

 **}**

 **rd.Close();**

 **con.Close();**

 **lblusername.Visible = false;**

 **lblpassword.Visible = false;**

 **}**

 **protected void btncancel\_Click(object sender, EventArgs e)**

 **{**

 **txtname.Text = "";**

 **txtage.Text = "";**

 **txtstreet1.Text = "";**

 **txtstreet2.Text = "";**

 **txtcity.Text = "";**

 **txtstate.Text = "";**

 **txtpinno.Text = "";**

 **txtphoneno.Text = "";**

 **txtemail.Text = "";**

 **txtname.Focus();**

 **}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Web;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.Configuration;**

**using System.Data.SqlClient;**

**public partial class admanagetest : System.Web.UI.Page**

**{**

 **SqlConnection con;**

 **SqlCommand cmd;**

 **string query;**

 **public void data()**

 **{**

 **string connstring = WebConfigurationManager.ConnectionStrings["connection"].ConnectionString;**

 **con = new SqlConnection(connstring);**

 **con.Open();**

 **}**

 **protected void Page\_Load(object sender, EventArgs e)**

 **{**

 **}**

 **protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)**

 **{**

 **GridView1.DataBind();**

 **}**

 **protected void GridView1\_RowUpdating(object sender, GridViewUpdateEventArgs e)**

 **{**

 **TextBox subtest = (TextBox)GridView1.Rows[e.RowIndex].FindControl("txtsubtestname");**

 **TextBox frmrange = (TextBox)GridView1.Rows[e.RowIndex].FindControl("txtfrmrange");**

 **TextBox frmmeasurement = (TextBox)GridView1.Rows[e.RowIndex].FindControl("txtfrmmeasurement");**

 **TextBox torange = (TextBox)GridView1.Rows[e.RowIndex].FindControl("txttorange");**

 **TextBox tomeasurement = (TextBox)GridView1.Rows[e.RowIndex].FindControl("txttomeasurement");**

 **string subtestname = GridView1.DataKeys[e.RowIndex].Values[0].ToString();**

 **data();**

 **query = "update newtesttbl set subtestname='" + subtest.Text + "', frmrange='" + frmrange.Text + "',fmeasurement='" + frmmeasurement.Text + "',torange='" + torange.Text + "',tmeasurement='" + tomeasurement.Text + "' where subtestname ='" + subtestname.ToString() + "'";**

 **SqlDataSource2.UpdateCommand = query;**

 **SqlDataSource2.Update();**

 **con.Close();**

 **GridView1.DataBind();**

 **}**

 **protected void GridView1\_RowDeleting(object sender, GridViewDeleteEventArgs e)**

 **{**

 **string subtestname = GridView1.DataKeys[e.RowIndex].Values[0].ToString();**

 **data();**

 **query = "delete from newtesttbl where subtestname ='" + subtestname.ToString() + "'";**

 **SqlDataSource2.DeleteCommand = query;**

 **SqlDataSource2.Delete();**

 **GridView1.DataBind();**

 **}**

**}**