\

**ACKNOWLEDGEMENT**

No project is ever complete without guidance of those experts who have already treaded this path before and have become master of it and as result, our teachers. So we would like to take this opportunity to thanks all those individuals who have helped us in visualizing this project.

We express our deep gratitude to our project guide Prof. **Ms. Rohini Inamdar** for providing timely assistance to queries and guidance that gave us owing to his experience in this field for the last many years. he has indeed been a lighthouse for us through this journey.

We extend our sincere appreciation to all our professors from **Yashwantrao Chavan College** for their valuable insights and tips during designing of the project. Their contributions have been valuable in so many that we find it difficult to acknowledge each of them individually.

CHETAN Y. SINGH

**ABSTRACT**

The project focuses on the development skills of the user. The knowledge acquired proves to be helpful and effective, only if it is implemented. The implementation of the knowledge acquired is through project and case studies based on real time scenarios

The “RESULT MANAGEMENT SYSTEM” undertaken as a project is based on relevant technologies. The projects are done individual where the student learn to work Alone, manage project schedules, follow the best coding standards and conventions and adhere to project documentation standards.

Here in in the project we use front end as VB .NET and backend as My SQL. We store information about the students and their mark who have given exam. All these details are stored in the database by entering into VB .NET and the details can be displayed from the database by writing the query in My SQL. This project involves several modules to carry out each action.

WE created database for storing the details in tables. We created table for USER DETAILS, a table for storing MARKS and another table for storing details of the students who have give the Exam namely STUDENT DETAILS.

We can also store the details at run time, that is we can enter the student details by entering the details of the student separately at run time. Likewise the many setting details can also be entered at run time. We can either store in a database and see the details when required or see the details at run time by selecting the option display.

Hence in the existing system for RESULT MANAGEMENT SYSTEM, the performance evaluation system and the maintenance are done manually. The proposed system will maintain all the information in a standard database and will be able to generate reports as and when necessary. Accessing and the user has the right to search based on the above mentioned criteria’s.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Topic** | **Page no.** |
| **1** | **Introduction** | **1** |
| **2** | **Objective and Scope of the project** | **4** |
| **3** | **Feasibility Study Report** | **6** |
| **4** | **System Analysis** | **10** |
| **5** | **Requirement analysis report** | **11** |
| **6** | **System design** | **13** |
| **7** | **Methodology Adopted** | **39** |
| **8** | **System Maintenance** | **48** |
| **9** | **System Testing and Result** | **54** |
| **11** | **Coding and Snapshot** |  |
| **12** | **Conclusion** |  |

**Objective and Scope of the project**

The project aims to develop software, which will store data about student’s and their marks info, the Course information and project reports aims at introduction to Result Management System developed to maintain and manipulate all the transaction performed in the Result Management.

This report depicts the information used for the system development for development for detail information scope of the project along with the table design and screen layouts are provided.

In this project .I have tried to work upon each aspect of the services provide as well as process taking place.

**Objectives of the Project**

The objective of the project is to create a system that -

* Provides security to the data.

This can achieved by using login and password method based on the user

* According to the Manager, provides the relevant data

Depending upon the requirements of the Manager, he can easily get access to all data stored by generating various data reports.

* Handles the user and student marks information.

The system is build to efficiently store all the relevant data about the people involved for the proper function

* Efficient data storage that will reduce the redundancy of data.

The data efficiently stored in the system in tubular form and it provides easy access to this data.

**Scope of the project**

* All the calculation part would be handled using this software which were done manually.
* The receipt can be generated to view the data such as fees information, etc. Hence the user can view data through receipt.
* The data is stored in a single database providing real time data across applications throughout the System. This encompasses human (and paper-based) information processing as well as data processing machines.
* The advantage of the system is that the Manager can store all the information about the Students or Marks under him/her.
* Application should generate final receipt.

**FEASIBILITY STUDY REPORT**

**Feasibility study :**

Feasibility is the process of implementing and designing the records of Application to keepsufficient manner and information provide to the user. Feasibility analysis determines whether it is operationally, economically and technically able to proceed with a particular action.

A feasibility study is an analysis of the viability of an idea, The feasibility study focuses on helping answer the essential question of “should we proceed with the proposed project idea?” All activated of the study are directed toward healing answer this question.

Feasibility studies can be used in many ways but primarily focus on proposed business ventures. Farmers and others with a business idea should contact a feasibility study to determine in viability of their idea before proceeding with the development of the business. Determining early-on that a business idea will not work saves time, money and heartache later.

A feasibility study is only one step in the business idea assessment and business development process. Reviewing this process and reading the information below will help But the role of the feasibility study in perspective.

**As there are three types of feasibility involved in feasibility report. They are as follows.**

1. Economical feasibility
2. Technical feasibility
3. Behavioral feasibility

**Economical Feasibility :**

Economic Feasibility means, the project benefit of the proposed system outweigh.

The estimated cost usually considered the whole cost of ownership (TCO) WHICH MUST INCLUDES:

1 - Outgoing support

2 - Maintenance cost

3 – Acquisition cost

To determine TCO, the analyst must estimated cost in each following areas:

People, include IT staff and user, Hardware and equipment, Software, house development, purchase from vendors, Formal and informal training, Licenses and feeds, Consulting expenses, Facility costs, Tangible benefits and intangible benefits.

The feasibility is most commonly known as cost / benefit analysis. As my project is Application Development (College Application), it is totally based on Application and software. There is no need of any additionally hardware. Therefore there is no need of extra cost.

We developed this project by using VB .NET as frontend and My SQL as a backend. VB .NET are open source language that makes my project is economically feasible.

**Technical Feasibility :**

Technological Feasibility is established upon completing of a detailed program design or a working model [in regard to setting standards for software accounting].

This is important in regards to how to treat costs incurred with production of software products. before [‘Technological Feasibility’] is established, costs are a ‘Research + Development’ expense. Once ‘Technological Feasibility’ established costs are capitalized and amortized to the current and future periods.

My Project has Technical Feasibility because it is developed by using VB .NET and My SQL which are today most widely us in Windows Programming for making High Level Application.

This project is feasible because,

* Easy language for programmer.
* Fast project planning with help of these languages.

**Behavioral Feasibility :**

People are inherently to change and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development of a computerized system.

It is common knowledge that computer installation have something to do with turnover, transfer, retraining, and changes in employee job status

As this project is based on Application, sp searching of college information is very easy. Access information from anywhere. Helpful to students and teachers. This project will provide more information in less time.

* It is simplest and cheapest way, by which college can get more publicity, marking etc.
* It is also very simple and cheap way to get information about the students Result for college.
* It is very easy to publish and update this Application

System Analysis and User Requirement

**1) Analysis**

System analysis is the way of studying a system with an eye on solving its problem using computer. It is the most essential part of the development of a project of a system analysis. System analysis consists of system element, process and technology. To analyze a system, has to study the systems in details.

The analyst has to understand the functioning and concept of the system in detail, before design the appropriate computer based system that will meet all the requirements of the existing system. The system analyst has to carry out a customary approach to use the computer for problem solving.

The above steps constitute the logical framework for the system analysis. After the preliminary investigation and feasibility study, the scope of the defined and comparable items are set forth and hence detailed investigation is executed. This allows the system analyst to comprehend the full scope of the project. Soon after the implementation of the newly developed system, followed by the training of the users, the system analysis is included.

**2) Design**

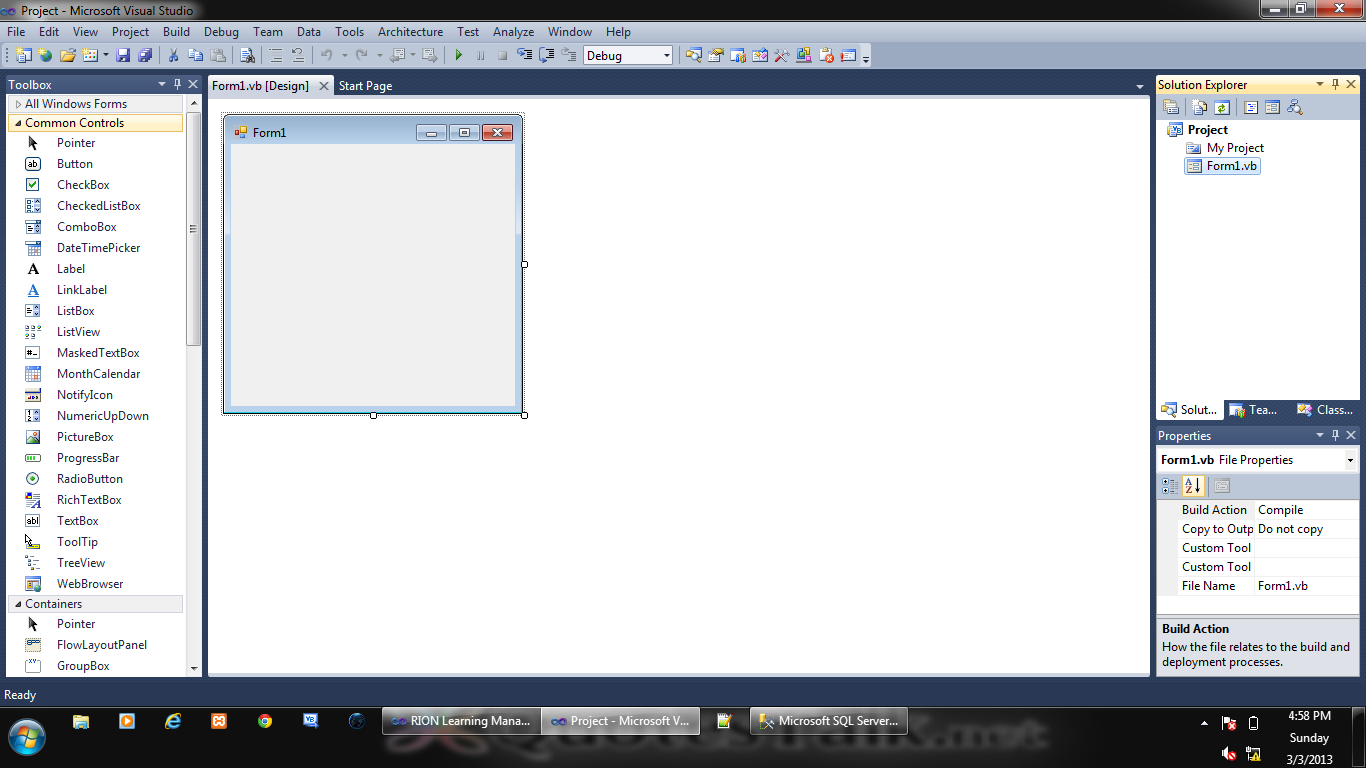
It includes important things like as below:

1. What the interface will look like – use of various controls and so on
2. The files needed - their structure, organization and processing
3. How the data will be validated
4. The procedures needed

The goal of the design process is to find the best possible design. The most important criterion to judge a design is verifiability, reliability and maintainability. In this project the interface used is user friendly and the front-end used is VB.Net and backend used is My SQL.

**Software requirement**

**Front End : Visual Studio 2010**

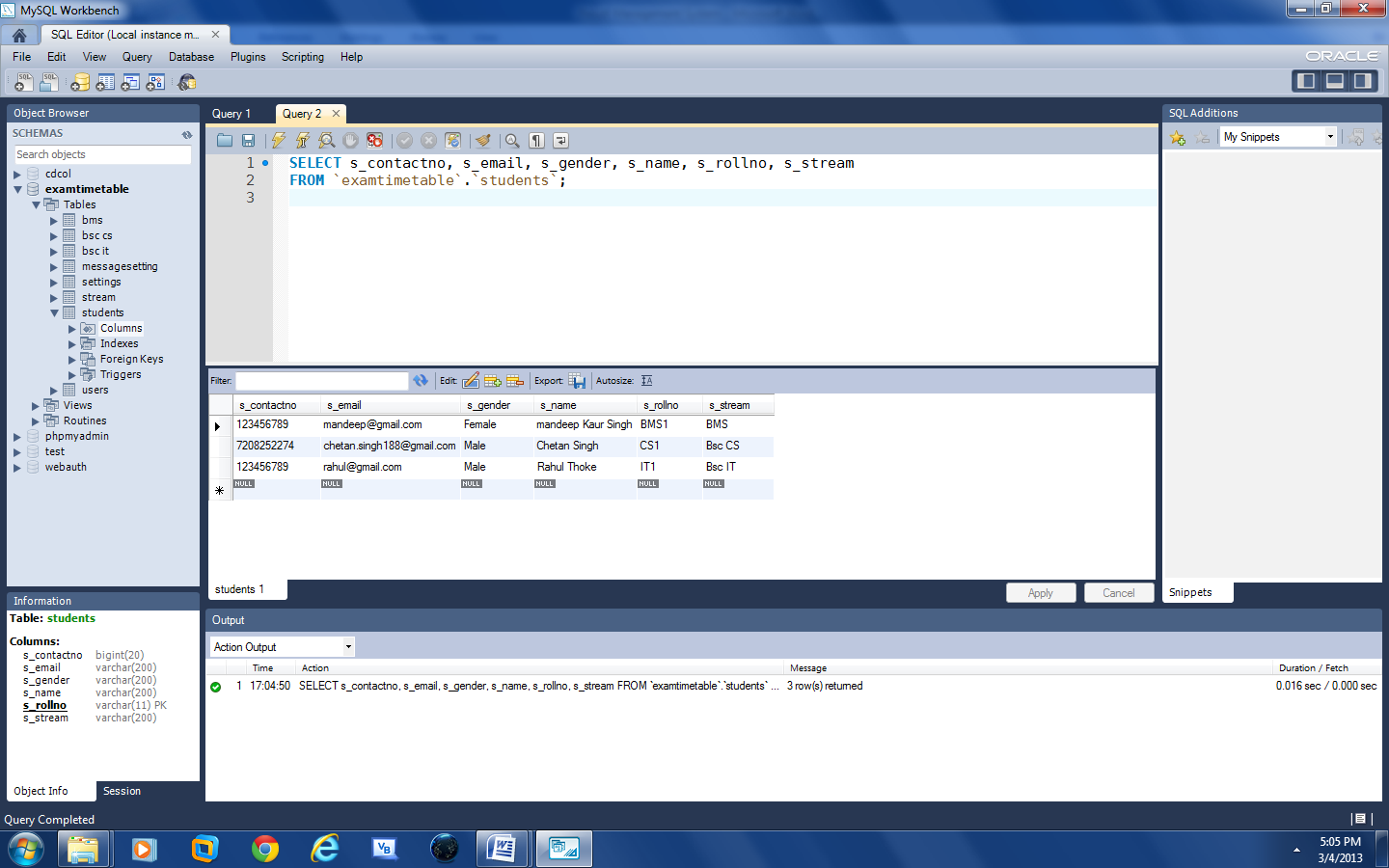
****

**Features:**

* Visual basic is event driven programming language. It has functionality of GUI (Graphical User Interface) which helps the programmer to have efficient functionality and better development
* Visual basic has many tools help in project development Process.
* Active X technologies allow you to use the functionality provided by other applications, Microsoft word processor, Microsoft Excel Spreadsheet and other Windows applications. You can even automate applications and object created using the professional or Enterprise Edition of Visual Basic.
* Data access features allow you to create database, front-end applications and scalable server side components for most popular scalable formats including Microsoft SQL server and other Enterprise Level database.

It has facility of make a executable version of current project which will run on any environment and find the necessary driver

**Back End : My SQL**

****

* Works on many different platforms
* Uses multi-layered server design with independent modules.
* Provides transactional and non transactional storage engines.
* Uses a very fast thread-based memory allocation system.
* Executes very fast joins using an optimized nested-loop join.
* Implements in-memory hash tables, which are used as temporary tables.
* Implements SQL functions using a highly optimized class library that should be as fast as possible. Usually there is no memory allocation at all after query initialization.
* Provides the server as a separate program for use in a client/server networked environment, and as a library that can be embedded (linked) into standalone applications. Such applications can be used in isolation or in environments where no network is available.

System Planning (Gantt chart)

A Gantt chart is a graphical representation of the duration of tasks against the progression of time. A Gantt chart is a useful tool for planning and scheduling projects.

A Gantt chart is helpful when monitoring a project's progress. A Gantt chart allows you to assess how long a project should take. A Gantt chart lays out the order in which tasks need to be carried out. A Gantt chart helps manage the dependencies between tasks .A Gantt chart allows you to see immediately what should have been achieved at a point in time. A Gantt chart allows you to see how remedial action may bring the project back on course.

A Gantt chart is a horizontal bar chart developed as a production control tool in 1917 by Henry L. Gantt, an American engineer and social scientist. Frequently used in project management, a Gantt chart provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project. Gantt charts may be simple versions created on graph paper or more complex automated versions created using project management applications such as Microsoft Project or Excel.

A Gantt chart is constructed with a horizontal axis representing the total time span of the project, broken down into increments (for example, days, weeks, or months) and a vertical axis representing the tasks that make up the project (for example, if the project is outfitting your computer with new software, the major tasks involved might be: conduct research, choose software, install software). Horizontal bars of varying lengths represent the sequences, timing, and time span for each task. Using the same example, you would put "conduct research" at the top of the vertical axis and draw a bar on the graph that represents the amount of time you expect to spend on the research, and then enter the other tasks below the first one and representative bars at the points in time when you expect to undertake them. The bar spans may overlap, as, for example, you may conduct research and choose software during the same time span. As the project progresses, secondary bars, arrowheads, or darkened bars may be added to indicate completed tasks, or the portions of tasks that have been completed. A vertical line is used to represent the report date.

Gantt charts give a clear illustration of project status, but one problem with them is that they don't indicate task dependencies - you cannot tell how one task falling behind schedule affects other tasks. The PERT chart, another popular project management charting method, is designed to do this. Automated Gantt charts store more information about tasks, such as the individuals assigned to specific tasks, and notes about the procedures. They also offer the benefit of being easy to change, which is helpful. Charts may be adjusted frequently to reflect the actual status of project tasks as; almost inevitably, they diverge from the original plan.

**Detailed Life Cycle**

**EVENT TABLE:**

1. List of events gathered during analysis are centered in even table. An Event table includes rows and columns representing events and their details respectively.
2. The rows record the information about every event. The columns record information about the event.

**TRIGGER:**

Tells the system that event has occurred for .an External event the

arrival of data the System must process is as trigger.

**SOURCE:**

It’s the External agent or actor that Supplies data to the System.

**ACTIVITY:**

Activity is the System reaction to the event when the student enquire for courses the System carries out the activity create new enquiry.

**RESPONSE:**

Response is an output from the system. There could be several Responses to one activity.

**DESTINATION:**

Destination is where the Response is sent which is again an external Manager or actor.An event could have no response

E.g.:- Student wants to update info, the info is recorded in the database but no output needs to be produced.

Event table is a convenient way of recording Key info about the requirement of the info system.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Event | Trigger | Source | Activity | Response | Destination |
| 1). | Enter subjects | New Student entry | Student | Make new entry | Confirm | Admin |
| 2). | Update subject details | Update Student | Student | Update and delete student | Change confirm | Admin |
| 3). | Enter result details | New Result Entry | Student | Make new entry | Confirm | Admin |
| 4). | Update result details | Update Result | Student | Update and Delete Result | Change confirm | Admin |
| 5). | Enter subject | New Subject | Subject | Make new Entry | Confirm | Admin |
| 6). | Edit subject details | Update subject | Subject | Update and Delete subject | Change confirm | Admin |
| 7). | Enter detail of student email | New student email | Student | Make new entry | Confirm | Admin |
| 8). | Edit detail of student | Update student email | Student | Update and delete student email | Change confirm | Admin |
| 9). | Generate Report | Result generated | Admin | Checking student marks generated result | Confirm result | admin |

**USE CASE DIAGRAM:**

A **use case diagram** in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

* **Use cases**

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

* **Actors**

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

* **System boundary boxes (optional)**

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not.

Connection links

Actor

**METHODS**

Use case

A] Most common methods is to use event table each event table is analyzed to determine way the system is used to support that event. The actor who initiate the event and other use that may be invoked because of the event are also determined.

B] Each event will become a use case. In some instances an event may spawn more than one use case.

C] Use Case can also be developed by first identifying all actors who use the system. . Actors are listed using the event table by looking at the trigger and source columns in the event table.

D] Actors could also be someone other than the course. In a system an actor may play various and specific role. It is thus necessary to identify every possible role that will use the system.

**Administrator**

**Figure: Administrator as Actor**

**Student**

**Figure: Student as Actor**

**ACTIVITY DIAGRAM**

Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows. The most important shape types:

* Rounded rectangles represent activities;
* Diamonds represent decisions;
* Bars represent the start (split) or end (join) of concurrent activities;
* A black circle represents the start (initial state) of the workflow;
* An encircled black circle represents the end (final state).

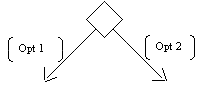
**Initial Activity:** This shows the starting point or first activity of the flow. Denoted by a solid circle. This is similar to the notation used for Initial State.



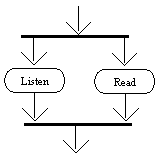
**Activity:** Represented by a rectangle with rounded (almost oval) edges.



**Decisions:** Similar to flowcharts, a logic where a decision is to be made is depicted by a diamond, with the options written on either sides of the arrows emerging from the diamond, within box brackets.



**Concurrent Activities:** Some activities occur simultaneously or in parallel. Such activities are called concurrent activities. For example, listening to the lecturer and looking at the blackboard is a parallel activity. This is represented by a horizontal split (thick dark line) and the two concurrent activities next to each other, and the horizontal line again to show the end of the parallel activity.



**Final Activity:** The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity.





****

**CLASS DIAGRAM :**

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

**Definition -**

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modeling paradigms have evolved.

1) In a class diagram, the classes are arranged in groups that share common characteristics.

2) A class diagram resembles a flowchart in which classes are portrayed as boxes, each box having three rectangles inside.

3) The top rectangle contains the name of the class; the middle rectangle contains the attributes of the class; the lower rectangle contains the methods, also called operations, of the class.

4) Lines, which may have arrows at one or both ends, connect the boxes. These lines define the relationships, also called associations, between the classes.

**A class with three sections.**

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

In the system design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the statical relations between those objects. With detailed modeling, the classes of the conceptual design are often split in a number of subclasses.

In order to further describe the behavior of systems, these class diagrams can be complemented by state diagram or UML state machine.

Marks

Roll\_no

Year

Sem

Subject1

Subject2

Subject3

Subject4

Subject5

Subject6

Total

Percentage

Grade

Result

add()

edit()

delete()

Students

s\_rollno

s\_stream

s\_name

s\_email

s\_gender

s\_contactno

add()

edit()

delete()

Message

Message\_id

Message\_stream

Message\_message

add()

edit()

delete()

Students

Stream

sem

Subject1

Subject2

Subject3

Subject4

Subject5

Subject6

add()

edit()

delete()

**OBJECT DIAGRAM**

An **object diagram** in the Unified Modeling Language , is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

1)An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances.

2)A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time.

3)Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.

Object Diagram or Instance Diagram illustrates objects links.

As in the case of class diagrams represent the static structure .The notation is used for Object Diagram is derived from that of class Diagram elements that are instance are underlined.

Object Diagram is preliminary used to show a context before or after an interaction

Representation of Objects

Each Object is represented by rectangle which contains either the name of the objects and class of the object (separated by colon), or only the objects class (in which case the Object is said to be anonymous).

Marks

Roll\_no

Year

Sem

Subject1

Subject2

Subject3

Subject4

Subject5

Subject6

Total

Percentage

Grade

result

add()

edit()

delete()

Students

s\_rollno

s\_stream

s\_name

s\_email

s\_gender

s\_contactno

add()

edit()

delete()

Message

Message\_id

Message\_stream

Message\_message

add()

edit()

delete()

Students

Stream

sem

Subject1

Subject2

Subject3

Subject4

Subject5

Subject6

add()

edit()

delete()

**SEQUENCE DIAGRAM**

A **sequence diagram** in Unified Modeling Language is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

1) A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

2) If the lifeline is that of an object, it demonstrates a role.

**Note: That leaving the instance name blank can represent anonymous and unnamed instances**.

3) In order to display interaction, messages are used. These are horizontal arrows with the message name written above them.

4) Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages.

5) Activation boxes, or method call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML).

Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing.

System

Admin

Student

Enter Course Details

Enter Subject Details

Enter Student Details

Enter Marks

Send Email

Generate Mark sheet

**ER DIAGRAM**

**Entity:**

The things the System needs to store information about in the traditional approach to information system.

**Tables:**

Tables are collection of rows &columns. They consist of information about the attribute of each entity.

**Relationships:**

A naturally occurring association among the entities is called a Relationship. Relationship exists in four types.

1. **Binary Relationship:**

Relationships between two different types of entities.

1. **Unary Relationship:**

Relationship between two entities of same type.

1. **Ternary relationship:**

A Relationship between three different types of entities.

1. **N-ary relationship**:

A Relationship between n different types of entities.

Student

Select

Stream

Gets

Generate

Mark Sheet

Marks

**COMPONENT DIAGRAM**

In the Unified Modeling Language, a **component diagram** depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole

1) Components are wired together by using an *assembly connector* to connect the required interface of one component with the provided interface of another component. This illustrates the *service consumer - service provider* relationship between the two components.

2) An *assembly connector* is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port."

3) When using a component diagram to show the internal structure of a component, the provided and required interfaces of the encompassing component can delegate to the corresponding interfaces of the contained components.

4) A *delegation connector* is a "connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component’s parts."

5) Component diagrams are different in terms of nature and behavior. Component diagrams are used to model physical aspects of a system.

Now the question is what are these physical aspects? Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node.

So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

So the purpose of the component diagram can be summarized as:

* Visualize the components of a system.
* Construct executables by using forward and reverse engineering.
* Describe the organization and relationships of the components.

Student

Course

Marks

Result

Management .exe

Result

**Database**

**PACKAGE DIAGRAM**

UML Packages are a grouping of objects into sets of objects that provide related services. The package has responsibilities that are strongly related. The package has low coupling and low cohesion with respect to interfacing with other packages in the system. Package diagrams can use packages containing use cases to illustrate the functionality of a software system.

Package diagrams can use packages that represent the different layers of a software system to illustrate the layered architecture of a software system. The dependencies between these packages can be adorned with labels / stereotypes to indicate the communication mechanism between the layers.

Package diagram shows the arrangement and organization of model elements in middle to large scale project. Package diagram can show both structure and dependencies between sub-systems or modules.

It is a high level diagram that identifies the major components of a system only two symbols are used in package diagram.

**Tabbed Rectangle**

It identifies the major system and the subsystem enclosure of sub system within the primary system indicates that it’s a part of major system.

**Dashed Arrow:**

The arrow is a dependency arrow. The arrow tail is connected to the independent package.

Result Main

Result Management

Result

Result Management

Send Email

Branch Email

Marks

Marks

Average

Grade

Student

Stream

Email

Marks

Website

Fetching Marks

Displaying Marks

**DEPLOYMENT DIAGRAM**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server.

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

1) The deployment diagram shows how a system will be physically deployed in the hardware environment. Its purpose is to show where the different components of the system will physically run and how they will communicate with each other.

Since the diagram models the physical runtime, a system's production staff will make considerable use of this diagram.

2) The notation in a deployment diagram includes the notation elements used in a component diagram, with a couple of additions, including the concept of a node. A node represents either a physical machine or a virtual machine node (e.g., a mainframe node).

3) To model a node, simply draw a three-dimensional cube with the name of the node at the top of the cube. Use the naming convention used in sequence diagrams.

In a deployment diagram the focus is an what software element uses which hardware that is how the system is deployed. The diagram shows where and how the system components will be deployed.

It is a specific map of the physical layout of the system. A Deployment diagram system it a production or test environment.

CPU

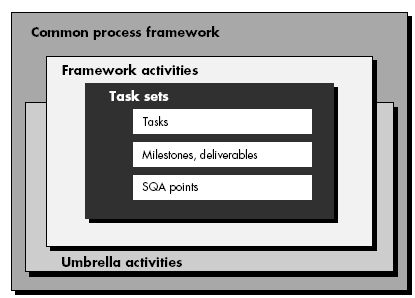
PRINTER

DB Source Result and Management

Application Result Management

Methodology Adopted

A software process can be characterized as shown in below figure. A common process framework is established by defining a small number of framework activities that are applicable to all software projects, regardless of their size or complexity. A number of task sets—each a collection of software engineering work tasks, project milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement2—overlay the process model. Umbrella activities are independent of any one framework activity and occur throughout the process.

****

**The Software Process**

In recent years, there has been a significant emphasis on “process maturity.” The Software Engineering Institute (SEI) has developed a comprehensive model predicated on a set of software engineering capabilities that should be present as organizations reach different levels of process maturity. To determine an organization’s current state of process maturity, the SEI uses an assessment that results in a five point grading scheme. The grading scheme determines compliance with a capability maturity model (CMM) [PAU93] that defines key activities required at different levels of process maturity. The SEI approach provides a measure of the global effectiveness of a company's software engineering practices and establishes five process maturity levels that are defined in the following manner:

**Level 1: Initial.** The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

**Level 2: Repeatable.** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

**Level 3: Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process fordeveloping andsupporting software. This level includes all characteristics defined for level 2.

**Level 4: Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3.

**Level 5: Optimizing.** Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.

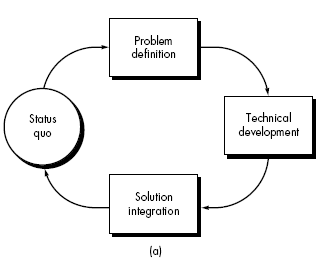
The five levels defined by the SEI were derived as a consequence of evaluating responses to the SEI assessment questionnaire that is based on the CMM. The results of the questionnaire are distilled to a single numerical grade that provides an indication of an organization's process maturity.

The SEI has associated key process areas (KPAs) with each of the maturity levels. The KPAs describe those software engineering functions (e.g., software project planning, requirements management) that must be present to satisfy good practice at a particular level. Each KPA is described by identifying the following characteristics:

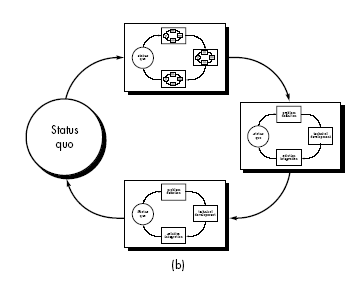
* **Goals—**the overall objectives that the KPA must achieve.
* **Commitments—**requirements (imposed on the organization) that must be met to achieve the goals or provide proof of intent to comply with the goals.
* **Abilities—**those things that must be in place (organizationally and technically) to enable the organization to meet the commitments.
* **Activities—**the specific tasks required to achieve the KPA function.
* **Methods for monitoring implementation—**the manner in which the activities are monitored as they are put into place.
* **Methods for verifying implementation—**the manner in which proper practice for the KPA can be verified.

Eighteen KPAs (each described using these characteristics) are defined across the maturity model and mapped into different levels of process maturity.

To solve actual problems in an industry setting, a software engineer or a team of engineers must incorporate a development strategy that encompasses the process, methods, and tools layers and the generic phases. This strategy is often referred to as a process model or a software engineering paradigm. A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required. In an intriguing paper on the nature of the software process, L.B. S. Raccoon [RAC95]uses fractals as the basis for a discussion of the true nature of the software process.

****

**The Phases of a problem-solving loop**

****

**The Phases within phases of a problem-solving loop**

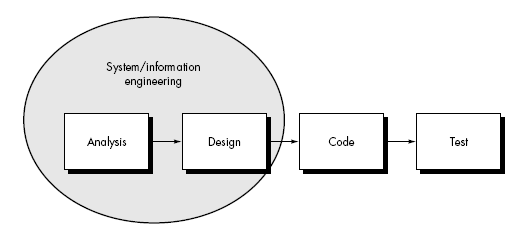
All software development can be characterized as a problem solving loop in which four distinct stages are encountered: status quo, problem definition, technical development, and solution integration. Status quo “represents the current state of affairs” [RAC95]; problem definition identifies the specific problem to be solved; technical development solves the problem through the application of some technology, and solution integration delivers the results (e.g., documents, programs, data, new business function, new product) to those who requested the solution in the first place. This problem solving loop applies to software engineering work at many different levels of resolution. It can be used at the macro level when the entire application is considered, at a mid-level when program components are being engineered, and even at the line of code level. Therefore, a fractal4 representation can be used to provide an idealized view of process. In Figure 3.2b, each stage in the problem solving loop contains an identical problem solving loop, which contains still another problem solving loop (this continues to some rational boundary; for software, a line of code).

**The Linear Sequential Model**

Sometimes called the classic life cycle or the waterfall model, the linear sequential model suggests a systematic, sequential approach5 to software development that begins at the system level and progresses through analysis, design, coding, testing, and support.

Figure 3.3 illustrates the linear sequential model for software engineering. Modeled after a conventional engineering cycle, the linear sequential model encompasses the following activities:

System/information engineering and modeling. Because software is always part of a larger system (or business), work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software. This system view is essential when software must interact with other elements such as hardware, people, and databases. System engineering and analysis encompass requirements gathering at the system level with a small amount of top level design and analysis. Information engineering encompasses requirements gathering at the strategic business level and at the business area level.

****

**The Linear Sequential Model**

**Software requirements analysis.** The requirements gathering process is intensified and focused specifically on software. To understand the nature of the program(s) to be built, the software engineer ("analyst") must understand the information domain for the software, as well as required function, behavior, performance, and interface. Requirements for both the system and the software are documented and reviewed with the customer.

**Design.** Software design is actually a multistep process that focuses on four distinct attributes of a program: data structure, software architecture, interface representations, and procedural (algorithmic)

detail. The design process translates requirements into a representation of the software that can be assessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.

**Code generation.** The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanistically.

**Testing.** Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals; that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

**Support.** Software will undoubtedly undergo change after it is delivered to the customer (a possible exception is embedded software). Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment (e.g., a change required because of a new operating system or peripheral device), or because the customer requires functional or performance enhancements. Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one.

The linear sequential model is the oldest and the most widely used paradigm for software engineering. However, criticism of the paradigm has caused even active supporters to question its efficacy [HAN95]. Among the problems that are sometimes encountered when the linear sequential model is applied are:

* Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.
* It is often difficult for the customer to state all requirements explicitly. The linear sequential model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
* The customer must have patience. A working version of the program(s) will not be available until late in the project time-span. A major blunder, if undetected until the working program is reviewed, can be disastrous.

In an interesting analysis of actual projects Bradac [BRA94], found that the linear nature of the classic life cycle leads to “blocking states” in which some project team members must wait for other members of the team to complete dependent tasks. In fact, the time spent waiting can exceed the time spent on productive work! The blocking state tends to be more prevalent at the beginning and end of a linear sequential process. Each of these problems is real. However, the classic life cycle paradigm has a definite and important place in software engineering work. It provides a template into which methods for analysis, design, coding, testing, and support can be placed. The classic life cycle remains a widely used procedural model for software engineering. While it does have weaknesses, it is significantly better than a haphazard approach to software development.

The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back.

**Advantages:**

1. Testing is inherent to every phase of the waterfall model
2. It is an enforced disciplined approach
3. It is documentation driven, that is, documentation is produced at every stage
4. Simple and easy to use.
5. Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
6. Phases are processed and completed one at a time.
7. Works well for smaller projects where requirements are very well understood.

**Disadvantages:**

1. It only incorporates iteration indirectly, thus changes may cause considerable confusion as the project progresses.
2. As The client usually only has a vague idea of exactly what is required from the software product, this WM has difficulty accommodating the natural uncertainty that exists at the beginning of the project.
3. The customer only sees a working version of the product after it has been coded. This may result in disaster if any undetected problems are precipitated to this stage.
4. Adjusting scope during the life cycle can kill a project
5. No working software is produced until late during the life cycle.
6. High amounts of risk and uncertainty.
7. Poor model for complex and object-oriented projects.
8. Poor model for long and ongoing projects.
9. Poor model where requirements are at a moderate to high risk of changing.

System Implementation

In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system. Many implementations may exist for a given specification or standard. For example, web browsers contain implementations of World Wide Web Consortium-recommended specifications, and software development tools contain implementations of programming languages.

In the IT Industry, implementation refers to post-sales process of guiding a client from purchase to use of the software or hardware that was purchased. This includes Requirements Analysis, Scope Analysis, Customizations, Systems Integrations, User Policies, User Training and Delivery.

Software Implementations involve several professionals that are relatively new to the knowledge-based economy such as Business Analysts, Technical Analysts, Solutions Architects, and Project Managers.

In this project the hardware and software details are as given below.

**Software Specification**

1. **Software Specification :**

1. Front End **:** Visual Basic Ultimate Edition 2010
2. Back End **:** My SQL
3. Data Report

2. **Hardware Specification:**

1. 256 RAM (Recommended)
2. 500 MB Disk Space (Recommended)
3. Extra as per second
4. **Operating System**
5. Windows XP
6. Windows 7
7. Windows 8

System Maintenance

1) There is no need of in depth knowledge of VB.Net and My SQL. Basic knowledge of maintaining any software is required.

2) The system can be modified as and when required.

3) Proper training of how to handle the software is given.

4) The user password can be changed so as to protect the database, since database is viewed only in insurance agent section and not in guest section.

5) Validation is done properly and even if the user is not entering proper data it can be viewed and deleted from the database.

6) Comments are given in code section so that the insurance agent can understand what exactly the code does.

7) Backup of database should be maintained by writing the important data to CD on a daily or weekly basis.

# System Maintenance

The basic routine maintenance tasks are:

* Data backup
* Malware management
* File system maintenance

**Backup**

To backup is to create a redundant copy, so that if anything should happen to the original file, you have recourse to the backup. The process can be as simple as copying files to diskettes, but this soon becomes a problem where files are too big for diskette, where there are too many files, or where too many diskettes are required.

A better solution is to use an archives (such as WinZip) or a backup utility to create a single compressed file from a collection of data files, and to split this over as many diskettes as required. This uses fewer diskettes and allows large files to be backed up even if the file is larger than a diskette can hold.

For large data sets, you may need to use a bulk storage medium such as tape, Zip disk, CDR or similar. These are generally faster and more reliable than diskettes.

In my system the reports can be exported anywhere and can be saved.

## Malware Management

There's more on safe computing and malware. Malware includes viruses, worms, Trojans, and increasingly invasive commercial applications, and management has several parts:

1. Risk avoidance and evaluation - choice of applications and system setup
2. Risk avoidance and evaluation - user education and safe computing practice
3. Risk detection and destruction - choice and use of antivirus software
4. Keeping abreast of malware - antivirus updates and ongoing user education

Simply running an antivirus utility is not enough, even if it is kept up to date.

For best performance, you can use on-demand rather than on-access antivirus scanners - but this requires the user to know when to use this, and act accordingly.

Updating an antivirus generally involves these steps:

* Go to antivirus vendor's web site via (say) Internet Explorer
* Navigate to the download section of the site
* Download any updates that are relevant, noting where these are saved
* Extract files from downloaded archive to the antivirus program directory

Some Windows-based antivirus utilities may automate this process to some extent, by accessing the Internet directly from within the program. You should check for updates at least once a week, and make sure your antivirus data files do not become more than a month out of date.

## File System Maintenance

Much can be done during system setup to improve the survivability, maintainability and recoverability of the file system and its data, as discussed on the data management page. Thereafter, there are three tasks required on a regular basis:

1. Check that sufficient free space is available; ideally 50M+ on C: volume
2. Check the file system for errors, and manage these
3. Defragment the file system once it is known to be error-free

The tools used here are Windows Explorer (or its "My Computer" incarnation), Scandisk, and Defrag. If free space is low, you can clear .TMP files from the Windows base directory.

**Limitations**

* The newly entered data for policies can be viewed in list view control and not in a separate form as it is done for displaying other policy information
* There is no separate backup kept so it has to be done manually by writing it to CD.
* No New user account can be added so there is only two sections- Guest and Insurance Agent.

Cost and Benefit Analysis

**Cost and Benefit analysis is three step process**

**1st step:-**

Estimate the anticipate development & operational cost.

* + **Development Cost :-**

Cost occurred during the development of the system.

* + **Operational Cost :-**

Cost that will occur after the installing of the system.

**2nd step:-**

Estimate the anticipate financial benefits.

* + **Financial Benefits :-**

They are expected annual saving or increase in revenue derived from the installation of the new system.

**3rd step :-**

The cost and benefit analysis is calculated based on detailed estimate cost & benefits.

**BENEFIT ANALYSIS**

After installing the Result Management System following are the benefits expected,

* Reducing staff due to automating function or increasing efficiency.
* Maintaining constant staff for project handling.
* Reducing bad accounts or bad credit losses.
* Reducing paper work cost with electronic data exchange.

System Testing:

The main objective of system testing is to find the errors or bugs in the system and to see whether the system fulfills expectations of the user. Being an accounting system there is no room for any errors or bugs to creep in; as a result the system has gone through vigorous testing at the regular intervals.

We planned a test plan before the start of our project. Our testing includes the following steps:

**1.** **Validation for all fields:**

For example certain fields are to be filled with numeric data. If alphabets are typed then a error message box appears reminding the user to enter valid data in the fields.

**2. Trapping of error:**

If certain run time error occurs at the execution of software then software should be able to take care of their errors.

**3. Exceptional handling:**

It should be able to handle all exceptions.

Test plan include two types of two viz. ; Black Box Testing and White Box Testing

**Black Box Testing**

1. Black Box testing is the testing of a piece of software without regards to

Its underlying implementation.

2. Specially, it dictates that test cases for a piece of software are to be

Generated based solely an examination of the specification.

3. Black Box testing involves exercising the code with nominal inputs for

Which the expected results are known.

Eg. Add new Student Details or Marks of student

4. The functionality of the module is checked in this testing. The inputs are

Provided and checked whether the module output is provided or not.

5. Black box Testing focuses on the functional requirements of the software.

6. Black Box Testing enables the software engineer to derive sets of inputs

Conditions that will fully exercise al functional requirements for a

Program.

7. Black Box testing attempts to find errors in the following categories:

Incorrect or missing functions,

Interface errors.

Errors in data structures or external database access,

Performance errors and,

Initialization and termination errors.

**White Box Testing**

1. White Box testing is the testing of the underlying implementation of a piece of software without regard to the specification for that software.

1. The goal of software testing of source code is to identify such items as

Infinite loops, paths through the code which should be allowed but which can’t be executed and dead.

1. White Box testing is one of the most important test methods. For a

Limited number of program paths, which usually suffices in practices, the

Test permits the correct manipulation of data structure and examination of the inner structure of the test object.

1. For White Box testing test case are selected on the basis of knowledge of the control flow structure of the test object.
2. The selection of test cases must consider the following:

* Every module and function of the test object must be invoked at least once.
* Every branch must be taken at least once.
* As many paths as possible must be followed.
* It is important to assure that every branch is actually taken.
* It is important to consider that, even for well structured programs, in

practice it remains impossible to test all possible paths.

**Kinds of testing**

Different Kinds of testing:

* **Unit testing :**

Unit testing focuses verification effort on the smallest unit of software component or module. Using the component level design description as a guide, important control path are tested to uncover errors within boundary of the module.

* **Integration testing**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover error associate with interfacing. The objective is to take unit tested components and build program structure that has been dictated by design.

* **Validation testing**

At the culmination of integration testing the software is completely assembled as a package ,interfacing errors have been uncovered and corrected ,and final series of software tests – validation testing – may begin. Validation succeeds when software function in a manner that can be reasonably expected by the customer.

* **System testing**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. Although each test have different purpose ,all work to verify that system elements have been verify that system elements have been properly integrated and perform allocated function.

User manual

1) This is a guide for the user of this system.

2) This contains all the relevant screen display that will help the user to understand the system.

3) There are proper uses of comments that will help the user to understand in case of error.

4) Proper validation is done and message box are prompted to the user before entering data into the database.

5) The comments describe what exactly the code block does.

There are some buttons are used in the project that are

**Submit**

1. After making all the entry if the user wants to save the data in the database then user should click on save button.
2. After clicking on save button all the records in the fields are got saved

**Cancel**

While entering the records if the user feel that he don’t want to enter the records then user should click on cancel button to move back previous position

**Delete**

1. If user feels that particular record is not useful then user should click on Delete button to delete the Record.
2. After Click on Delete Button the Current Record is no present.

**Search**

1. If the user wants to search a record by entering a particular condition then user should click on Search button.
2. After click on Search button the required records are listed in sorted format.

**Exit**

1. If the user wants to exist from the project then he should click on Logout Button.

DATA DICTIONARY

Table Name :- Stream

Description :- Holds the name of courses

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Description |
| Stream\_id | INT | Primary\_key | Stream id |
| Stream\_name | VARCHAR | - | Stream Name |

Table Name :- Setting

Description :- Holds the subject name related to courses

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data type | Constraint | Description |
| Stream | VARCHAR | Primary\_key | Stream Name |
| Sem | VARCHAR | - | Semester Index |
| Subject1 | VARCHAR | - | Subject1 Name |
| Subject2 | VARCHAR | - | Subject2 Name |
| Subject3 | VARCHAR | - | Subject3 Name |
| Subject4 | VARCHAR | - | Subject4 Name |
| Subject5 | VARCHAR | - | Subject5 Name |
| Subject6 | VARCHAR | - | Subject6 Name |

Table Name :- Message Setting

Description :- Holds Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Description |
| Message\_id | INT | Primary\_key | Message ID |
| Message\_stream | VARCHAR | - | Message Stream |
| Message\_message | VARCHAR | - | Message\_message |
|  |  |  |  |

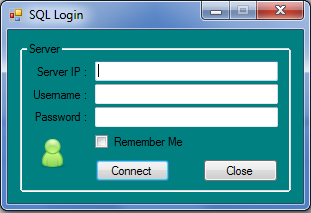
Table Name :- Course Marks

Description :- Holds the M arks related to students

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Description |
| Roll\_no | INT | Primary\_key | Student Roll No |
| Year | VARCHAR | - | Student Year |
| Sem | VARCHAR | - | Exam Sem |
| Subject1 | INT | - | Student Subject1 Marks |
| Subject2 | INT | - | Student Subject2 Marks |
| Subject3 | INT | - | Student Subject3 Marks |
| Subject4 | INT | - | Student Subject4 Marks |
| Subject5 | INT | - | Student Subject5 Marks |
| Subject6 | INT | - | Student Subject6 Marks |
| Total | INT | - | Student Total |
| Percentage | Double | - | Student Percentage |
| Grade | VARCHAR | - | Student Grade |
| Result | VARCHAR | - | Student Result |

**CODING**

SQL Login.vb



SQL Login Code

Imports MySql.Data.MySqlClient

Public Class FrmSQLLogin

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim ConString As String

Private Sub btnconnect\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnconnect.Click

ConString = "server=" & txtboxserver.Text & ";" \_

& "user id=" & txtusername.Text & ";" \_

& "password=" & txtpassword.Text & ";" \_

& "database=examtimetable"

If CheckBoxRememberMe.Checked Then

My.Settings.Hostname = txtboxserver.Text

My.Settings.Username = txtusername.Text

My.Settings.Password = txtpassword.Text

My.Settings.Save()

End If

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = ConString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

Me.Hide()

Login.GetSqlString = ConString

Login.Show()

Me.Close()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub btnclose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnclose.Click

Application.Exit()

End Sub

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

Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2

Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2

Me.AcceptButton = btnconnect

Me.CancelButton = btnclose

If Not My.Settings.Hostname = "" And Not My.Settings.Username = "" And Not My.Settings.Password = "" Then

ConString = "server=" & My.Settings.Hostname & ";" \_

& "user id=" & My.Settings.Username & ";" \_

& "password=" & My.Settings.Password & ";" \_

& "database=examtimetable"

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = ConString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

Me.Hide()

Login.GetSqlString = ConString

Login.Show()

Me.Close()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stoppedx " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

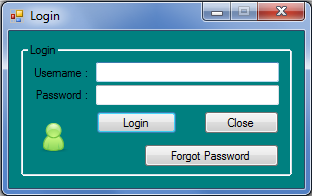
End Try

End If

End Sub

End Class

Login.vb



Login Code

Imports MySql.Data.MySqlClient

Public Class Login

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Private Sub btnconnect\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnconnect.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT user\_id FROM users WHERE username = @username AND password = @password"

myCom.Parameters.AddWithValue("@username", txtusername.Text)

myCom.Parameters.AddWithValue("@password", txtpassword.Text)

Dim exResult As Integer = myCom.ExecuteScalar

If exResult >= 1 Then

Me.Hide()

MainForm.GetSqlString = SQLString

MainForm.GetUserID = exResult

MainForm.Show()

Me.Close()

Else

MessageBox.Show("Incorrect Username/Password", "Login Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub btnclose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnclose.Click

Application.Exit()

End Sub

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

Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2

Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2

Me.AcceptButton = btnconnect

Me.CancelButton = btnclose

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim forobj As New ForgotPass

forobj.GetSqlString = SQLString

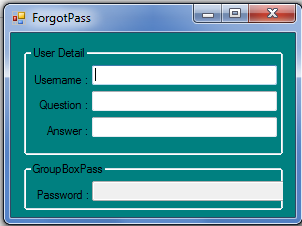
forobj.Show()

Me.Close()

End Sub

End Class

Forgot Password.vb



Forgot Password Code

Imports MySql.Data.MySqlClient

Public Class ForgotPass

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim pass As String = Nothing

Dim secans As String = Nothing

Private Sub TextBoxUname\_KeyUp(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyEventArgs) Handles TextBoxUname.KeyUp

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT secquestion,secanswer,password FROM users WHERE username = @username"

myCom.Parameters.AddWithValue("@username", TextBoxUname.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

TextBoxQuest.Text = Reader("secquestion").ToString

secans = Reader("secanswer").ToString

pass = Reader("password").ToString

End While

myCom.Parameters.Clear()

Reader.Close()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub TextBoxAns\_KeyUp(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyEventArgs) Handles TextBoxAns.KeyUp

If TextBoxAns.Text = secans Then

TextBoxRecPass.Text = pass

End If

End Sub

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

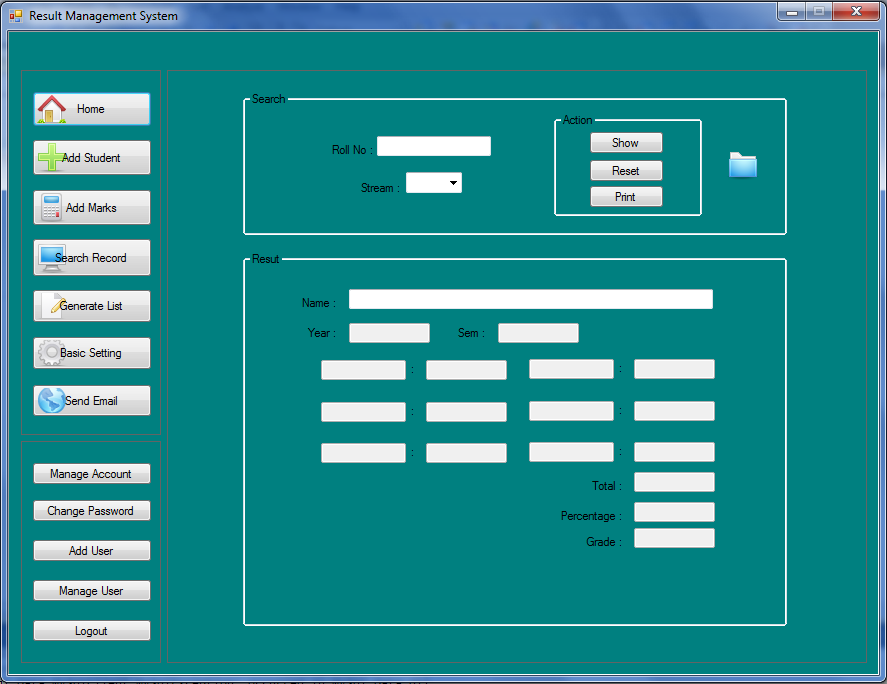
Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2

Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2

End Sub

End Class

MainPanel.vb



MainPanel Code

Imports MySql.Data.MySqlClient

Public Class MainForm

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim UserID As Integer

Public WriteOnly Property GetUserID

Set(ByVal value)

UserID = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

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

Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2

Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT rank FROM users WHERE user\_id = @user\_id"

myCom.Parameters.AddWithValue("@user\_id", UserID)

Dim Result As String = myCom.ExecuteScalar

If Result = "Admin" Then

BtnAddUser.Enabled = True

BtnUser.Enabled = True

ElseIf Result = "User" Then

BtnAddUser.Enabled = False

BtnUser.Enabled = False

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

While Reader.Read

ComboBoxStream.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub AboutToolStripMenuItem1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

About.Show()

End Sub

Private Sub AdfToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Application.Exit()

End Sub

Private Sub NotepadToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Shell("C:\WINDOWS\notepad")

End Sub

Private Sub CalculatorToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Shell("C:\WINDOWS\system32\calc")

End Sub

Private Sub btnaddStudent\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnaddMarks.Click

Load\_AddMarks()

End Sub

#Region "Function"

Public Sub Load\_AddMarks()

PanelMain.Controls.Clear()

Dim objaddmark As New AddMarks

objaddmark.Size = PanelMain.Size

objaddmark.TopLevel = False

objaddmark.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objaddmark.GetSqlString = SQLString

objaddmark.Show()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_AddStudent()

PanelMain.Controls.Clear()

Dim objaddstud As New AddStudent()

objaddstud.Size = PanelMain.Size

objaddstud.TopLevel = False

objaddstud.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objaddstud.GetSqlString = SQLString

objaddstud.Show()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_AddUser()

PanelMain.Controls.Clear()

Dim objadduser As New AddUser()

objadduser.Size = PanelMain.Size

objadduser.TopLevel = False

objadduser.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objadduser.GetSqlString = SQLString

objadduser.Show()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_ChangePass()

Dim objchangepassword As New ChangePassword()

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objchangepassword.GetSqlString = SQLString

objchangepassword.GetUserID = UserID

objchangepassword.Show()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_ManageUser()

PanelMain.Controls.Clear()

Dim objmanageuser As New ManageUser()

objmanageuser.Size = PanelMain.Size

objmanageuser.TopLevel = False

objmanageuser.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objmanageuser.GetSqlString = SQLString

objmanageuser.GetUserID = UserID

objmanageuser.Show()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_ManageAccount()

PanelMain.Controls.Clear()

Dim objmanageaccount As New ManageAccount()

objmanageaccount.Size = PanelMain.Size

objmanageaccount.TopLevel = False

objmanageaccount.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objmanageaccount.GetSqlString = SQLString

objmanageaccount.GetUserID = UserID

objmanageaccount.Show()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_SearchStudent()

PanelMain.Controls.Clear()

Dim objsearchstud As New SearchStudent()

objsearchstud.Size = PanelMain.Size

objsearchstud.TopLevel = False

objsearchstud.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objsearchstud.GetSqlString = SQLString

objsearchstud.Show()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_Home()

PanelMain.Controls.Clear()

GroupBoxSearch.Parent = PanelMain

GroupBoxSearch.Show()

GroupBoxResult.Parent = PanelMain

GroupBoxResult.Show()

Load\_comboitem()

End Sub

Public Sub Load\_GenerateList()

PanelMain.Controls.Clear()

Dim objgeneratelist As New GenerateList()

objgeneratelist.Size = PanelMain.Size

objgeneratelist.TopLevel = False

objgeneratelist.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objgeneratelist.GetSqlString = SQLString

objgeneratelist.Show()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub Load\_BasiSetting()

PanelMain.Controls.Clear()

Dim objset As New BasicSetting

objset.Size = PanelMain.Size

objset.TopLevel = False

objset.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objset.GetSqlString = SQLString

objset.Show()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

#End Region

Private Sub BtnHome\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnHome.Click

Load\_Home()

End Sub

Private Sub BtnAddUser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAddUser.Click

Load\_AddUser()

End Sub

Private Sub BtnChangePass\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnChangePass.Click

Load\_ChangePass()

End Sub

Private Sub BtnUser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnUser.Click

Load\_ManageUser()

End Sub

Private Sub BtnManageAcc\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnManageAcc.Click

Load\_ManageAccount()

End Sub

Private Sub BtnSearchRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnSearchRecord.Click

Load\_SearchStudent()

End Sub

Private Sub BtnCheckResult\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnGenerateList.Click

Load\_GenerateList()

End Sub

Private Sub MainForm\_FormClosed(ByVal sender As System.Object, ByVal e As System.Windows.Forms.FormClosedEventArgs) Handles MyBase.FormClosed

Application.Exit()

End Sub

Private Sub ButtonCheck\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonCheck.Click

ShowResult(ComboBoxStream.SelectedItem)

End Sub

Private Sub ShowResult(ByVal Streamx As String)

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT `" + ComboBoxStream.Text + "`.\* , students.\* FROM `" + ComboBoxStream.Text + "` , `students` WHERE `" + ComboBoxStream.Text + "`.roll\_no = @roll\_no AND students.s\_rollno = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", TextBoxRollNo.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

TextBoxName.Text = Reader("s\_name").ToString

TextBoxYear.Text = Reader("year").ToString

TextBoxSem.Text = Reader("sem").ToString

TextBoxSub1Result.Text = Reader("subject1").ToString

TextBoxSub2Result.Text = Reader("subject2").ToString

TextBoxSub3Result.Text = Reader("subject3").ToString

TextBoxSub4Result.Text = Reader("subject4").ToString

TextBoxSub5Result.Text = Reader("subject5").ToString

TextBoxSub6Result.Text = Reader("subject6").ToString

TextBoxTotal.Text = Reader("total").ToString

TextBoxPercentage.Text = Reader("percentage").ToString

TextBoxGrade.Text = Reader("grade").ToString

End While

Reader.Close()

If TextBoxPercentage.Text > 35 Then

LabelResult.Text = "Congratulation you are Passed"

Else

LabelResult.Text = "Sorry, you are Failed"

End If

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM settings WHERE stream = '" + ComboBoxStream.Text + "' And sem = @sem"

myCom.Parameters.AddWithValue("@sem", TextBoxSem.Text)

Reader = myCom.ExecuteReader

While Reader.Read

TextBoxSub1.Text = Reader("subject1").ToString

TextBoxSub2.Text = Reader("subject2").ToString

TextBoxSub3.Text = Reader("subject3").ToString

TextBoxSub4.Text = Reader("subject4").ToString

TextBoxSub5.Text = Reader("subject5").ToString

TextBoxSub6.Text = Reader("subject6").ToString

End While

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Catch ex As Exception

MessageBox.Show("Please Select Year First" + ex.ToString, "Error", MessageBoxButtons.OK, MessageBoxIcon.Information)

Finally

conn.Dispose()

End Try

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Catch ex As Exception

MessageBox.Show("Enter the Proper Roll No ", "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ComboBoxStream\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxStream.SelectedIndexChanged

If ComboBoxStream.SelectedItem = "IT" Then

TextBoxSub6Result.Clear()

End If

End Sub

Private Sub ButtonReset\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonReset.Click

TextBoxRollNo.Clear()

TextBoxName.Clear()

TextBoxYear.Clear()

TextBoxSem.Clear()

TextBoxSub1.Clear()

TextBoxSub2.Clear()

TextBoxSub3.Clear()

TextBoxSub4.Clear()

TextBoxSub5.Clear()

TextBoxSub6.Clear()

TextBoxSub1Result.Clear()

TextBoxSub2Result.Clear()

TextBoxSub3Result.Clear()

TextBoxSub4Result.Clear()

TextBoxSub5Result.Clear()

TextBoxSub6Result.Clear()

TextBoxTotal.Clear()

TextBoxGrade.Clear()

TextBoxPercentage.Clear()

LabelResult.Text = ""

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim objprint As New PrintResult

objprint.Show()

objprint.TextBoxName.Text = TextBoxName.Text

objprint.TextBoxYear.Text = TextBoxYear.Text

objprint.TextBoxSem.Text = TextBoxSem.Text

objprint.TextBoxSub1.Text = TextBoxSub1.Text

objprint.TextBoxSub2.Text = TextBoxSub2.Text

objprint.TextBoxSub3.Text = TextBoxSub3.Text

objprint.TextBoxSub4.Text = TextBoxSub4.Text

objprint.TextBoxSub5.Text = TextBoxSub5.Text

objprint.TextBoxSub6.Text = TextBoxSub6.Text

objprint.TextBoxSub1Result.Text = TextBoxSub1Result.Text

objprint.TextBoxSub2Result.Text = TextBoxSub2Result.Text

objprint.TextBoxSub3Result.Text = TextBoxSub3Result.Text

objprint.TextBoxSub4Result.Text = TextBoxSub4Result.Text

objprint.TextBoxSub5Result.Text = TextBoxSub5Result.Text

objprint.TextBoxSub6Result.Text = TextBoxSub6Result.Text

objprint.TextBoxTotal.Text = TextBoxTotal.Text

objprint.TextBoxPercentage.Text = TextBoxPercentage.Text

objprint.TextBoxGrade.Text = TextBoxGrade.Text

End Sub

Private Sub BtnPrintEmail\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnPrintEmail.Click

PanelMain.Controls.Clear()

Dim objaddemail As New SendResult

objaddemail.Size = PanelMain.Size

objaddemail.TopLevel = False

objaddemail.Parent = PanelMain

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

objaddemail.GetSqlString = SQLString

objaddemail.Show()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ButtonLogout\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonLogout.Click

conn = New MySqlConnection

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

Me.Hide()

Login.GetSqlString = SQLString

Login.Show()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ButtonBSetting\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonBSetting.Click

Load\_BasiSetting()

End Sub

Private Sub BtnAddStudent\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAddStudent.Click

Load\_AddStudent()

End Sub

Private Sub Load\_comboitem()

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

While Reader.Read

ComboBoxStream.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

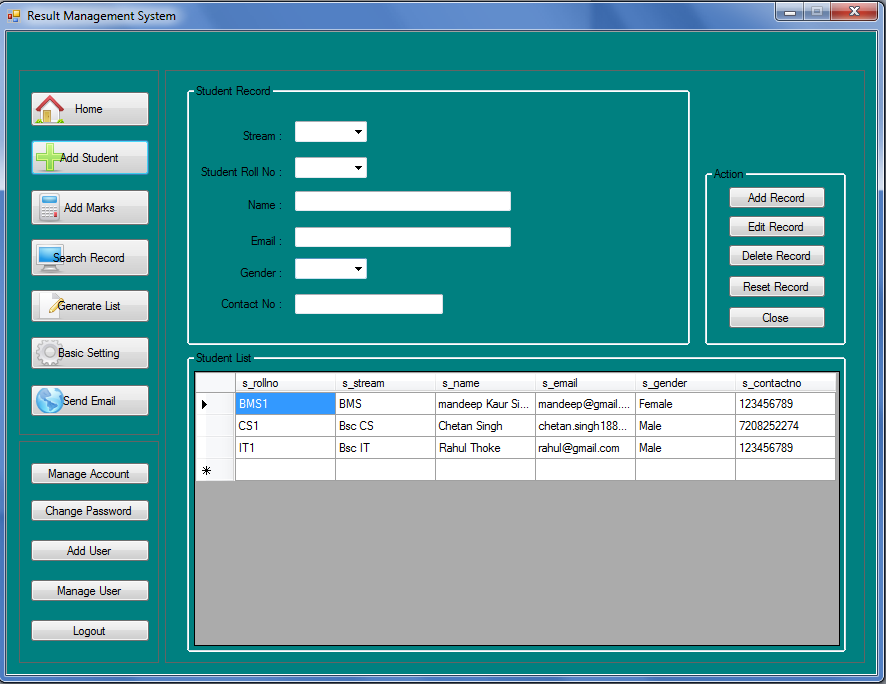
conn.Dispose()

End Try

End Sub

End Class

Add Update Student.vb



Add Update Student Code

Imports MySql.Data.MySqlClient

Public Class AddStudent

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim myAdap As MySqlDataAdapter

Dim DataTab As DataTable

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

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

While Reader.Read

ComboBoxStream.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

ComboBoxGender.Items.Add("Male")

ComboBoxGender.Items.Add("Female")

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM students"

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewStudentList.DataSource = DataTab

DataGridViewStudentList.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub BtnAddRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAddRecord.Click

Dim RollNo As String = ComboBoxRollNo.Text

Dim Stream As String = ComboBoxStream.SelectedItem.ToString

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "INSERT INTO students VALUES(@rollno,@stream,@name,@email,@gender,@contactno)"

myCom.Parameters.AddWithValue("@rollno", RollNo)

myCom.Parameters.AddWithValue("@stream", Stream)

myCom.Parameters.AddWithValue("@name", TextBoxName.Text)

myCom.Parameters.AddWithValue("@email", TextBoxEmail.Text)

myCom.Parameters.AddWithValue("@gender", ComboBoxGender.Text)

myCom.Parameters.AddWithValue("@contactno", TextBoxContactno.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Student Marks Added Successfuly")

conn.Close()

ComboBoxStream.Text = ""

ComboBoxRollNo.Text = ""

TextBoxName.Clear()

TextBoxEmail.Clear()

ComboBoxGender.Text = ""

TextBoxContactno.Clear()

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ComboBoxStream\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxStream.SelectedIndexChanged

ComboBoxRollNo.Items.Clear()

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM students WHERE s\_stream = @stream"

myCom.Parameters.AddWithValue("@stream", ComboBoxStream.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

ComboBoxRollNo.Items.Add(Reader("s\_rollno").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM students WHERE s\_stream = @stream"

myCom.Parameters.AddWithValue("@stream", ComboBoxStream.Text)

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewStudentList.DataSource = DataTab

DataGridViewStudentList.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ComboBoxRollNo\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxRollNo.SelectedIndexChanged

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM students WHERE s\_rollno = @rollno"

myCom.Parameters.AddWithValue("@rollno", ComboBoxRollNo.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

TextBoxName.Text = Reader("s\_name").ToString

TextBoxEmail.Text = Reader("s\_email").ToString

ComboBoxGender.Text = Reader("s\_gender").ToString

TextBoxContactno.Text = Reader("s\_contactno").ToString

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub BtnEditRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnEditRecord.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "UPDATE students SET s\_name = @name WHERE s\_rollno = @roll\_no; " \_

& "UPDATE students SET s\_email = @email WHERE s\_rollno = @roll\_no; " \_

& "UPDATE students SET s\_gender = @gender WHERE s\_rollno = @roll\_no; " \_

& "UPDATE students SET s\_contactno = @contactno WHERE s\_rollno = @roll\_no;"

myCom.Parameters.AddWithValue("@roll\_no", ComboBoxRollNo.Text)

myCom.Parameters.AddWithValue("@name", TextBoxName.Text)

myCom.Parameters.AddWithValue("@email", TextBoxEmail.Text)

myCom.Parameters.AddWithValue("@gender", ComboBoxGender.Text)

myCom.Parameters.AddWithValue("@contactno", TextBoxContactno.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Data Updated Successfully", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

ShowUserList(ComboBoxStream.Text)

myCom.Parameters.Clear()

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ShowUserList(ByVal table As String)

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM students WHERE s\_stream = @stream"

myCom.Parameters.AddWithValue("@stream", table)

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewStudentList.DataSource = DataTab

DataGridViewStudentList.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub BtnDeleteRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnDeleteRecord.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "DELETE FROM students WHERE roll\_no = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", ComboBoxRollNo.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Deleted Record Successfuly", "Deleted", MessageBoxButtons.OK, MessageBoxIcon.Error)

ShowUserList("CS")

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ButtonReset\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonReset.Click

ComboBoxStream.Text = ""

ComboBoxRollNo.Text = ""

TextBoxName.Clear()

TextBoxEmail.Clear()

ComboBoxGender.Text = ""

TextBoxContactno.Clear()

End Sub

Private Sub BtnClose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnClose.Click

MainForm.PanelMain.Controls.Clear()

MainForm.GroupBoxSearch.Parent = MainForm.PanelMain

MainForm.GroupBoxSearch.Show()

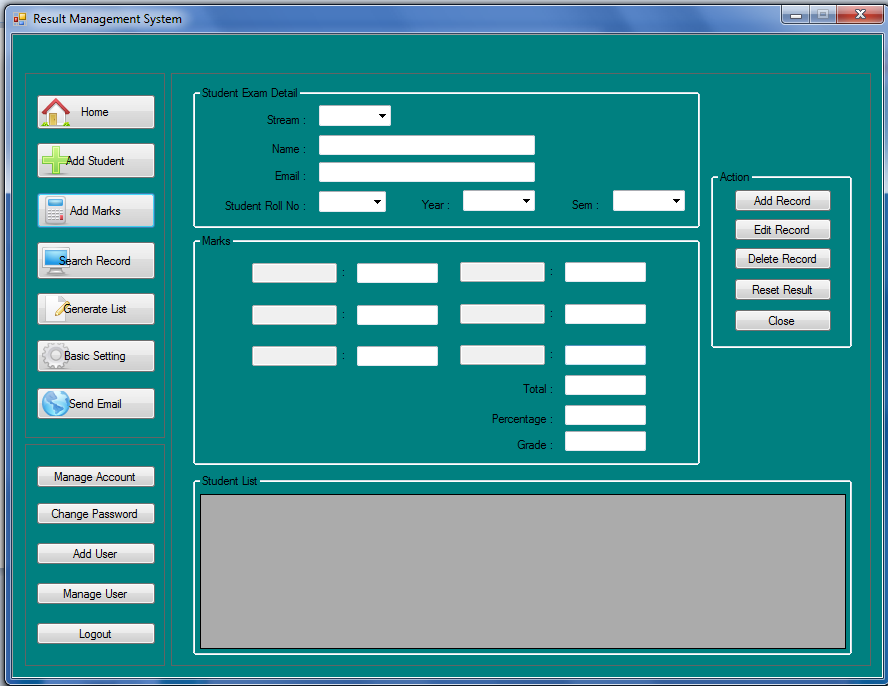
MainForm.GroupBoxResult.Parent = MainForm.PanelMain

MainForm.GroupBoxResult.Show()

End Sub

End Class

Add Marks.vb



Add Marks Code

Imports MySql.Data.MySqlClient

Public Class AddMarks

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim myAdap As MySqlDataAdapter

Dim DataTab As DataTable

'Normal Variable

Public Stream As String = Nothing

Public Year As String = Nothing

Public Sem As String = Nothing

'Marks Calculation

Public SubM1 As Integer() = {0, 100}

Public SubM2 As Integer() = {0, 100}

Public SubM3 As Integer() = {0, 100}

Public SubM4 As Integer() = {0, 100}

Public SubM5 As Integer() = {0, 100}

Public SubM6 As Integer() = {0, 100}

Dim Sum As Integer = Nothing

Dim Percentage As Double = Nothing

'Grade

Public Grade As String() = {"Distinction", "A", "B", "C", "Pass", "Fail"}

Public Result As String = Nothing

Private Sub Add\_Student\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

While Reader.Read

ComboBoxStream.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

ComboBoxYear.Items.Add("First")

ComboBoxYear.Items.Add("Second")

ComboBoxYear.Items.Add("Third")

ComboBoxSem.Items.Add("Sem-1")

ComboBoxSem.Items.Add("Sem-2")

ComboBoxSem.Items.Add("Sem-3")

ComboBoxSem.Items.Add("Sem-4")

ComboBoxSem.Items.Add("Sem-5")

ComboBoxSem.Items.Add("Sem-6")

End Sub

Private Sub ComboBoxStream\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxStream.SelectedIndexChanged

Stream = ComboBoxStream.Text

TextBoxSub1Result.Clear()

TextBoxSub2Result.Clear()

TextBoxSub3Result.Clear()

TextBoxSub4Result.Clear()

TextBoxSub5Result.Clear()

TextBoxSub6Result.Clear()

TextBoxPercentage.Clear()

TextBoxGrade.Clear()

TextBoxTotal.Clear()

If Stream.Length > 0 Then

If Year = "First" Then

ComboBoxSem.Items.Clear()

TextBoxSub6Result.Enabled = True

ComboBoxSem.Items.Add("Sem-1")

ComboBoxSem.Items.Add("Sem-2")

ComboBoxSem.SelectedIndex = 0

ElseIf Year = "Second" Then

ComboBoxSem.Items.Clear()

TextBoxSub6Result.Enabled = True

ComboBoxSem.Items.Add("Sem-3")

ComboBoxSem.Items.Add("Sem-4")

ComboBoxSem.SelectedIndex = 0

ElseIf Year = "Third" Then

ComboBoxSem.Items.Clear()

TextBoxSub6Result.Enabled = True

ComboBoxSem.Items.Add("Sem-5")

ComboBoxSem.Items.Add("Sem-6")

ComboBoxSem.SelectedIndex = 0

End If

End If

'If Stream = "CS" Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM students WHERE s\_stream = @stream"

myCom.Parameters.AddWithValue("@stream", ComboBoxStream.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxRollNo.Items.Clear()

While Reader.Read

ComboBoxRollNo.Items.Add(Reader("s\_rollno").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

ShowUserList(ComboBoxStream.Text)

End Sub

Private Sub ComboBoxYear\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxYear.SelectedIndexChanged

Year = ComboBoxYear.Text

Sem = ComboBoxSem.Text

Stream = ComboBoxStream.Text

If Stream.Length > 0 Then

If Year = "First" Then

ComboBoxSem.Items.Clear()

ComboBoxSem.Items.Add("Sem-1")

ComboBoxSem.Items.Add("Sem-2")

ElseIf Year = "Second" Then

ComboBoxSem.Items.Clear()

ComboBoxSem.Items.Add("Sem-3")

ComboBoxSem.Items.Add("Sem-4")

ElseIf Year = "Third" Then

ComboBoxSem.Items.Clear()

ComboBoxSem.Items.Add("Sem-5")

ComboBoxSem.Items.Add("Sem-6")

End If

End If

End Sub

Private Sub BtnClose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnClose.Click

MainForm.PanelMain.Controls.Clear()

MainForm.GroupBoxSearch.Parent = MainForm.PanelMain

MainForm.GroupBoxSearch.Show()

MainForm.GroupBoxResult.Parent = MainForm.PanelMain

MainForm.GroupBoxResult.Show()

End Sub

Private Sub ComboBoxSem\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxSem.SelectedIndexChanged

Sem = ComboBoxSem.Text

Year = ComboBoxYear.Text

Stream = ComboBoxStream.Text

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM settings WHERE stream = '" + ComboBoxStream.Text + "' And sem = @sem"

myCom.Parameters.AddWithValue("@sem", ComboBoxSem.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

TextBoxSub1.Text = Reader("subject1").ToString

TextBoxSub2.Text = Reader("subject2").ToString

TextBoxSub3.Text = Reader("subject3").ToString

TextBoxSub4.Text = Reader("subject4").ToString

TextBoxSub5.Text = Reader("subject5").ToString

TextBoxSub6.Text = Reader("subject6").ToString

End While

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Catch ex As Exception

MessageBox.Show("Please Select Year First" + ex.ToString, "Error", MessageBoxButtons.OK, MessageBoxIcon.Information)

Finally

conn.Dispose()

End Try

End Sub

Private Sub TextBoxTotal\_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBoxTotal.Enter

If TextBoxTotal.Text.Length = 0 Then

SubM1(0) = CType(TextBoxSub1Result.Text, Integer)

SubM2(0) = CType(TextBoxSub2Result.Text, Integer)

SubM3(0) = CType(TextBoxSub3Result.Text, Integer)

SubM4(0) = CType(TextBoxSub4Result.Text, Integer)

SubM5(0) = CType(TextBoxSub5Result.Text, Integer)

If TextBoxSub6Result.Text = "" Then

SubM6(0) = 0

Else

SubM6(0) = CType(TextBoxSub6Result.Text, Integer)

End If

Sum = SubM1(0) + SubM2(0) + SubM3(0) + SubM4(0) + SubM5(0) + SubM6(0)

TextBoxTotal.Text = Sum.ToString

'ElseIf Stream = "IT" And TextBoxTotal.Text.Length = 0 Then

' SubM1(0) = CType(TextBoxSub1Result.Text, Integer)

' SubM2(0) = CType(TextBoxSub2Result.Text, Integer)

' SubM3(0) = CType(TextBoxSub3Result.Text, Integer)

' SubM4(0) = CType(TextBoxSub4Result.Text, Integer)

' SubM5(0) = CType(TextBoxSub5Result.Text, Integer)

' Sum = SubM1(0) + SubM2(0) + SubM3(0) + SubM4(0) + SubM5(0)

' TextBoxTotal.Text = Sum.ToString

End If

End Sub

Private Sub TextBoxPercentage\_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBoxPercentage.Enter

If TextBoxPercentage.Text.Length = 0 And TextBoxTotal.Text.Length <> 0 Then

Dim resultf As Integer

If TextBoxSub6Result.Text = "" Then

resultf = 0

Else

resultf = SubM6(1)

End If

Percentage = (Sum \* 100) / (SubM1(1) + SubM2(1) + SubM3(1) + SubM4(1) + SubM5(1) + resultf)

TextBoxPercentage.Text = Percentage.ToString

'ElseIf Stream = "IT" And TextBoxPercentage.Text.Length = 0 And TextBoxTotal.Text.Length <> 0 Then

' Percentage = (Sum \* 100) / (SubM1(1) + SubM2(1) + SubM3(1) + SubM4(1) + SubM5(1))

' TextBoxPercentage.Text = Percentage.ToString

End If

End Sub

Private Sub TextBoxGrade\_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBoxGrade.Enter

Dim Result6Marks As Integer

If TextBoxSub6Result.Text = "" Then

Result6Marks = 35

Else

Result6Marks = TextBoxSub6Result.Text

End If

If TextBoxGrade.Text.Length = 0 And TextBoxTotal.Text.Length <> 0 And TextBoxPercentage.Text.Length <> 0 Then

If TextBoxSub1Result.Text >= 35 And TextBoxSub2Result.Text >= 35 And TextBoxSub3Result.Text >= 35 And TextBoxSub4Result.Text >= 35 And TextBoxSub5Result.Text >= 35 Then

If Percentage >= 75 Then

TextBoxGrade.Text = Grade(0)

Result = "Pass"

ElseIf Percentage >= 60 And Percentage < 75 Then

TextBoxGrade.Text = Grade(1)

Result = "Pass"

ElseIf Percentage >= 50 And Percentage < 60 Then

TextBoxGrade.Text = Grade(2)

Result = "Pass"

ElseIf Percentage >= 42 And Percentage < 50 Then

TextBoxGrade.Text = Grade(3)

Result = "Pass"

ElseIf Percentage >= 35 And Percentage < 42 Then

TextBoxGrade.Text = Grade(4)

Result = "Pass"

Else

TextBoxGrade.Text = Grade(5)

Result = "Fail"

End If

Else

TextBoxGrade.Text = Grade(5)

Result = "Fail"

End If

End If

End Sub

Private Sub ButtonReset\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonReset.Click

TextBoxTotal.Clear()

TextBoxGrade.Clear()

TextBoxPercentage.Clear()

End Sub

Private Sub BtnAddRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAddRecord.Click

'If Stream = "CS" Then

Dim RollNo As String = ComboBoxRollNo.Text

Dim Stream As String = ComboBoxStream.SelectedItem

Dim Year As String = ComboBoxYear.SelectedItem

Dim Sem As String = ComboBoxSem.SelectedItem

If TextBoxName.Text.Length <> 0 And Stream.Length <> 0 And Year.Length <> 0 And Sem.Length <> 0 Then

If TextBoxSub1Result.Text.Length <> 0 And TextBoxSub2Result.Text.Length <> 0 And TextBoxSub3Result.Text.Length <> 0 And TextBoxSub4Result.Text.Length <> 0 And TextBoxSub5Result.Text.Length <> 0 Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "INSERT INTO `" + ComboBoxStream.Text.ToLower + "` VALUES(@rollno,@year,@sem,@sub1,@sub2,@sub3,@sub4,@sub5,@sub6" \_

& ",@total,@percentage,@grade,@result)"

myCom.Parameters.AddWithValue("@rollno", ComboBoxRollNo.Text)

myCom.Parameters.AddWithValue("@year", ComboBoxYear.SelectedItem)

myCom.Parameters.AddWithValue("@sem", ComboBoxSem.SelectedItem)

myCom.Parameters.AddWithValue("@sub1", TextBoxSub1Result.Text)

myCom.Parameters.AddWithValue("@sub2", TextBoxSub2Result.Text)

myCom.Parameters.AddWithValue("@sub3", TextBoxSub3Result.Text)

myCom.Parameters.AddWithValue("@sub4", TextBoxSub4Result.Text)

myCom.Parameters.AddWithValue("@sub5", TextBoxSub5Result.Text)

If TextBoxSub6.Text = "" Then

myCom.Parameters.AddWithValue("@sub6", 0)

Else

myCom.Parameters.AddWithValue("@sub6", TextBoxSub6Result.Text)

End If

myCom.Parameters.AddWithValue("@total", TextBoxTotal.Text)

myCom.Parameters.AddWithValue("@percentage", TextBoxPercentage.Text)

myCom.Parameters.AddWithValue("@grade", TextBoxGrade.Text)

myCom.Parameters.AddWithValue("@result", Result)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Student Marks Added Successfuly")

conn.Close()

TextBoxName.Clear()

TextBoxSub1Result.Clear()

TextBoxSub2Result.Clear()

TextBoxSub3Result.Clear()

TextBoxSub4Result.Clear()

TextBoxSub5Result.Clear()

TextBoxSub6Result.Clear()

TextBoxPercentage.Clear()

TextBoxGrade.Clear()

TextBoxTotal.Clear()

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

Else

MessageBox.Show("Please fill up all the fieldx", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

End If

Else

MessageBox.Show("Please fill up all the fields", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

End If

ShowUserList(ComboBoxStream.Text)

End Sub

Private Sub ShowUserList(ByVal x As String)

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM `" + x + "`"

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewStudentList.DataSource = DataTab

DataGridViewStudentList.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ComboBoxRollNo\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxRollNo.SelectedIndexChanged

Dim Stream As String = ComboBoxStream.Text

conn = New MySqlConnection

myCom = New MySqlCommand

Dim Reader As MySqlDataReader

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Parameters.Clear()

myCom.Connection = conn

myCom.CommandText = "SELECT count(\*) FROM `" + Stream.ToLower + "`"

Dim count As Integer = myCom.ExecuteScalar

If count = 0 Then

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM students WHERE s\_rollno = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", ComboBoxRollNo.Text)

Else

myCom.Parameters.Clear()

myCom.CommandText = "SELECT `" + ComboBoxStream.Text + "`.\* , students.\* FROM `" + ComboBoxStream.Text + "` , `students` WHERE `" + ComboBoxStream.Text + "`.roll\_no = @roll\_no AND students.s\_rollno = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", ComboBoxRollNo.Text)

End If

Reader = myCom.ExecuteReader

While Reader.Read

If count = 0 Then

TextBoxName.Text = Reader("s\_name").ToString

TextBoxEmail.Text = Reader("s\_email").ToString

ElseIf count > 0 Then

TextBoxName.Text = Reader("s\_name").ToString

TextBoxEmail.Text = Reader("s\_email").ToString

ComboBoxYear.Text = Reader("year").ToString

ComboBoxSem.Text = Reader("sem").ToString

TextBoxSub1Result.Text = Reader("subject1").ToString

TextBoxSub2Result.Text = Reader("subject2").ToString

TextBoxSub3Result.Text = Reader("subject3").ToString

TextBoxSub4Result.Text = Reader("subject4").ToString

TextBoxSub5Result.Text = Reader("subject5").ToString

TextBoxSub6Result.Text = Reader("subject6").ToString

TextBoxTotal.Text = Reader("total").ToString

TextBoxPercentage.Text = Reader("percentage").ToString

TextBoxGrade.Text = Reader("grade").ToString

End If

End While

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub BtnEditRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnEditRecord.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "UPDATE students SET s\_name = @name WHERE s\_rollno = @roll\_no; " \_

& "UPDATE students SET s\_email = @email WHERE s\_rollno = @roll\_no; " \_

& "UPDATE students SET s\_stream = @stream WHERE s\_rollno = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET year = @year WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET sem = @sem WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET subject1 = @subject1 WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET subject2 = @subject2 WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET subject3 = @subject3 WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET subject4 = @subject4 WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET subject5 = @subject5 WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET subject6 = @subject6 WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET total = @total WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET percentage = @percentage WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET grade = @grade WHERE roll\_no = @roll\_no; " \_

& "UPDATE `" + ComboBoxStream.Text.ToLower + "` SET result = @result WHERE roll\_no = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", ComboBoxRollNo.Text)

myCom.Parameters.AddWithValue("@name", TextBoxName.Text)

myCom.Parameters.AddWithValue("@email", TextBoxEmail.Text)

myCom.Parameters.AddWithValue("@stream", ComboBoxStream.Text)

myCom.Parameters.AddWithValue("@year", ComboBoxYear.Text)

myCom.Parameters.AddWithValue("@sem", ComboBoxSem.Text)

myCom.Parameters.AddWithValue("@subject1", TextBoxSub1Result.Text)

myCom.Parameters.AddWithValue("@subject2", TextBoxSub2Result.Text)

myCom.Parameters.AddWithValue("@subject3", TextBoxSub3Result.Text)

myCom.Parameters.AddWithValue("@subject4", TextBoxSub4Result.Text)

myCom.Parameters.AddWithValue("@subject5", TextBoxSub5Result.Text)

myCom.Parameters.AddWithValue("@subject6", TextBoxSub6Result.Text)

myCom.Parameters.AddWithValue("@total", TextBoxTotal.Text)

myCom.Parameters.AddWithValue("@percentage", TextBoxPercentage.Text)

myCom.Parameters.AddWithValue("@grade", TextBoxGrade.Text)

myCom.Parameters.AddWithValue("@result", TextBoxGrade.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Data Updated Successfully", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

ShowUserList(ComboBoxStream.Text)

myCom.Parameters.Clear()

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub BtnDeleteRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnDeleteRecord.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "DELETE FROM `" + ComboBoxStream.Text + "` WHERE roll\_no = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", ComboBoxRollNo.SelectedItem)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Deleted Record Successfuly", "Deleted", MessageBoxButtons.OK, MessageBoxIcon.Error)

ShowUserList(ComboBoxStream.Text)

TextBoxSub1Result.Clear()

TextBoxSub2Result.Clear()

TextBoxSub3Result.Clear()

TextBoxSub4Result.Clear()

TextBoxSub5Result.Clear()

TextBoxSub6Result.Clear()

TextBoxTotal.Clear()

TextBoxGrade.Clear()

TextBoxPercentage.Clear()

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

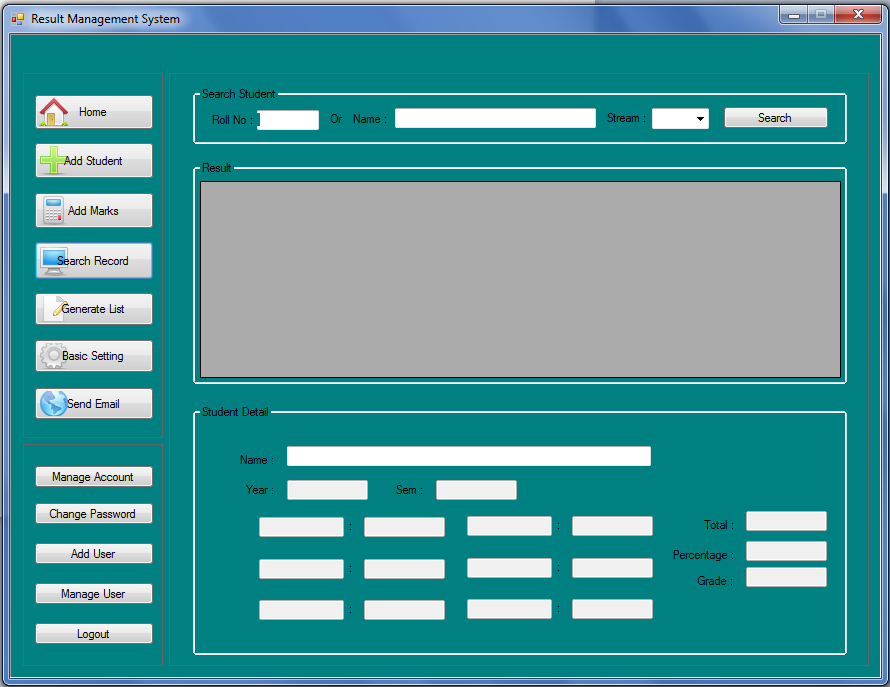
conn.Dispose()

End Try

End Sub

End Class

Search Records.vb



Search Record Code

Imports MySql.Data.MySqlClient

Public Class SearchStudent

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim Streamo As String = Nothing

Dim roll\_noo As String = Nothing

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim myAdap As MySqlDataAdapter

Dim DataTab As DataTable

Private Sub Btnsearch\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnsearch.Click

conn = New MySqlConnection

myCom = New MySqlCommand

DataTab = New DataTable

myAdap = New MySqlDataAdapter

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

If Txtboxrollno.Text.Length <> 0 And Not ComboBoxStream.Text = "" Then

ShowResultRoll()

ElseIf Txtboxname.Text.Length <> 0 And Not ComboBoxStream.SelectedItem Is Nothing Then

ShowResultName()

Else

MessageBox.Show("Please check that Stream is selected and all the fields are filled")

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub showResult()

myCom.CommandText = "SELECT \* FROM `" + ComboBoxStream.Text + "` WHERE rollno = '" + Txtboxrollno.Text + "'"

End Sub

Private Sub Txtboxrollno\_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Txtboxrollno.Enter

Txtboxname.Clear()

End Sub

Private Sub Txtboxname\_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Txtboxname.Enter

Txtboxrollno.Clear()

End Sub

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

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

While Reader.Read

ComboBoxStream.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ShowResultRoll()

myCom.CommandText = "SELECT \* FROM `" + ComboBoxStream.Text + "` WHERE roll\_no = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", Txtboxrollno.Text)

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewShowResult.DataSource = DataTab

End Sub

Private Sub ShowResultName()

myCom.CommandText = "SELECT \* FROM `" + ComboBoxStream.Text.ToLower + "` JOIN students WHERE students.s\_name LIKE @name AND `" + ComboBoxStream.Text + "`.roll\_no = students.s\_rollno"

myCom.Parameters.AddWithValue("@name", "%" & Txtboxname.Text & "%")

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewShowResult.DataSource = DataTab

End Sub

Private Sub DataGridViewShowResult\_CellClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles DataGridViewShowResult.CellClick

MsgBox(DataGridViewShowResult.Rows(e.RowIndex).Cells(0).Value)

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT `" + ComboBoxStream.Text + "`.\* , students.\* FROM `" + ComboBoxStream.Text + "` , `students` WHERE `" + ComboBoxStream.Text + "`.roll\_no = @roll\_no AND students.s\_rollno = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", DataGridViewShowResult.Rows(e.RowIndex).Cells(0).Value)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

TextBoxName.Text = Reader("s\_name").ToString

TextBoxYear.Text = Reader("year").ToString

TextBoxSem.Text = Reader("sem").ToString

TextBoxSub1Result.Text = Reader("subject1").ToString

TextBoxSub2Result.Text = Reader("subject2").ToString

TextBoxSub3Result.Text = Reader("subject3").ToString

TextBoxSub4Result.Text = Reader("subject4").ToString

TextBoxSub5Result.Text = Reader("subject5").ToString

TextBoxSub6Result.Text = Reader("subject6").ToString

TextBoxTotal.Text = Reader("total").ToString

TextBoxPercentage.Text = Reader("percentage").ToString

TextBoxGrade.Text = Reader("grade").ToString

End While

Reader.Close()

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT \* FROM settings WHERE stream = '" + ComboBoxStream.Text + "' And sem = @sem"

myCom.Parameters.AddWithValue("@sem", TextBoxSem.Text)

Reader = myCom.ExecuteReader

While Reader.Read

TextBoxSub1.Text = Reader("subject1").ToString

TextBoxSub2.Text = Reader("subject2").ToString

TextBoxSub3.Text = Reader("subject3").ToString

TextBoxSub4.Text = Reader("subject4").ToString

TextBoxSub5.Text = Reader("subject5").ToString

TextBoxSub6.Text = Reader("subject6").ToString

End While

myCom.Parameters.Clear()

conn.Close()

Catch ex As MySqlException

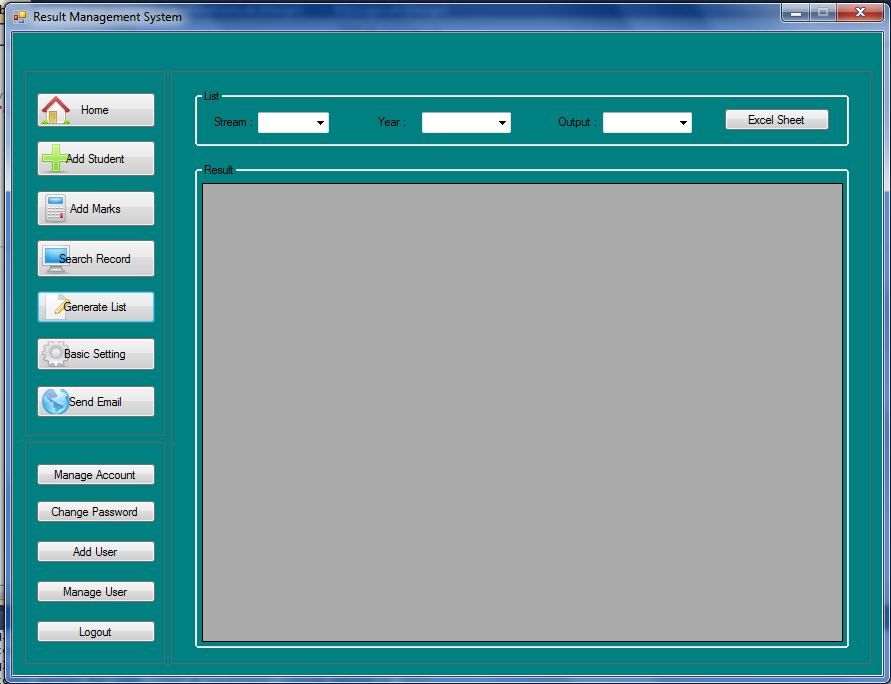
MessageBox.Show("Connection to Database is stopped " + ex.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

End Class

Generate List.vb



Generate List Code

Imports System.Data

Imports Excel = Microsoft.Office.Interop.Excel

Imports MySql.Data.MySqlClient

Public Class GenerateList

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim myAdap As MySqlDataAdapter

Dim DataSet As DataSet

Dim DataTab As DataTable

Dim i, j As Integer

Dim xlApp As Excel.Application

Dim xlWorkBook As Excel.Workbook

Dim xlWorkSheet As Excel.Worksheet

Dim misValue As Object = System.Reflection.Missing.Value

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

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxstream.Items.Clear()

While Reader.Read

ComboBoxstream.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

ComboBoxyear.Items.Add("First")

ComboBoxyear.Items.Add("Second")

ComboBoxyear.Items.Add("Third")

ComboBoxOutput.Items.Add("All")

ComboBoxOutput.Items.Add("Passed")

ComboBoxOutput.Items.Add("Failed")

End Sub

Private Sub ComboBoxstream\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxstream.SelectedIndexChanged

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

If ComboBoxstream.SelectedItem.ToString.Length <> 0 And ComboBoxyear.SelectedItem Is Nothing And ComboBoxOutput.SelectedItem Is Nothing Then

ShowResult(ComboBoxstream.Text)

conn.Close()

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ComboBoxyear\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxyear.SelectedIndexChanged

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

Try

If ComboBoxstream.SelectedItem.ToString.Length <> 0 And ComboBoxyear.SelectedItem.ToString.Length <> 0 And ComboBoxOutput.SelectedItem.ToString.Length <> 0 Then

ShowYearResult(ComboBoxyear.Text, ComboBoxstream.Text)

conn.Close()

End If

Catch ex As Exception

MessageBox.Show("Please Select Proper Filed", "Error", MessageBoxButtons.OK, MessageBoxIcon.Information)

End Try

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ComboBoxOutput\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxOutput.SelectedIndexChanged

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

Try

If ComboBoxstream.SelectedItem.ToString.Length <> 0 And ComboBoxyear.SelectedItem.ToString.Length <> 0 And ComboBoxOutput.SelectedItem.ToString.Length <> 0 Then

ShowYearResultOut(ComboBoxyear.Text, ComboBoxstream.Text, ComboBoxOutput.Text)

conn.Close()

End If

Catch ex As Exception

MessageBox.Show("Please Select Proper Filed", "Error", MessageBoxButtons.OK, MessageBoxIcon.Information)

End Try

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub ShowResult(ByVal ShowStream As String)

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM `" + ComboBoxstream.Text + "`"

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewResult.DataSource = DataTab

DataGridViewResult.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

End Sub

Public Sub ShowYearResult(ByVal ShowYear As String, ByVal Streamx As String)

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM `" + ComboBoxstream.Text + "` WHERE year = @year"

myCom.Parameters.AddWithValue("@year", ComboBoxyear.SelectedItem)

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewResult.DataSource = DataTab

DataGridViewResult.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

End Sub

Public Sub ShowYearResultOut(ByVal ShowYear As String, ByVal Streamx As String, ByVal Outx As String)

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM `" + ComboBoxstream.Text + "` WHERE year = @year And result = @result"

myCom.Parameters.AddWithValue("@year", ComboBoxyear.SelectedItem)

myCom.Parameters.AddWithValue("@result", ComboBoxOutput.Text)

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewResult.DataSource = DataTab

DataGridViewResult.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

End Sub

Private Sub Btnsearch\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnsearch.Click

With SaveFileDialogExcel

.Filter = "Excel File (\*.xlsx) | \*.xlsx | All files (\*.\*)) | \*.\*"

.FilterIndex = 1

.InitialDirectory = "C:\\"

.Title = "Save as Excel File"

End With

xlApp = New Excel.Application

xlWorkBook = xlApp.Workbooks.Add(misValue)

xlWorkSheet = xlWorkBook.Sheets("sheet1")

conn = New MySqlConnection

DataSet = New DataSet

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

Try

If ComboBoxstream.SelectedItem Is Nothing And ComboBoxyear.SelectedItem Is Nothing And ComboBoxOutput Is Nothing Then

MessageBox.Show("Please Select the fields")

ElseIf Not ComboBoxstream.SelectedItem.ToString = "" And Not ComboBoxyear.Text <> "" And Not ComboBoxOutput.Text <> "" Then

SaveDialog1(ComboBoxstream.Text)

ElseIf Not ComboBoxstream.SelectedItem.ToString = "" And Not ComboBoxyear.SelectedItem.ToString = "" And ComboBoxOutput.SelectedItem Is Nothing Then

SaveDialog2(ComboBoxstream.Text, ComboBoxyear.Text)

ElseIf Not ComboBoxstream.SelectedItem.ToString = "CS" And Not ComboBoxyear.SelectedItem.ToString = "" And Not ComboBoxOutput.SelectedItem = "" Then

SaveDialog3(ComboBoxstream.Text, ComboBoxyear.Text, ComboBoxOutput.Text)

End If

Catch ex As Exception

MessageBox.Show("Please Select Proper Filed", "Error", MessageBoxButtons.OK, MessageBoxIcon.Information)

End Try

If myCom.CommandText.Length <> 0 Then

myAdap.SelectCommand = myCom

myAdap.Fill(DataSet)

For Me.i = 0 To DataSet.Tables(0).Rows.Count - 1

For Me.j = 0 To DataSet.Tables(0).Columns.Count - 1

xlWorkSheet.Cells(i + 1, j + 1) = \_

DataSet.Tables(0).Rows(i).Item(j)

Next

Next

If SaveFileDialogExcel.ShowDialog() = DialogResult.OK Then

xlWorkSheet.SaveAs(SaveFileDialogExcel.FileName.ToString)

End If

xlWorkBook.Close()

xlApp.Quit()

releaseObject(xlApp)

releaseObject(xlWorkBook)

releaseObject(xlWorkSheet)

conn.Close()

MsgBox("You can find the file " + SaveFileDialogExcel.FileName.ToString)

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub SaveDialog1(ByVal Streams As String)

myCom.CommandText = "SELECT \* FROM `" + ComboBoxstream.Text + "`"

End Sub

Public Sub SaveDialog2(ByVal Streams As String, ByVal Years As String)

myCom.CommandText = "SELECT \* FROM `" + ComboBoxstream.Text + "` WHERE year = @year"

myCom.Parameters.AddWithValue("@year", ComboBoxyear.Text)

End Sub

Public Sub SaveDialog3(ByVal Streams As String, ByVal Years As String, ByVal Outputs As String)

myCom.CommandText = "SELECT \* FROM `" + ComboBoxstream.Text + "` WHERE year = @year"

myCom.Parameters.AddWithValue("@year", ComboBoxyear.Text)

myCom.Parameters.AddWithValue("@result", ComboBoxOutput.Text)

End Sub

Private Sub releaseObject(ByVal obj As Object)

Try

System.Runtime.InteropServices.Marshal.ReleaseComObject(obj)

obj = Nothing

Catch ex As Exception

obj = Nothing

Finally

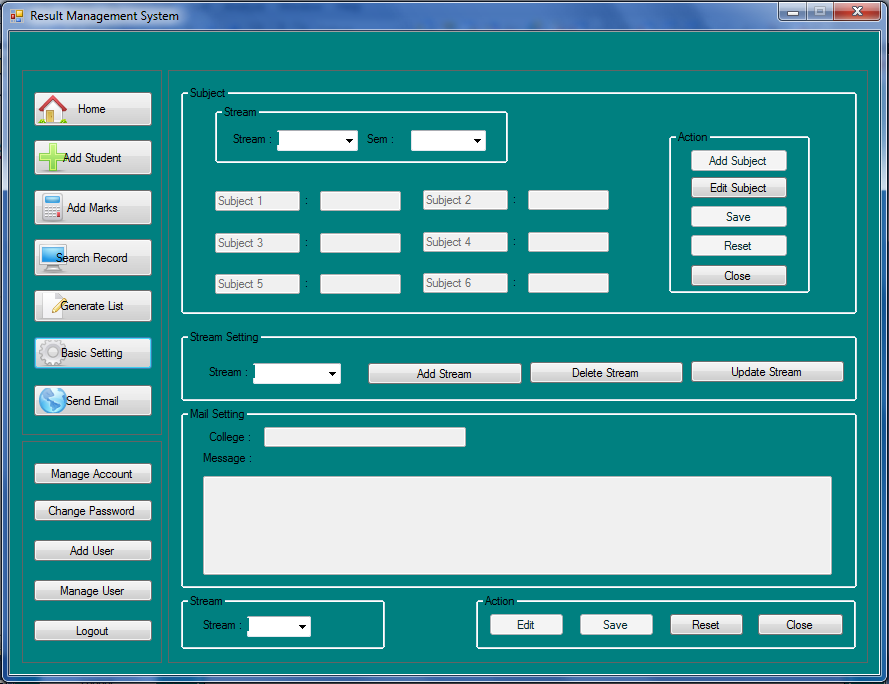
GC.Collect()

End Try

End Sub

End Class

BasicSetting.vb



Basic Setting Code

Imports MySql.Data.MySqlClient

Public Class BasicSetting

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim myAdap As MySqlDataAdapter

Dim DataTab As DataTable

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

Load\_comboitem()

TextBoxSub1.Text = "Subject 1"

TextBoxSub2.Text = "Subject 2"

TextBoxSub3.Text = "Subject 3"

TextBoxSub4.Text = "Subject 4"

TextBoxSub5.Text = "Subject 5"

TextBoxSub6.Text = "Subject 6"

TextBoxSub1Result.Enabled = False

TextBoxSub2Result.Enabled = False

TextBoxSub3Result.Enabled = False

TextBoxSub4Result.Enabled = False

TextBoxSub5Result.Enabled = False

TextBoxSub6Result.Enabled = False

BtnSave.Enabled = False

ButtonReset.Enabled = False

BtnEdit.Enabled = False

BtnMsgEdit.Enabled = False

BtnEdit.Enabled = False

Button2.Enabled = False

TextBoxCollege.Enabled = False

End Sub

Private Sub ComboBoxStream\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxStream.SelectedIndexChanged

ComboBoxSem.Items.Clear()

ComboBoxSem.Items.Add("Sem-1")

ComboBoxSem.Items.Add("Sem-2")

ComboBoxSem.Items.Add("Sem-3")

ComboBoxSem.Items.Add("Sem-4")

ComboBoxSem.Items.Add("Sem-5")

ComboBoxSem.Items.Add("Sem-6")

End Sub

Private Sub ComboBoxSem\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxSem.SelectedIndexChanged

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM settings WHERE stream = @stream And sem = @sem"

myCom.Parameters.Clear()

myCom.Parameters.AddWithValue("@stream", ComboBoxStream.Text)

myCom.Parameters.AddWithValue("@sem", ComboBoxSem.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

TextBoxSub1Result.Text = Reader("subject1").ToString

TextBoxSub2Result.Text = Reader("subject2").ToString

TextBoxSub3Result.Text = Reader("subject3").ToString

TextBoxSub4Result.Text = Reader("subject4").ToString

TextBoxSub5Result.Text = Reader("subject5").ToString

TextBoxSub6Result.Text = Reader("subject6").ToString

End While

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

If TextBoxSub1Result.Text = "" And TextBoxSub2Result.Text = "" And TextBoxSub3Result.Text = "" And TextBoxSub4Result.Text = "" And TextBoxSub5Result.Text = "" And TextBoxSub6Result.Text = "" Then

Button2.Enabled = True

End If

End Sub

Private Sub BtnAddEditSubject\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAddEditSubject.Click

TextBoxSub1Result.Enabled = True

TextBoxSub2Result.Enabled = True

TextBoxSub3Result.Enabled = True

TextBoxSub4Result.Enabled = True

TextBoxSub5Result.Enabled = True

TextBoxSub6Result.Enabled = True

BtnAddEditSubject.Enabled = False

BtnReset.Enabled = True

BtnSave.Enabled = True

End Sub

Private Sub BtnSave\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnSave.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "UPDATE settings SET subject1 = @subject1 WHERE stream = @stream And sem = @sem; " \_

& "UPDATE settings SET subject2 = @subject2 WHERE stream = @stream And sem = @sem; " \_

& "UPDATE settings SET subject3 = @subject3 WHERE stream = @stream And sem = @sem; " \_

& "UPDATE settings SET subject4 = @subject4 WHERE stream = @stream And sem = @sem; " \_

& "UPDATE settings SET subject5 = @subject5 WHERE stream = @stream And sem = @sem; " \_

& "UPDATE settings SET subject6 = @subject6 WHERE stream = @stream And sem = @sem; "

myCom.Parameters.AddWithValue("@sem", ComboBoxSem.Text)

myCom.Parameters.AddWithValue("@stream", ComboBoxStream.Text)

myCom.Parameters.AddWithValue("@subject1", TextBoxSub1Result.Text)

myCom.Parameters.AddWithValue("@subject2", TextBoxSub2Result.Text)

myCom.Parameters.AddWithValue("@subject3", TextBoxSub3Result.Text)

myCom.Parameters.AddWithValue("@subject4", TextBoxSub4Result.Text)

myCom.Parameters.AddWithValue("@subject5", TextBoxSub5Result.Text)

myCom.Parameters.AddWithValue("@subject6", TextBoxSub6Result.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Data Updated Successfully", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

myCom.Parameters.Clear()

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

BtnAddEditSubject.Enabled = True

BtnSave.Enabled = False

ButtonReset.Enabled = False

TextBoxSub1Result.Enabled = False

TextBoxSub2Result.Enabled = False

TextBoxSub3Result.Enabled = False

TextBoxSub4Result.Enabled = False

TextBoxSub5Result.Enabled = False

TextBoxSub6Result.Enabled = False

End Sub

Private Sub BtnClose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnClose.Click

MainForm.PanelMain.Controls.Clear()

MainForm.GroupBoxSearch.Parent = MainForm.PanelMain

MainForm.GroupBoxSearch.Show()

MainForm.GroupBoxResult.Parent = MainForm.PanelMain

MainForm.GroupBoxResult.Show()

End Sub

Private Sub BtnSend\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnMsgEdit.Click

TextBoxCollege.Enabled = True

If Not TextBoxCollege.Text = "" Then

TextBoxMessage.Enabled = True

BtnMsgEdit.Enabled = False

BtnEdit.Enabled = True

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT count(\*) FROM messagesetting WHERE message\_stream = '" + ComboBoxStreamMsg.Text + "'"

If myCom.ExecuteScalar > 0 Then

myCom.CommandText = "UPDATE messagesetting SET message\_message = '" + TextBoxMessage.Text + "' WHERE message\_stream = '" + ComboBoxStreamMsg.Text + "'"

myCom.ExecuteNonQuery()

Else

myCom.Parameters.Clear()

myCom.CommandText = "INSERT INTO messagesetting (message\_stream, message\_message) VALUES(@mstream,@mmessage)"

myCom.Parameters.AddWithValue("@mstream", ComboBoxStreamMsg.Text)

myCom.Parameters.AddWithValue("@mmessage", TextBoxMessage.Text)

myCom.ExecuteNonQuery()

End If

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

Else

MessageBox.Show("Please Enter Your College Name", "College Name Required", MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

End Sub

Private Sub BtnEdit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnEdit.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "UPDATE messagesetting SET message\_message = '" + TextBoxMessage.Text + "' WHERE message\_stream = '" + ComboBoxStreamMsg.Text + "'"

myCom.ExecuteNonQuery()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

BtnEdit.Enabled = False

TextBoxMessage.Enabled = False

BtnMsgEdit.Enabled = True

My.Settings.CollegeName = TextBoxCollege.Text

My.Settings.Save()

TextBoxCollege.Enabled = False

End Sub

Private Sub ComboBoxStreamMsg\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxStreamMsg.SelectedIndexChanged

TextBoxCollege.Text = My.Settings.CollegeName

If Not ComboBoxStreamMsg.Text = "" Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT message\_message FROM messagesetting WHERE message\_stream = '" + ComboBoxStreamMsg.Text + "'"

TextBoxMessage.Text = myCom.ExecuteScalar()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stoppeds " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

BtnMsgEdit.Enabled = True

ElseIf ComboBoxStreamMsg.Text = "" Then

BtnEdit.Enabled = False

End If

End Sub

Private Sub BtnReset\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnReset.Click

TextBoxMessage.Clear()

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

MainForm.PanelMain.Controls.Clear()

MainForm.GroupBoxSearch.Parent = MainForm.PanelMain

MainForm.GroupBoxSearch.Show()

MainForm.GroupBoxResult.Parent = MainForm.PanelMain

MainForm.GroupBoxResult.Show()

End Sub

Private Sub ButtonAddStream\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonAddStream.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.Parameters.Clear()

myCom.CommandText = "SELECT count(\*) FROM stream WHERE stream\_name = @st\_name"

myCom.Parameters.AddWithValue("@st\_name", ComboBoxStreamSt.Text)

If myCom.ExecuteScalar = 0 Then

myCom.CommandText = "CREATE TABLE `" + ComboBoxStreamSt.Text + "`" \_

& "(roll\_no VARCHAR(20),year VARCHAR(200),sem VARCHAR(200),subject1 INT," \_

& "subject2 INT,subject3 INT,subject4 INT,subject5 INT," \_

& "subject6 INT,total INT, percentage FLOAT, grade VARCHAR(200), result VARCHAR(200))"

myCom.ExecuteNonQuery()

myCom.CommandText = "INSERT INTO stream (stream\_name) VALUES(@streamname)"

myCom.Parameters.AddWithValue("@streamname", ComboBoxStreamSt.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("New Stream Added Successfully", "Done", MessageBoxButtons.OK, MessageBoxIcon.Information)

Load\_comboitem()

End If

Else

MessageBox.Show("Sorry Stream Already Exist", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "INSERT INTO settings VALUES(@stream,@sem,@subject1,@subject2,@subject3,@subject4,@subject5,@subject6)"

myCom.Parameters.AddWithValue("@stream", ComboBoxStream.Text)

myCom.Parameters.AddWithValue("@sem", ComboBoxSem.Text)

myCom.Parameters.AddWithValue("@subject1", TextBoxSub1Result.Text)

myCom.Parameters.AddWithValue("@subject2", TextBoxSub2Result.Text)

myCom.Parameters.AddWithValue("@subject3", TextBoxSub3Result.Text)

myCom.Parameters.AddWithValue("@subject4", TextBoxSub4Result.Text)

myCom.Parameters.AddWithValue("@subject5", TextBoxSub5Result.Text)

myCom.Parameters.AddWithValue("@subject6", TextBoxSub6Result.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Data Added Successfully", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

myCom.Parameters.Clear()

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

Button1.Enabled = False

End Sub

Private Sub ButtonDeleteStream\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonDeleteStream.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "DROP TABLE `" + ComboBoxStreamSt.Text.ToLower + "`"

myCom.ExecuteNonQuery()

myCom.CommandText = "DELETE FROM stream WHERE stream\_name = '" + ComboBoxStreamSt.Text + "'"

If myCom.ExecuteNonQuery Then

MessageBox.Show("Stream Deleted Successfully", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

myCom.Parameters.Clear()

Load\_comboitem()

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub Load\_comboitem()

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

ComboBoxStreamMsg.Items.Clear()

ComboBoxStreamSt.Items.Clear()

While Reader.Read

ComboBoxStream.Items.Add(Reader("stream\_name").ToString)

ComboBoxStreamMsg.Items.Add(Reader("stream\_name").ToString)

ComboBoxStreamSt.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

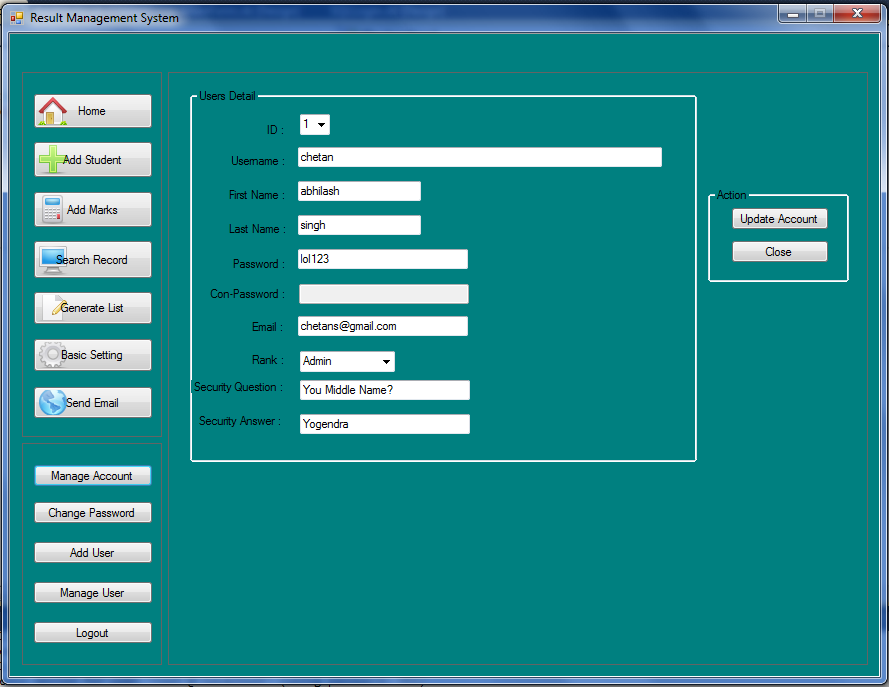
conn.Dispose()

End Try

End Sub

End Class

ManageAccount.vb



Manage Account Code

Imports MySql.Data.MySqlClient

Imports System.Text.RegularExpressions

Public Class ManageAccount

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim UserID As Integer

Public WriteOnly Property GetUserID

Set(ByVal value)

UserID = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

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

ComboBoxRank.Items.Add("User")

ComboBoxRank.Items.Add("Admin")

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM users WHERE user\_id = @user\_id"

myCom.Parameters.AddWithValue("@user\_id", UserID)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

ComboBoxID.Items.Add(Reader("user\_id").ToString)

TextBoxEmail.Text = Reader("email").ToString

TextBoxFName.Text = Reader("firstname").ToString

TextBoxLName.Text = Reader("lastname").ToString

TextBoxPassword.Text = Reader("password").ToString

TextBoxUname.Text = Reader("username").ToString

ComboBoxRank.Text = (Reader("rank").ToString)

TextBoxSecQuest.Text = Reader("secquestion").ToString

TextBoxSecAns.Text = Reader("secanswer").ToString

End While

ComboBoxID.SelectedIndex = 0

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub BtnClose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnClose.Click

MainForm.PanelMain.Controls.Clear()

MainForm.GroupBoxSearch.Parent = MainForm.PanelMain

MainForm.GroupBoxSearch.Show()

MainForm.GroupBoxResult.Parent = MainForm.PanelMain

MainForm.GroupBoxResult.Show()

End Sub

Private Sub BtnUpdateAccount\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnUpdateAccount.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "UPDATE users SET firstname = @firstname WHERE user\_id = @u\_id; " \_

& "UPDATE users SET lastname = @lastname WHERE user\_id = @u\_id; " \_

& "UPDATE users SET username = @username WHERE user\_id = @u\_id; " \_

& "UPDATE users SET password = @password WHERE user\_id = @u\_id; " \_

& "UPDATE users SET email = @email WHERE user\_id = @u\_id; " \_

& "UPDATE users SET rank = @rank WHERE user\_id = @u\_id;" \_

& "UPDATE users SET secquestion = @secquestion WHERE user\_id = @u\_id;" \_

& "UPDATE users SET secanswer = @secanswer WHERE user\_id = @u\_id"

myCom.Parameters.AddWithValue("@u\_id", ComboBoxID.SelectedItem)

myCom.Parameters.AddWithValue("@firstname", TextBoxFName.Text)

myCom.Parameters.AddWithValue("@lastname", TextBoxLName.Text)

myCom.Parameters.AddWithValue("@username", TextBoxUname.Text)

myCom.Parameters.AddWithValue("@password", TextBoxPassword.Text)

myCom.Parameters.AddWithValue("@email", TextBoxEmail.Text)

myCom.Parameters.AddWithValue("@rank", ComboBoxRank.SelectedItem)

myCom.Parameters.AddWithValue("@secquestion", TextBoxSecQuest.Text)

myCom.Parameters.AddWithValue("@secanswer", TextBoxSecAns.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Data Updated Successfully", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub TextBoxEmail\_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBoxEmail.Leave

Dim pattern As String = "^[a-zA-Z][\w\.-]\*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]\*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.]\*[a-zA-Z]$"

Dim emailAddressMatch As Match = Regex.Match(TextBoxEmail.Text, pattern)

If Not emailAddressMatch.Success Then

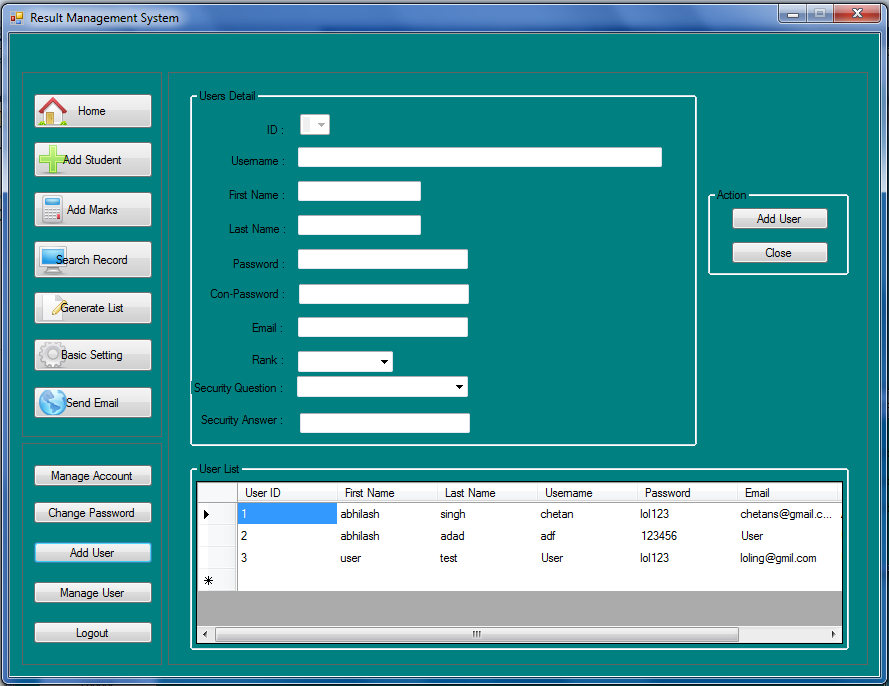
MessageBox.Show("Enter correct email address", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

End If

End Sub

End Class

Add New User.vb



Add User Code

Imports MySql.Data.MySqlClient

Imports System.Text.RegularExpressions

Public Class AddUser

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim myAdap As MySqlDataAdapter

Dim DataTab As DataTable

Private Sub BtnAddUser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAddUser.Click

Dim Fname As String = TextBoxFName.Text

Dim Lname As String = TextBoxLName.Text

Dim Uname As String = TextBoxUname.Text

Dim Rank As String = ComboBoxRank.SelectedItem

Dim Email As String = TextBoxEmail.Text

If TextBoxFName.Text.Length <> 0 And TextBoxLName.Text.Length <> 0 And TextBoxUname.Text.Length <> 0 And TextBoxPassword.Text.Length <> 0 And TextBoxEmail.Text.Length <> 0 And TextBoxConPass.Text.Length <> 0 And Rank.Length <> 0 Then

conn = New MySqlConnection

myCom = New MySqlCommand

If TextBoxPassword.Text = TextBoxConPass.Text Then

Dim Password As String = TextBoxPassword.Text

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "INSERT INTO users (firstname,lastname,username,password,email,rank,secquestion,secanswer) " \_

& "VALUES(@firstname,@lastname,@username,@password,@email,@rank,@secquestion,@secanswer)"

myCom.Parameters.AddWithValue("@firstname", Fname)

myCom.Parameters.AddWithValue("@lastname", Lname)

myCom.Parameters.AddWithValue("@username", Uname)

myCom.Parameters.AddWithValue("@password", Password)

myCom.Parameters.AddWithValue("@email", Email)

myCom.Parameters.AddWithValue("@rank", Rank)

myCom.Parameters.AddWithValue("@secquestion", ComboBoxQuestion)

myCom.Parameters.AddWithValue("@secanswer", TextBoxSecAns.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("User has been added", "Added", MessageBoxButtons.OK, MessageBoxIcon.Information)

conn.Close()

TextBoxConPass.Clear()

TextBoxEmail.Clear()

TextBoxFName.Clear()

TextBoxLName.Clear()

TextBoxPassword.Clear()

TextBoxUname.Clear()

ComboBoxRank.Items.Clear()

TextBoxSecAns.Clear()

ComboBoxQuestion.Items.Clear()

ShowUserList()

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

Else

MessageBox.Show("Sorry password does not match", "Failed", MessageBoxButtons.OK, MessageBoxIcon.Information)

End If

Else

MessageBox.Show("Please Fill up all the fields", "Error", MessageBoxButtons.OK, MessageBoxIcon.Information)

End If

End Sub

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

ComboBoxRank.Items.Clear()

ComboBoxRank.Items.Add("User")

ComboBoxRank.Items.Add("Admin")

ShowUserList()

ComboBoxQuestion.Items.Clear()

ComboBoxQuestion.Items.Add("You Middle Name?")

ComboBoxQuestion.Items.Add("Your Grand Mother Name?")

ComboBoxQuestion.Items.Add("Your Birth Place?")

End Sub

Private Sub BtnClose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnClose.Click

MainForm.PanelMain.Controls.Clear()

MainForm.GroupBoxSearch.Parent = MainForm.PanelMain

MainForm.GroupBoxSearch.Show()

MainForm.GroupBoxResult.Parent = MainForm.PanelMain

MainForm.GroupBoxResult.Show()

End Sub

Private Sub ShowUserList()

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT user\_id AS `User ID`, firstname AS `First Name`, lastname As `Last Name`, " \_

& "username AS Username, password AS Password, email AS Email, rank AS Rank FROM users"

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewUserList.DataSource = DataTab

DataGridViewUserList.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub TextBoxEmail\_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBoxEmail.Leave

Dim pattern As String = "^[a-zA-Z][\w\.-]\*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]\*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.]\*[a-zA-Z]$"

Dim emailAddressMatch As Match = Regex.Match(TextBoxEmail.Text, pattern)

If Not emailAddressMatch.Success Then

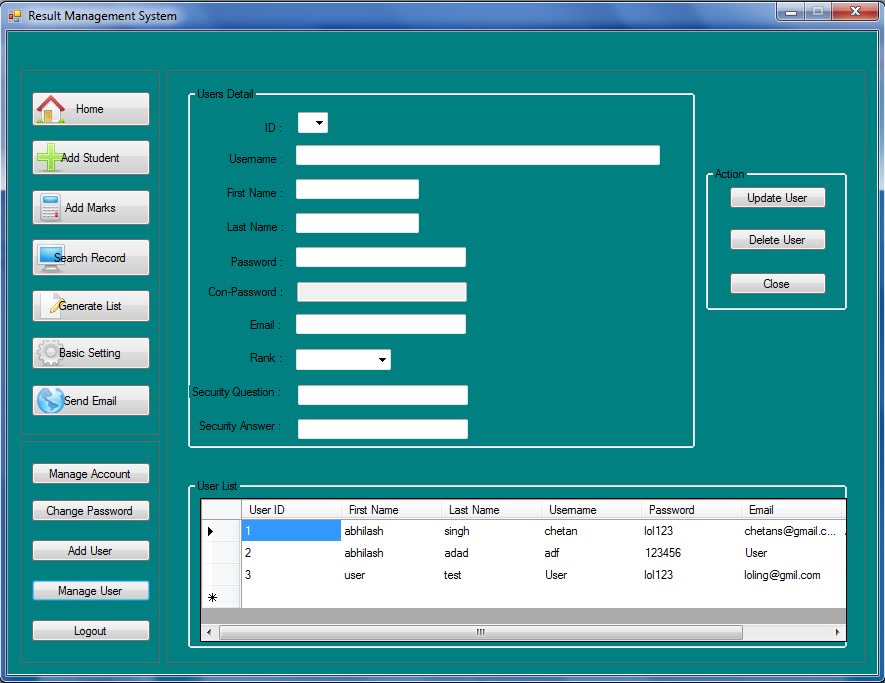
MessageBox.Show("Enter correct email address", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

End If

End Sub

End Class

Manage User.vb



Manage User Code

Imports MySql.Data.MySqlClient

Imports System.Text.RegularExpressions

Public Class ManageUser

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim UserID As Integer

Public WriteOnly Property GetUserID

Set(ByVal value)

UserID = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim myAdap As MySqlDataAdapter

Dim DataTab As DataTable

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

ComboBoxRank.Items.Add("User")

ComboBoxRank.Items.Add("Admin")

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT user\_id FROM users"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

ComboBoxID.Items.Add(Reader("user\_id").ToString)

End While

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

ShowUserList()

End Sub

Private Sub BtnClose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnClose.Click

MainForm.PanelMain.Controls.Clear()

MainForm.GroupBoxSearch.Parent = MainForm.PanelMain

MainForm.GroupBoxSearch.Show()

MainForm.GroupBoxResult.Parent = MainForm.PanelMain

MainForm.GroupBoxResult.Show()

End Sub

Private Sub BtnUpdateUser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnUpdateUser.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "UPDATE users SET firstname = @firstname WHERE user\_id = @u\_id; " \_

& "UPDATE users SET lastname = @lastname WHERE user\_id = @u\_id; " \_

& "UPDATE users SET username = @username WHERE user\_id = @u\_id; " \_

& "UPDATE users SET password = @password WHERE user\_id = @u\_id; " \_

& "UPDATE users SET email = @email WHERE user\_id = @u\_id; " \_

& "UPDATE users SET rank = @rank WHERE user\_id = @u\_id" \_

& "UPDATE users SET secquestion = @secquestion WHERE user\_id = @u\_id;" \_

& "UPDATE users SET secanswer = @secanswer WHERE user\_id = @u\_id"

myCom.Parameters.AddWithValue("@u\_id", ComboBoxID.SelectedItem)

myCom.Parameters.AddWithValue("@firstname", TextBoxFName.Text)

myCom.Parameters.AddWithValue("@lastname", TextBoxLName.Text)

myCom.Parameters.AddWithValue("@username", TextBoxUname.Text)

myCom.Parameters.AddWithValue("@password", TextBoxPassword.Text)

myCom.Parameters.AddWithValue("@email", TextBoxEmail.Text)

myCom.Parameters.AddWithValue("@rank", ComboBoxRank.SelectedItem)

myCom.Parameters.AddWithValue("@secquestion", TextBoxSecQuest.Text)

myCom.Parameters.AddWithValue("@secanswer", TextBoxSecAns.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Data Updated Successfully", "Updated", MessageBoxButtons.OK, MessageBoxIcon.Information)

ShowUserList()

End If

myCom.Parameters.Clear()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ComboBoxID\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxID.SelectedIndexChanged

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM users WHERE user\_id = @user\_id"

myCom.Parameters.AddWithValue("@user\_id", ComboBoxID.SelectedItem)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

TextBoxEmail.Text = Reader("email").ToString

TextBoxFName.Text = Reader("firstname").ToString

TextBoxLName.Text = Reader("lastname").ToString

TextBoxPassword.Text = Reader("password").ToString

TextBoxUname.Text = Reader("username").ToString

ComboBoxRank.Text = Reader("rank").ToString

TextBoxSecQuest.Text = Reader("secquestion").ToString

TextBoxSecAns.Text = Reader("secanswer").ToString

End While

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub ShowUserList()

conn = New MySqlConnection

myCom = New MySqlCommand

myAdap = New MySqlDataAdapter

DataTab = New DataTable

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT user\_id AS `User ID`, firstname AS `First Name`, lastname As `Last Name`, " \_

& "username AS Username, password AS Password, email AS Email, rank AS Rank FROM users"

myAdap.SelectCommand = myCom

myAdap.Fill(DataTab)

DataGridViewUserList.DataSource = DataTab

DataGridViewUserList.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub TextBoxEmail\_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBoxEmail.Leave

Dim pattern As String = "^[a-zA-Z][\w\.-]\*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]\*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.]\*[a-zA-Z]$"

Dim emailAddressMatch As Match = Regex.Match(TextBoxEmail.Text, pattern)

If Not emailAddressMatch.Success Then

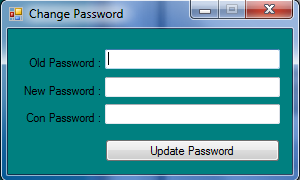
MessageBox.Show("Enter correct email address", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

End If

End Sub

End Class

Change Password.vb



Change Password Code

Imports MySql.Data.MySqlClient

Public Class ChangePassword

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim UserID As Integer

Public WriteOnly Property GetUserID

Set(ByVal value)

UserID = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Private Sub BtnUpdatePass\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnUpdatePass.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT password FROM users WHERE user\_id = @us\_id"

myCom.Parameters.AddWithValue("@us\_id", UserID)

Dim pass As String = myCom.ExecuteScalar

If txtboxoldpass.Text = pass Then

If txtboxnewpass.Text = txtpassconpass.Text Then

myCom.CommandText = "UPDATE users SET password = @password WHERE user\_id = @u\_id"

myCom.Parameters.AddWithValue("@u\_id", UserID)

myCom.Parameters.AddWithValue("@password", txtboxnewpass.Text)

If myCom.ExecuteNonQuery Then

MessageBox.Show("Password updated succesfully", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

txtboxnewpass.Clear()

txtboxoldpass.Clear()

txtpassconpass.Clear()

End If

myCom.Parameters.Clear()

conn.Close()

Else

MessageBox.Show("Password do not match", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

End If

Else

MessageBox.Show("This is not you password", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

End If

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

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

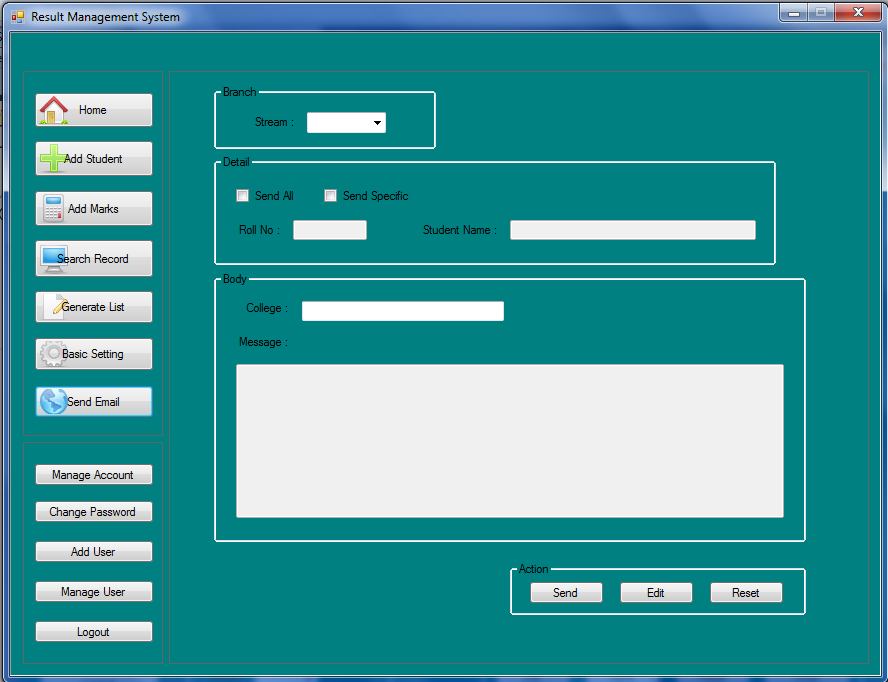
Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2

Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2

End Sub

End Class

Send Mail.vb



Send Mail Code

Imports MySql.Data.MySqlClient

Imports System.Net.Mail

Public Class SendResult

Dim SQLString As String

Public WriteOnly Property GetSqlString

Set(ByVal value)

SQLString = value

End Set

End Property

Dim conn As MySqlConnection

Dim myCom As MySqlCommand

Dim Studemail As String = Nothing

Dim roll\_no As String = Nothing

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

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

ComboBoxStream.Items.Clear()

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM stream"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read

ComboBoxStream.Items.Add(Reader("stream\_name").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stoppeds " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

CheckBoxAll.Checked = True

If CheckBoxAll.Checked = True Then

CheckBoxAll.Checked = False

TextBoxName.Enabled = False

TextBoxRoll.Enabled = False

End If

End Sub

Private Sub TextBoxRoll\_KeyUp(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyEventArgs) Handles TextBoxRoll.KeyUp

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM students WHERE s\_rollno = @roll\_no"

myCom.Parameters.AddWithValue("@roll\_no", TextBoxRoll.Text)

Dim Reader As MySqlDataReader = myCom.ExecuteReader

While Reader.Read()

TextBoxName.Text = Reader("s\_name").ToString

Studemail = Reader("s\_email").ToString

End While

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Private Sub BtnSend\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnSend.Click

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

If CheckBoxSpecific.Checked = True Then

SendEmail("Single")

ElseIf CheckBoxAll.Checked = True And Not ComboBoxStream.SelectedItem = "" Then

SendEmail("All")

End If

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End Sub

Public Sub SendEmail(ByVal emailx As String)

If emailx = "Single" Then

Dim MyMailMessage As New MailMessage()

MyMailMessage.From = New MailAddress("chetan.singh188.cs@gmail.com")

MyMailMessage.To.Add(Studemail)

MyMailMessage.Subject = My.Settings.CollegeName

If BtnEdit.Enabled = True Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM messagesetting WHERE message\_stream = '" + ComboBoxStream.Text + "'"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

While Reader.Read

TextBoxMessage.Text = Reader("message\_message").ToString.Replace("NULL", TextBoxRoll.Text)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

MyMailMessage.Body = TextBoxMessage.Text

End If

Dim SMTPServer As New SmtpClient("smtp.gmail.com")

SMTPServer.Port = 587

SMTPServer.Credentials = New System.Net.NetworkCredential("chetan.singh188.cs@gmail.com", "25287421")

SMTPServer.EnableSsl = True

Try

SMTPServer.Send(MyMailMessage)

MessageBox.Show("Email Sent")

Catch ex As SmtpException

MessageBox.Show(ex.Message)

End Try

ElseIf emailx = "All" Then

myCom.CommandText = "SELECT s\_email,s\_rollno from students WHERE s\_stream = '" + ComboBoxStream.Text + "'"

Dim Readerx As MySqlDataReader = myCom.ExecuteReader

Try

While Readerx.Read

Dim MyMailMessage As New MailMessage()

MyMailMessage.From = New MailAddress("chetan.singh188.cs@gmail.com")

MyMailMessage.To.Add(Readerx("s\_email").ToString)

MyMailMessage.Subject = My.Settings.CollegeName

If BtnEdit.Enabled = True Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

myCom.Connection = conn

myCom.CommandText = "SELECT \* FROM messagesetting WHERE message\_stream = '" + ComboBoxStream.Text + "'"

Dim Reader As MySqlDataReader = myCom.ExecuteReader

ComboBoxStream.Items.Clear()

While Reader.Read

TextBoxMessage.Text = Reader("message\_message").ToString.Replace("NULL", Readerx("s\_rollno").ToString)

End While

Reader.Close()

myCom.Parameters.Clear()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

MyMailMessage.Body = TextBoxMessage.Text

End If

Dim SMTPServer As New SmtpClient("smtp.gmail.com")

SMTPServer.Port = 587

SMTPServer.Credentials = New System.Net.NetworkCredential("chetan.singh188.cs@gmail.com", "25287421")

SMTPServer.EnableSsl = True

SMTPServer.Send(MyMailMessage)

End While

MessageBox.Show("Email Sent")

Catch ex As SmtpException

MessageBox.Show(ex.Message)

End Try

End If

End Sub

Private Sub BtnReset\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnReset.Click

TextBoxName.Clear()

TextBoxRoll.Clear()

End Sub

Private Sub BtnEdit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnEdit.Click

TextBoxMessage.Enabled = True

TextBoxMessage.Clear()

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

TextBoxMessage.Clear()

myCom.Connection = conn

myCom.CommandText = "SELECT message\_message FROM messagesetting WHERE message\_stream = '" + ComboBoxStream.Text + "'"

TextBoxMessage.Text = myCom.ExecuteScalar()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

ComboBoxStream.Enabled = False

BtnEdit.Enabled = False

End Sub

Private Sub CheckBoxAll\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CheckBoxAll.Click

If CheckBoxAll.Checked Then

TextBoxCollege.Text = My.Settings.CollegeName

If CheckBoxAll.Checked = True Then

CheckBoxSpecific.Checked = False

TextBoxName.Enabled = False

TextBoxRoll.Enabled = False

TextBoxRoll.Clear()

TextBoxName.Clear()

If BtnEdit.Enabled = True Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

TextBoxMessage.Clear()

myCom.Connection = conn

myCom.CommandText = "SELECT message\_message FROM messagesetting WHERE message\_stream = '" + ComboBoxStream.Text + "'"

TextBoxMessage.Text = myCom.ExecuteScalar()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End If

End If

End If

End Sub

Private Sub ComboBoxStream\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBoxStream.SelectedIndexChanged

If CheckBoxAll.Checked Then

TextBoxCollege.Text = My.Settings.CollegeName

If CheckBoxAll.Checked = True Then

CheckBoxSpecific.Checked = False

TextBoxName.Enabled = False

TextBoxRoll.Enabled = False

TextBoxRoll.Clear()

TextBoxName.Clear()

If BtnEdit.Enabled = True Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

TextBoxMessage.Clear()

myCom.Connection = conn

myCom.CommandText = "SELECT message\_message FROM messagesetting WHERE message\_stream = '" + ComboBoxStream.Text + "'"

TextBoxMessage.Text = myCom.ExecuteScalar()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End If

End If

ElseIf CheckBoxSpecific.Checked Then

If CheckBoxSpecific.Checked = True Then

CheckBoxAll.Checked = False

TextBoxName.Enabled = True

TextBoxRoll.Enabled = True

TextBoxRoll.Clear()

TextBoxName.Clear()

If BtnEdit.Enabled = True Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

TextBoxMessage.Clear()

myCom.Connection = conn

myCom.CommandText = "SELECT message\_message FROM messagesetting WHERE message\_stream = '" + ComboBoxStream.Text + "'"

TextBoxMessage.Text = myCom.ExecuteScalar()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End If

End If

End If

End Sub

Private Sub CheckBoxSpecific\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CheckBoxSpecific.Click

If CheckBoxSpecific.Checked Then

TextBoxCollege.Text = My.Settings.CollegeName

If CheckBoxSpecific.Checked = True Then

TextBoxName.Enabled = True

TextBoxRoll.Enabled = True

CheckBoxAll.Checked = False

TextBoxRoll.Clear()

TextBoxName.Clear()

If BtnEdit.Enabled = True Then

conn = New MySqlConnection

myCom = New MySqlCommand

If conn.State = ConnectionState.Closed Then

conn.ConnectionString = SQLString

End If

Try

If conn.State = ConnectionState.Closed Then

conn.Open()

End If

TextBoxMessage.Clear()

myCom.Connection = conn

myCom.CommandText = "SELECT message\_message FROM messagesetting WHERE message\_stream = '" + ComboBoxStream.Text + "'"

TextBoxMessage.Text = myCom.ExecuteScalar()

conn.Close()

Catch exMyError As MySqlException

MessageBox.Show("Connection to Database is stopped " + exMyError.ToString, "Database Connection", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

conn.Dispose()

End Try

End If

End If

End If

End Sub

End Class

**Conclusion**

At last I have buildup and made a perfect Result Management System as per my capability and named as “Result Management System” according to the user and college requirement. Our project is very simple, user friendly and easy to use.

So, i hope that our modified application will be beneficial to every Bachelor College and new user with all new look and feels.

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient GUI based component. This application is working properly and meeting to all user requirements, This component can be easily plugged in many other Windows Based System in .NET environment.