1. **INTRODUCTION**

The project titled **VACCINATION VERIFICATION USING FACIAL RECOGNITION BASED ON EMBEDDED AI** is vaccine verification system software for check and dublicate finding in a vaccination certificate .The project “**Vaccination Verification Using Facial Recognition Based On Embedded Ai”** is developed in DOTNET, which mainly focuses on basic operations in a verify like adding new beneficiaries, viewing certificate, and verifying the beneficiaries, manipulate user data.

The “**Vaccination Verification Using Facial Recognition Based On Embedded Ai**” is a windows application written for 32-bit Windows operating systems, designed to help users maintain and organize verification center.

Our software is easy to use for both beginners and advanced users. It features a familiar and well thought-out, an attractive user interface, combined with strong facial recognition Identification and view the certification.

The certificate generation facility of beneficiaries helps to get a good idea of which are the view the records, makes users possible to generate reports’ hard copy.

# SYSTEM ANALYSIS

**2.1 Existing System:**

System Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. Here the key question is- what all problems exist in the present system? What must be done to solve the problem? Analysis begins when a user or manager begins a study of the program using existing system.

During analysis, data collected on the various files, decision points and transactions handled by the present system.

The commonly used tools in the system are Data Flow Diagram, interviews, etc. Training, experience and common sense are required for collection of relevant information needed to develop the system. The success of the system depends largely on how clearly the problem is defined, thoroughly investigated and properly carried out through the choice of solution. A good analysis model should provide not only the mechanisms of problem understanding but also the frame work of the solution. Thus it should be studied thoroughly by collecting data about the system. Then the proposed system should be analyzed thoroughly in accordance with the needs.

In our existing system all the transaction of books are done manually, So taking more time for a transaction like borrowing a book or returning a book and also for searching of members and books. Another major disadvantage is that to preparing the list of books borrowed and the available books in the library will take more time, currently it is doing as a one day process for verifying all records. So after conducting the feasibility study we decided to make the manual Library management system to be computerized.

**2.2 Proposed System:**

Proposed system is an Enhanced Library Management System. Through our software user can add members, add books, search members, search books, update information, edit information, borrow and return books and also calculate the fine delay the fine amount in quick time. Our proposed system has the following advantages.

* User friendly interface
* Fast access to database
* Less error
* More Storage Capacity
* Facial recognation facility
* Look and Feel Environment
* Quick verification

All the manual difficulties in managing the Library have been rectified by implementing computerization.

* 1. **Modules Description:**

This project “Enhanced Library Management System” is developed in dot net. Which is mainly focuses on basic operations in a Library like adding new member ,new books , and updating new information ,searching books and members and facility to borrow and return books and also calculate the fine amount.

In earlier system all the transaction of books are done manually ,so taking more time for a transaction like borrowing a book and returning a book and also for searching of members and books.

Through our software user can add members, add books, search members, search books, update information, edit information, borrow and return books in quick time.

The software Library Management System has four main modules.

1. Member Maintenance Module

2. Book Maintenance Module

3. Transaction Maintenance

4. Report Maintenance

1. **Beneficiaries module :**

This module is keep track of all important information of beneficiaries, By the help of this module user can add or create new beneficiaries details, modify or delete existing details.

1. **Admin panel module :**

In this module administator user can create new user account,modify and delete the existing data,view the user record only by using This module.

1. **Verify Modules :**

This module is used only facial verification of beneficiaries and checking the data.

1. **View Modules :**

In this module helps only for view the beneficiaries details and generating vaccination certificate useful reports.

1. **SYSTEM SPECIFICATION**

**3.1 HARDWARE REQUIREMENTS :**

**Processor : AMD Pro**

**Ram : 4 GB**

**Hard disk : 50 GB**

**3.2 SOFTWARE REQUIREMENTS :**

**Operating System : Windows 10**

**Package : Visual Studio2010**

**Front End : C#.Net**

**Back End : SQL Server**

**3.3 SOFTWARE DESCRIPTION :**

**3.3.1 Introduction to Dot Net Framework :**

The DOTNET is the technology from Microsoft, on which all other Microsoft technologies will be depending on in future. DOTNET technology was introduced by Microsoft, to catch the market from the SUN's Java. The market very fast. With the world depending more and more on the Internet/Web and java related tools becoming the best choice for the web applications. DOTNET framework comes with a single class library. And that’s all programmers need to learn!!

Whether they write the code in C# or VB.NET or J#, it doesn't matter, you just use the DOTNET class library. There is no classes specific to any language. There is nothing more you can do in a language, which you can't do in any other DOTNET language. C# with the same number of lines of code, same performance and same efficiency, because everyone uses same DOTNET class library.

**Features of DOTNET :**

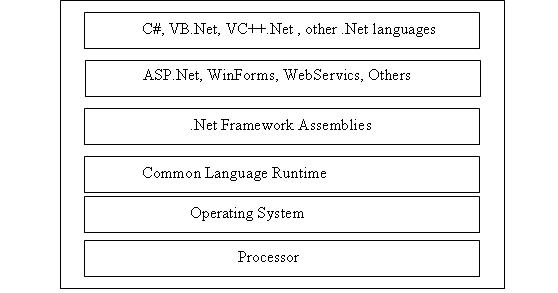
* It is a platform neutral framework.
* It is a layer between the operating system and the programming language.
* DOTNET provides a common set of class libraries.
* In future versions of Windows, DOTNET will be freely distributed as part of operating system and users will never have to install DOTNET separately.

**Architecture of DOTNET Framework :**

The DOTNET framework can only be exploited by languages that are compliant with DOTNET. Most of Microsoft languages have been made to fully comply with DOTNET. DOTNET also introduces Web Forms, Web Services and Windows Forms. The reason why they have been shown separately and not as a part of a particular language is that these technologies can be used by any DOTNET compliant language.

These are the common class libraries (much like Java packages) that can be used by any DOTNET compliant language.

The diagram given below describes various components of DOTNET Framework.

****

**DOTNET Technology :**

* Third generation component model
* IPC in COM is replaced by Intermediate Language(IL or MSIL)
* Interoperability by compiling code into IL.
* Metadata

**Common Language Runtime (CLR) :**

The CLR is the heart of DOTNET framework. It is DOTNET equivalent of Java Virtual Machine (JVM).It is the runtime that converts a MSIL (Micro Soft Intermediate Language) code into the host machine language code, which is then executed appropriately.

* The CLR provides a number of services that include:
* Loading and execution of codes
* Memory isolation for application
* Verification of type safety
* Compilation of IL into native executable code
* Providing metadata
* Automatic garbage collection
* Enforcement of Security
* Interoperability with other systems
* Managing exceptions and errors
* Provide support for debugging and profiling

**Common Type System (CTS) :**

The language interoperability, and DOTNET Class Framework, are not possible without all the language sharing the same data types. What this means is that an in should mean the same in VB, VC++, C# and all other DOTNET compliant languages. Same idea follows for all the other data types. This is achieved through introduction of Common Type System (CTS). CTS, much like Java, defines every data type as a Class. Every DOTNET compliant language must stick to this definition. Since CTS defines every data type as a class; this means that only Object- Oriented (or Object-Based) languages can achieve DOTNET compliance.

**Common Language Specification (CLS) :**

One of the obvious themes of DOTNET is unification and interoperability between various programming languages. In order to achieve this; certain rules must be laid and all the languages must follow these rules.

* In other words we cannot have languages running around creating their own extensions and their own fancy new data types.
* CLS is the collection of the rules and constraints that every language (that seeks to achieve DOTNET compatibility) must follow.
* Microsoft has defined three level of CLS compatibility/compliance.
* The goals and objectives of each compliance level have been set aside.
* The three compliance levels with their brief description are given below:

**Compliant producer :**

* The component developed in this type of language can be used by any other language.

**Consumer :**

* The language in this category can use classes produced in any other language.
* In simple words this means that the language can instantiate classes developed in other language.
* This is similar to how COM components can be instantiated by your ASP code.

**Extender :**

Languages in this category cannot just use the classes as in CONSUMER category; but can also extend classes using inheritance. Languages that come with Microsoft Visual Studio namely Visual C++, Visual Basic and C#; all satisfy the above three categories .Vendors can select any of the above categories as the targeted compliance level(s) for their languages.

**Microsoft Intermediate Language (MSIL) :**

A DOTNET programming language (C#, VB.NET, J# etc.) does not compile into executable code; instead it compiles into an intermediate code called Microsoft Intermediate Language (MSIL). As a programmer one need not worry about the syntax of MSIL - since our source code in automatically converted to MSIL.

The MSIL code is then send to the CLR (Common Language Runtime) that converts the code to machine language which is then run on the host machine.

**Managed Code :**

The role of CLR doesn‘t end once we have compiled our code to MSIL and a JIT compiler has compiled this to native code .Code written using the DOTNET framework, is managed code when it is executed. This stage is usually referred to as being at runtime. This means that the CLR looks after our applications, by managing memory, handling security, allowing cross language debugging and so on. By contrast, applications that do not run under the control of the CLR are said to be unmanaged and certain languages such as C++ can be used to write such applications, that for example, to access low level functions of the operating systems. However in C# we can only write code that runs in a managed environment.

**Unified classes :**

The term DOTNET framework refers to the group of technologies that form the development foundation for the Microsoft DOTNET platform. The key technologies in this group are the run time and the class libraries .The DOTNET programming languages including Visual Basic DOTNET, Microsoft Visual C# and C++ managed extensions and many other programming languages from various vendors utilize DOTNET services and features through a common set of unified classes .The unified classes provide a consistent method of accessing the platforms functionality.

**Framework Base Classes :**

The DOTNET Framework has an extensive set of class libraries. This includes classes for:

**Data Access :**

* High Performance data access classes for connecting to SQL Server or any other OLEDB provider. 

**XML Supports :**

* Next generation XML support that goes far beyond the functionality of MSXML.



**Directory Services :**

* Support for accessing Active Directory/LDPA using ADSI. 

**Regular Expression :**

* Support for above and beyond that found in Perl 5. 

**Queuing Supports :**

* Provides a clean object-oriented set of classes for working with MSMQ.

**3.3.2 ADO.NET:**

**ADO.NET** is a data access technology from the Microsoft .NET Framework that provides communication between relational and non-relational systems through a

common set of components. ADO.NET is a set of computer software components

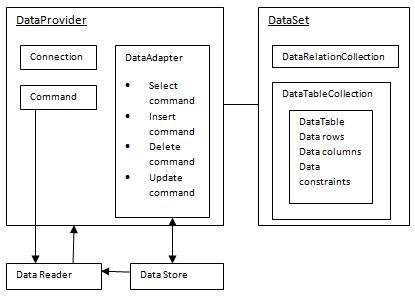
that programmers can use to access data and data services from a database. It is a

part of the base class library that is included with the Microsoft .NET Framework. It

is commonly used by programmers to access and modify data stored in relational

database systems, though it can also access data in non-relational data sources.

ADO.NET is sometimes considered an evolution of ActiveX Data Objects (ADO) technology, but was changed so extensively that it can be considered an entirely new product.

****

ADO.NET is conceptually divided into consumers and data providers. The

consumers are the applications that need access to the data, and the providers are

the software components that implement the interface and thereby provide the

data to the consumer.

Functionality exists in Visual Studio IDE to create specialized subclasses of the DataSet classes for a particular database schema, allowing convenient access to each field in the schema through strongly typed properties. This helps catch more programming errors at compile-time and enhances the IDE's Intelligence feature.

**Data provider :**

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

1. **Connection :**

The Connection object is the first component of ADO.NET. The connection object opens a connection to your data source.

You can connect your C#.Net application to data in a SQL Server database using the Microsoft DOTNET Framework Data Provider for SQL Server. The first step in a C#.Net application is to create an instance of the Server object and to establish its connection to an instance of SQL Server.

The SqlConnection Object is Handling the part of physical communication between the application and the [SQL Server](http://vb.net-informations.com/ado.net-dataproviders/ado.net-sqlconnection.htm) Database. An instance of the SqlConnection class in .NET Framework is supported the [Data Provider](http://vb.net-informations.com/ado.net-dataproviders/ado.net-dataproviders-tutorial.htm) for SQL Server Database. The SqlConnection instance takes [Connection String](http://vb.net-informations.com/ado.net/connectionstring.htm) as argument and pass the value to the Constructor statement. When the connection is established , SQL Commands may be executed, with the help of the Connection Object, to retrieve or manipulate data in the database. Once the Database activities over , Connection should be closed and release the database resources .

All of the configurable aspects of a database connection are represented in the Connection object, which includes ConnectionString() and ConnectionTimeout.

Connection object helps in accessing and manipulating a database. Database transactions are also dependent upon the Connection object

connetionString = "Data Source=ServerName;Initial Catalog=DatabaseName;User ID=UserName;Password=Password"

In ADO.NET the type of the Connection is depended on what Database system you are working with. The following are the commonly using the connections in the ADO.NET

* SqlConnection
* OleDbConnection
* OdbcConnection

1. **Command :**

The Command Object in ADO.NET executes SQL statements and Stored Procedures against the data source specified in the [Connection](http://vb.net-informations.com/ado.net-dataproviders/ado.net-connection.htm) Object. The Command Object required an instance of a Connection Object for executing the SQL statements. That is, for retrieving data or execute an SQL statement against a Data Source , you have to create a Connection Object and open a connection to the Data Source, and assign the open connection to the connection property of the Command Object. When the Command Object return result set , a [Data Reader](http://vb.net-informations.com/ado.net-dataproviders/ado.net-datareader.htm) is used to retrieve the result set.

The Command Object has a property called CommandText, which contains a String value that represents the command that will be executed in the Data Source. When the CommandType property is set to StoredProcedure, the CommandText property should be set to the name of the stored procedure.

[**ExecuteNonQuery**](http://vb.net-informations.com/ado.net-dataproviders/ado.net-executenonquery-sqlcommand.htm) ExecuteNonQuery method will return number of rows effected with INSERT, DELETE or UPDATE operations. This ExecuteNonQuery method will be used only for insert, update and delete, Create, and SETstatements.

com. [ExecuteNonQuery](http://vb.net-informations.com/ado.net-dataproviders/ado.net-executenonquery-sqlcommand.htm)();

[**ExecuteReader**](http://vb.net-informations.com/ado.net-dataproviders/ado.net-executereader-sqlcommand.htm)Execute Reader will be used to return the set of rows, on execution of SQL Query or Stored procedure using command object. This one is forward only retrieval of records and it is used to read the table values from first to last.

Rd=com.ExecuteReader();

[**ExecuteScalar**](http://vb.net-informations.com/ado.net-dataproviders/ado.net-executescalar-sqlcommand.htm)Execute Scalar will return first row first column value i.e. it will return single value and ignore other values on execution of SQL Query or Stored procedure using command object. It’s very fast to retrieve single values from database.

cmd.ExecuteScalar();

**CommandText** The Command Object has a property called CommandText, which contains a String value that represents the command that will be executed in the Data Source .The Command object is used to perform action on the data source. Command object can execute stored procedures and T-SQL commands.

Com.CommendText=”select \* from tab”;

1. **DataReader :**

DataReader Object in ADO.NET is a stream-based , forward-only, read-only retrieval of query results from the Data Source, which do not update the data. The DataReader cannot be created directly from code, they created only by calling the ExecuteReader method of a Command Object

**ExecuteReader()**

The DataReader is built as a way to retrieve and examine the rows returned in response to your query as quickly as possible.

**DataReader = Command.ExecuteReader()**

No DataSet is created; in fact, no more than one row of information from the data source is in-memory at a time. This makes the DataReader quiet efficient at returning large amounts of data.

**Read()** method in the DataReader is used to read the rows from DataReader and it always moves forward to a new valid row, if any row exist .

**DataReader.Read()**

The data returned by a DataReader is always read only.  This class was built to be a lightweight forward only, read only, way to run through data quickly (this was called a firehose cursor in ADO).

However, if you need to manipulate schema or use some advance display features such as automatic paging, you must use a DataAdapter and DataSet.

 DataReader object works in connected model.

1. **DataAdapter** **:**

DataAdapter is a part of the [ADO.NET](http://vb.net-informations.com/ado.net/ado.net-architecture.htm) [Data Provider](http://vb.net-informations.com/ado.net-dataproviders/ado.net-dataproviders-tutorial.htm). DataAdapter provides the communication between the Dataset and the Datasource . We can use the DataAdapter in combination with the DataSet Object. That is these two objects combine to enable both data access and data manipulation capabilities.

**Fill** method The DataAdapter takes the results of a database query from a Command object and pushes them into a DataSet using the DataAdapter.Fill() method.

**Update** method Additionally the DataAdapter.Update() method will negotiate any changes to a DataSet back to the original data source.

DataAdapter object works in connected model. DataAdapter performs five following steps:

1. Create/open the connection
2. Fetch the data as per command specified
3. Generate XML file of data
4. Fill data into DataSet.
5. Close connection.

**Command Builder**

 It is used to save changes made in-memory cache of data on backend. The work of Command Builder is to generate Command as per changes in Data Rows. Command Builder generates command on basis of row state. There are five row state:

1. Unchanged
2. Added
3. Deleted
4. Modified
5. Detached

Command Builder works on add, delete and modified row state only. Detached is used when object is not created from row state.

**Transaction**  The Transaction object is used to execute backend transaction. Transactions are used to ensure that multiple changes to database rows occur as a single unit of work. The Connection class has a Begin Transaction method that can be used to create a Transaction.

A definite best practice is to ensure that Transactions are placed in Using statements for rapid cleanup if they are not committed.  Otherwise the objects (and any internal locks that may be needed) will remain active until the GC gets around to cleaning it up.

**Parameter**  object is used to solve SQL Injection attack problem while dealing with the user input parameters. Parameter object allows passing parameters into a Command object the Parameter class allows you to quickly put parameters into a query without string concatenation.

**3.3.3 C -Sharp Language (C#) :**

**Introduction :**

Microsoft Corporation, developed a new computer programming language C# pronounced as ‗C-Sharp‘. C# is a simple, modern, object oriented, and type safe programming language derived from C and C++. C# is a purely object-oriented language like as Java. It has been designed to support the key features of .NET framework. C# modernize C++ by enhancing some of its features and adding a few new features .C# borrows Java‘s features such as grouping of classes, interface and implementation together in one file so the programmers can easily edit the codes .C# also handles objects using reference, the same way as Java .C# uses VB‘s approach to form designing, namely, dragging controls from a tool box, dropping them onto forms, and writing events handlers for them.

**Object Oriented Programming:**

It is a type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.

**OOPS CONCEPTS :**

**Object :**

Objects are the basic run-time entities in an object-oriented system. Programming problem is analyzed in terms of objects and nature of communication between them. When a program is executed, objects interact with each other by sending messages. Different objects can also interact with each other without knowing the details of their data or code.

An object is an instance of a class. A class must be instantiated into an object before it can be used in the software. More than one instance of the same class can be in existence at any one time.

**Class :**

A class is a collection of objects of a similar type. Once a class is defined, any number of objects can be created which belong to that class. A class is a blueprint, or prototype, that defines the variables and the methods common to all objects of a certain kind.

**Instance :**

The instance is the actual object created at runtime. One can have an instance of a class or a particular object.

**State :**

The set of values of the attributes of a particular object is called its state. The object consists of state and the behavior that's defined in the object's class.

**Method :**

Method describes the object’s abilities. A Dog has the ability to bark. So bark() is one of the methods of the Dog class.

**Message Passing :**

The process by which an object sends data to another object or asks the other object to invoke a method. Message passing corresponds to "method calling".

**Abstraction :**

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes.

**Encapsulation :**

It is the mechanism that binds together code and data in manipulates, and keeps both safe from outside interference and misuse. In short, it isolates a particular code and data from all other codes and data. A well-defined interface controls the access to that particular code and data.</p. /> The act of placing data and the operations that perform on that data in the same class. The class then becomes the 'capsule' or container for the data and operations. Storing data and functions in a single unit (class) is encapsulation. Data cannot be accessible to the outside world and only those functions which are stored in the class can access it.

**Inheritance :**

It is the process by which one object acquires the properties of another object. This supports the hierarchical classification. Without the use of hierarchies, each object would need to define all its characteristics explicitly. However, by use of inheritance, an object need only define those qualities that make it unique within its class. It can inherit its general attributes from its parent. A new sub-class inherits all of the attributes of all of its ancestors.

**Polymorphism :**

Polymorphism means the ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends on the data types used in the operation.

It is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. In general, polymorphism means "one interface, multiple methods", This means that it is possible to design a generic interface to a group of related activities. This helps reduce complexity by allowing the same interface to be used to specify a general class of action. It is the compiler's job to select the specific action (that is, method) as it applies to each situation.

**Generalization :**

Generalization describes an is-a relationship which represent a hierarchy between classes of objects. Eg :- a "fruit" is a generalization of "apple", "orange", "mango" and many others.animal is the generalization of pet.

**Specialization :**

Specialization means an object can inherit the common state and behavior of a generic object. However, each object needs to define its own special and particular state and behavior. Specialization means to subclass. animal is the generalization and pet is the specialization, indicating that a pet is a special kind of animal .

**Design goals :**

The language is intended to be a simple, modern, general purpose, object-oriented programming language .The language is intended for use in developing software components suitable for deployment in distributed environments. Portability is very important for source code and programmers, especially those already familiar with Ch and C++. Support for internationalization is very important. C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.

**Features of C#**

**1. Simplicity**

* All the Syntax of java is like C++. There is no preprocessor, and much larger library.
* C# code does not require header files. All code is written inline.

**2. Consistent behavior**

* C# introduced an unified type system which eliminates the problem of varying ranges of integer types.
* All types are treated as objects and developers can extend the type system simply and easily.

**3. Modern programming language**

C# supports number of modern features, such as:

* Automatic Garbage Collection
* Error Handling features
* Modern debugging features
* Robust Security features

**4. Pure Object- Oriented programming language**

* In C#, everything is an object. There are no more global functions, variable and constants. It supports all three object oriented features:
* Encapsulation
* Inheritance
* Polymorphism

**5. Type Safety**

* Type safety promotes robust programming. Some examples of type safety are:
* All objects and arrays are initialized by zero dynamically
* An error message will be produced , on use of any uninitialized variable
* Automatic checking of array out of bound and etc.

**6. Feature of Versioning**

* Making new versions of software module work with the existing applications is known as versioning.
* Its achieve by the keywords **new** and **override.**

**7. Compatible with other language**

* C# enforces the .NET common language specifications (CLS) and therefore allows inter-operation with other .NET language.

**8. Inter-operability**

* C# provides support for using COM objects, no matter what language was used to author them.
* C# also supports a special feature that enables a program to call out any native API.

**Syntax :**

Semicolons are used to denote the end of a statement. Curly brackets are used to group statements. Statements are commonly grouped into methods (functions), methods into classes, and classes into namespaces. Variables are assigned using an equals sign, but compared using two consecutive equals signs. Square brackets are used with arrays, both to declare them and to get a value at a given index in one of them.

**Distinguishing features :**

1. **Portability :**

By design, C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI).Most of its intrinsic types correspond to value-types implemented by the CLI framework .However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++.

1. **Meta programming :**

Meta programming via C# attributes is part of the language. Many of these attributes duplicate the functionality of GCC's and Visual C++'s platform-dependent preprocessor directives.

1. **Methods and functions :**

Methods in programming language are the members of a class in a project, some methods have signatures and some don't have signatures. Methods can be void or can return something like string, integer, double, decimal, float and Boolean. If a method is void it means that the method does not return any data type. Like C++, and unlike Java, C# programmers must use the keyword virtual to allow methods to be overridden by subclasses. Extension methods in C# allow programmers to use static methods as if they were methods from a class's method table, allowing programmers to add methods to an object that they feel should exist on that object and its derivatives. The type dynamic allows for run-time method binding, allowing for JavaScript-like method calls and run-time object composition. C# has support for strongly-typed function pointers via the keyword delegate. Like the Qt framework's pseudo-C++ *signal* and C# has semantics specifically surrounding publish-subscribe style events, though C# uses delegates to do so.

1. **Property :**

C# provides properties as syntactic sugar for a common pattern in which a pair of methods, accessory (getter) and matador(setter) encapsulate operations on a single attribute of a class. No redundant method signatures for the getter/setter implementations need be written, and the property may be accessed using attribute syntax rather than more verbose method calls.

**(V) Namespace :**

A C# namespace provides the same level of code isolation as a Java package or a C++ **namespace**, with very similar rules and features to a package.

1. **Memory access :**

In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references, which always either point to a "live" object or have the well-defined null value; it is impossible to obtain a reference to a "dead" object (one that has been garbage collected), or to a random block of memory. An unsafe pointer can point to an instance of a value-type, array, string, or a block of memory allocated on a stack. Code that is not marked as unsafe can still store and manipulate pointers through the System.IntPtrtype, but it cannot dereference them.

1. **Exception :**

Checked exceptions are not present in C# (in contrast to Java). This has been a conscious decision based on the issues of scalability and version ability.

1. **Language Integrated Query – LINQ :**

C# has the ability to utilize LINQ through the Microsoft.NET Framework with the I Enumerable Interface a developer can query any DOTNET collection class, XML documents, ADO.NET datasets, and SQL databases .There are some advantages to using LINQ in C# and they are as follows: intelligence support, strong filtering capabilities, type safety with compile error checking ability, and brings consistency for querying data over a variety of sources. There are several different language structures that can be utilized with C# with LINQ and they are query expressions, lambda expressions, anonymous types, implicitly typed variables, extension methods, and object initializes.

1. **CLASS LIBRARY :**

DOTNET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

1. **OVERLOADING :**

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

1. **MULTITHREADING :**

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

1. **STRUCTURED EXCEPTION HANDLING :**

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

**(xii) OBJECTIVES OF dotNET FRAMEWORK :**

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.

2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.

3. Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications.

**Data Types, Identifiers, Variables, Constants and Literals :**

**Identifiers & Variables :**

Identifiers refer to the names of variables, functions arrays, classes, etc. created by programmer. They are fundamental requirement of any language. Each language has its own rules for naming these identifiers. To name the variables of your program, you must follow strict rules. In fact, everything else in your program must have a name. There are some rules you must follow when naming your objects. On this site, here are the rules we will follow:

* The name must start with a letter or an underscore
* After the first letter or underscore, the name can have letters, digits, and/or underscores
* The name must not have any special characters other than the underscore
* The name cannot have a space

**C# Keywords :**

C# uses a series of words, called keywords, for its internal use. This means that you must avoid naming your objects using one of these keywords. They are:

EX:

* Const
* continue
* else
* class
* or

**Data types :**

C# is a type-safe language. Variables are declared as being of a particular type, and each variable is constrained to hold only values of its declared type. Variables can hold either value types or reference types, or they can be pointers. Here's a quick recap of the difference between value types and reference types.

**(i) value type :**

* it directly contains an object with some value.

**(ii) reference types :**

* it directly contains is something which refers to an object.

|  |
| --- |
| **Value Type :** |
| * C# defines the following value types : |
| * Primitives - int i; * Enum - enum state { off, on }; * Struct - struct Point{ int x, y; }; |

It is possible in C# to define your own value types by declaring enumerations or structs. These user-defined types are mostly treated in exactly the same way as C#'s predefined value types, although compilers are optimized for the latter.

**Reference Types :**

The pre-defined reference types are object and string, where object - is the ultimate base class of all other types. New reference types can be defined using 'class', 'interface', and 'delegate' declarations. There fore the reference types are :

**Predefined Reference Types :**

* Object
* String

**User Defined Reference Types :**

* Classes
* Interfaces
* Delegates
* Arrays

Reference types actually hold the value of a memory address occupied by the object they reference. Consider the following piece of code, in which two variables are given a reference to the same object.

**3.3.4 FACIAL RECOGNITION :**

The facial recognition has been a problem worked on around the world for many persons; this problem has emerged in multiple fields and sciences, especially in computer science, other fields that are very interested in this technology are: Mechatronic, Robotic, criminalistics, etc. In this article, I work on this interesting topic using EmguCV cross platform .NET wrapper to the Intel OpenCV image processing library and C# .NET, these libraries allow me to capture and process image of a capture device in real time. The main goal of this article is to show and explain the easiest way in which to implement a face detector and recognizer in real time for multiple persons using Principal Component Analysis (PCA) with eigenface for implementing it in multiple fields.

Facial recognition is a computer application composed for complex algorithms that use mathematical and matricial techniques, these get the image in raster mode (digital format) and then process and compare pixel by pixel using different methods for obtaining faster and reliable results. Obviously, these results depend on the machine used to process this due to the huge computational power that these algorithms, functions and routines require, these are the most popular techniques used for solving this modern problem.

**Techniques :**

**Traditional**

Some facial recognition algorithms identify faces by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face detection. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation. Recognition algorithms can be divided into two main approaches, geometric, which looks at distinguishing features, or photometric, which is a statistical approach that distill an image into values and comparing the values with templates to eliminate variances. Popular recognition algorithms include Principal Component Analysis with eigenface, Linear Discriminate Analysis, Elastic Bunch Graph Matching fisherface, the Hidden Markov model, and the neuronal motivated dynamic link matching.

Text taken from **[1]**

An example of EigenFaces:



Image taken from **[4]**

**3-D**

A newly emerging trend, claimed to achieve previously unseen accuracies, is three-dimensional face recognition. This technique uses 3-D sensors to capture information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin. One advantage of 3-D facial recognition is that it is not affected by changes in lighting like other techniques. It can also identify a face from a range of viewing angles, including a profile view. Even a perfect 3D matching technique could be sensitive to expressions. For that goal, a group at the Technion applied tools from metric geometry to treat expressions as isometries.

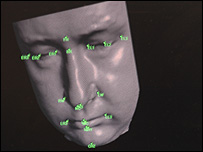


Image taken from **[2]**

**Skin Texture Analysis**

Another emerging trend uses the visual details of the skin, as captured in standard digital or scanned images. This technique, called skin texture analysis, turns the unique lines, patterns, and spots apparent in a person’s skin into a mathematical space Tests have shown that with the addition of skin texture analysis, performance in recognizing faces can increase 20 to 25 percent. It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems.

Text taken from **[1]**

**EmguCV**



Emgu CV is a cross platform .NET wrapper to the Intel OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython, etc. The wrapper can be compiled in Mono and run on Linux / Mac OS X.

Text taken from **[3]**

In my own words, EmguCV is an awesome Wrapper, this allows to make very interesting things and tasks of computer vision. This library set lets do an unlimited amount of wonderful projects in this field, EmguCV have many functions that let us work with CPU and GPU increases the performance dramatically with the latest mentioned.

This awesome SW project let work and do:

* Optical Character Recognition(OCR)
* Face Detection
* Pedestrian Detection
* Kinect projects
* 3D reconstruction
* SURF feature detector... between many others interesting tasks

EmguCV Basics: How I Start to Work?

If you have never worked with this wrapper, you want see how to add references to project or solve problems, take a look at this good article/tutorial by **C\_Johnson**:

* [Creating Your First EMGU Image Processing Project](http://www.codeproject.com/Articles/257502/Creating-Your-First-EMGU-Image-Processing-Project)

Another interesting Web/blog with multiple tutorials to star with emguCV, image processing, and face recognition by **mehwish87** is:

* [EmguCV and basic image processing tutorials](http://fewtutorials.bravesites.com/tutorials)

Using the Code

First, declare all variables and important objects to use:

C#

Copy Code

*//Declaration of all variables, vectors and haarcascades*

Image<bgr,> currentFrame;

Capture grabber;

HaarCascade face;

HaarCascade eye;

MCvFont font = new MCvFont(FONT.CV\_FONT\_HERSHEY\_TRIPLEX, 0.5d, 0.5d);

Image<gray,> result, TrainedFace = null;

Image<gray,> gray = null;

List<image<gray,>> trainingImages = new List<image<gray,>>();

List<string> labels= new List<string>();

List<string> NamePersons = new List<string>();

int ContTrain, NumLabels, t;

string name, names = null;

Then load the haarcascades for face detection, then I do a little “procedure” to load previous trained faces and labels for each image stored previously:

C#

Shrink ▲   Copy Code

*//Load haarcascades for face detection*

face = new HaarCascade("haarcascade\_frontalface\_alt\_tree.xml");

eye = new HaarCascade("haarcascade\_eye.xml");

try

{

*//Load of previous trained faces and labels for each image*

string Labelsinfo =

File.ReadAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt");

string[] Labels = Labelsinfo.Split('%');

NumLabels = Convert.ToInt16(Labels[0]);

ContTrain = NumLabels;

string LoadFaces;

for (int tf = 1; tf < NumLabels+1; tf++)

{

LoadFaces = "face" + tf + ".bmp";

trainingImages.Add(new Image<gray,>

(Application.StartupPath + "/TrainedFaces/" + LoadFaces));

labels.Add(Labels[tf]);

}

}

catch(Exception e)

{

*//MessageBox.Show(e.ToString());*

MessageBox.Show("Nothing in binary database,

please add at least a face", "Trained faces load",

MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

Initialize the capture device, and FrameGrabber event that performs the detection and process of images for each frame captured:

C#

Copy Code

grabber = new Capture();

grabber.QueryFrame();

*//Initialize the FrameGraber event*

Application.Idle += new EventHandler(FrameGrabber);

button1.Enabled = false;

Passing to FrameGrabber event (main part of prototype), we use the most important methods and objects: DetectHaarCascade and EigenObjectRecognizer and perform operations for each face detected in one frame:

C#

Shrink ▲   Copy Code

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

*//Action for each element detected*

foreach (MCvAvgComp f in facesDetected[0])

{

t = t + 1;

result = currentFrame.Copy(f.rect).Convert<gray,>().

Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

*//draw the face detected in the 0th (gray) channel with blue color*

currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);

if (trainingImages.ToArray().Length != 0)

{

*//TermCriteria for face recognition with*

*//numbers of trained images like maxIteration*

MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

*//Eigen face recognizer*

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(

trainingImages.ToArray(),

labels.ToArray(),

5000,

ref termCrit);

name = recognizer.Recognize(result);

*//Draw the label for each face detected and recognized*

currentFrame.Draw(name, ref font,

new Point(f.rect.X - 2, f.rect.Y - 2), new Bgr(Color.LightGreen));

}

}

**Parameters**

* haarObj: Haar classifier cascade in internal representation scaleFactor: The factor by which the search window is scaled between the subsequent scans, for example, 1.1 means increasing window by 10%.
* minNeighbors: Minimum number (minus 1) of neighbor rectangles that makes up an object. All the groups of a smaller number of rectangles than min\_neighbors-1 are rejected. If min\_neighbors is 0, the function does not have any grouping at all and returns all the detected candidate rectangles, which may be useful if the user wants to apply a customized grouping procedure
* flag: Mode of operation. Currently, the only flag that may be specified is CV\_HAAR\_DO\_CANNY\_PRUNING. If it is set, the function uses Canny edge detector to reject some image regions that contain too few or too many edges and thus cannot contain the searched object. The particular threshold values are tuned for face detection and in this case, the pruning speeds up the processing.
* minSize: Minimum window size. By default, it is set to the size of samples the classifier has been trained on (~20x20 for face detection).

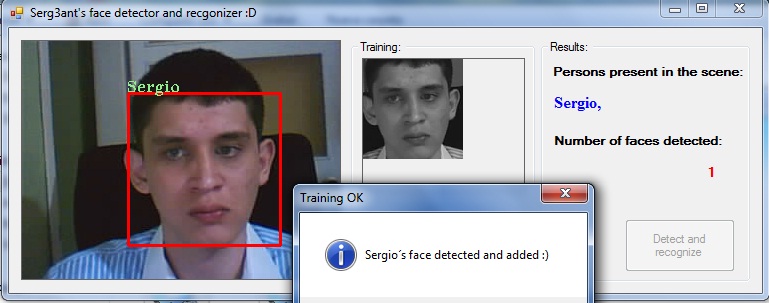
How to Train the Prototype?

I do this part in the easiest way possible, the prototype detects faces constantly (Each frame) and you can add this detected face in the image database with one label respectably, the face trained image will show in the imageBoxFrameGrabber and the process will be finished!!

**Keep in mind**: The face recognition algorithms based in PCA (Principal Component Analysis) do multiple comparisons and matches between a face detected and the trained images stored in binary database for this reason, and for improving the accuracy of recognition, you should add several images of the same person in different angles, po

sitions and luminance conditions, this training makes this prototype solid and very accurate.

Example:



Code of training button (This performs the adding of training faces and labels for each):

C#

Shrink ▲   Copy Code

try

{

*//Trained face counter*

ContTrain = ContTrain + 1;

*//Get a gray frame from capture device*

gray = grabber.QueryGrayFrame().Resize

(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

*//Face Detector*

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

*//Action for each element detected*

foreach (MCvAvgComp f in facesDetected[0])

{

TrainedFace = currentFrame.Copy(f.rect).Convert<gray,>();

break;

}

*//resize face detected image for force to compare the same size with the*

*//test image with cubic interpolation type method*

TrainedFace = result.Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

trainingImages.Add(TrainedFace);

labels.Add(textBox1.Text);

*//Show face added in gray scale*

imageBox1.Image = TrainedFace;

*//Write the number of trained faces in a file text for further load*

File.WriteAllText(Application.StartupPath +

"/TrainedFaces/TrainedLabels.txt",

trainingImages.ToArray().Length.ToString() + "%");

*//Write the labels of trained faces in a file text for further load*

for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)

{

trainingImages.ToArray()[i - 1].Save

(Application.StartupPath + "/TrainedFaces/face" + i + ".bmp");

File.AppendAllText(Application.StartupPath +

"/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");

}

MessageBox.Show(textBox1.Text + "´s face detected and added :)",

"Training OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch

{

MessageBox.Show("Enable the face detection first",

"Training Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

'Enable the face detection first' warning?

For add a face to the model you need initiate the camera and face recognition engine first, you only should press the "Detect and recognize" button.

when the camera shows your face on live image you can add and train the model.

How to Improve the Recognition?

The default parameters (scale\_factor=1.1, min\_neighbors=3, flags=0) are tuned for accurate yet slow object detection.

Also, you may modify the size for a big value, modify this in the code:

C#

Copy Code

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.1,

3,

0,

new Size(20, 20));

Additionally, modify the third param to **2500** or **3000** instead 5000, this modification makes the EigenObjectRecognizer be more strict/accurate.

C#

Copy Code

*//Eigen face recognizer*

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(

trainingImages.ToArray(),

labels.ToArray(),

5000,

ref termCrit);

How to Improve the Performance for Slower CPUs?

All image processing algorithms demand many computational power, in this case, the internals process carried on for the CPU with this sw prototype are so hard for slower or monocore CPUS, the easy way for improving the performance of this demo is modify the

parameters that use the DetectHaarCascade method, these allow to decrement the

number of iterations, critic sections and comparisons of the real time image captured for the Webcam improving notoriously the application performance.

Keep in mind: reduce the values of these parameters will affect the efficiency of

recognition Algorithms.

First Option

For a faster operation on real video images, the settings are: scale\_factor=1.2, min\_neighbors=2, flags=CV\_HAAR\_DO\_CANNY\_PRUNING, min\_size=<minimum> (for example, ~1/4 to 1/16 of the image area in case of video conferencing).

Also, you may modify the Minsize parameter for a big value.

C#

Copy Code

*// DetectHaarCascade Config for optimal performance*

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

2,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

Second Option

Get a “thumbnail”or resize the original image capture for reducing the time of processing in the FrameGrabber method modify the size values for a minor size (originally is 320x240).

Example:

C#

Copy Code

*//Get the current frame form capture device*

currentFrame = grabber.QueryFrame().Resize

(260, 200, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

Remember to do the same in the Training button:

C#

Copy Code

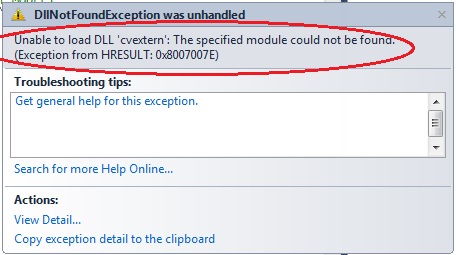
gray = grabber.QueryGrayFrame().Resize(260, 200, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

The Prototype Does Not Show the Strangers as Unknown, Why?

Is a limitation of Eigenface approach, isn't error, use LBPH algorithm instead (have pros and shortcomings) available on EmguCV 2.4.2 and above.

What Do You Need to Run/Use this Project without Errors?

For avoid errors like this:



1. First, download and decompress *FaceRecPro\_Demo.zip* it has “external” OS libraries(DLLs) used for some OpenCV functions, after copying these DLLs in *C:/Windows/System32* or in *bin* folder of this project.
2. Then download EmguCV(Wrapper of OpenCV for C#) at <http://sourceforge.net/projects/emgucv/files/>, install it and then go to *C:\Emgu\emgucv-windows-x86 2.2.1.1150\bin* folder and copy: opencv\_calib3d220, opencv\_contrib220, opencv\_core220, opencv\_features2d220, opencv\_ffmpeg220, opencv\_flann220, opencv\_gpu220, opencv\_highgui220, opencv\_imgproc220, opencv\_legacy220, opencv\_ml220, opencv\_objdetect220 and opencv\_video220 in *C:/Windows/System32* or in *bin* folder of this project.

Remember these DLLs are OpenCV Libraries and are necessary to run any project that uses EgmuCV.

**3.4 BACKEND DESIGN :**

**SQL SERVER :**

**Microsoft SQL Server** is a [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft). As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many [concurrent users](http://en.wikipedia.org/wiki/Concurrent_user). Its primary [query languages](http://en.wikipedia.org/wiki/Query_language) are [T-SQL](http://en.wikipedia.org/wiki/Transact-SQL) and [ANSI SQL](http://en.wikipedia.org/wiki/SQL).

**HISTORY:**

**GENESIS**

Prior to version 7.0 the [code base](http://en.wikipedia.org/wiki/Code_base) for MS SQL Server was sold by [Sybase SQL Server](http://en.wikipedia.org/wiki/Sybase_SQL_Server) to Microsoft, and was Microsoft's entry to the enterprise-level database market, competing against [Oracle](http://en.wikipedia.org/wiki/Oracle_database), [IBM](http://en.wikipedia.org/wiki/IBM), and, later, [Sybase](http://en.wikipedia.org/wiki/Sybase). Microsoft, Sybase and [Ashton-Tate](http://en.wikipedia.org/wiki/Ashton-Tate) originally worked together to create and market the first version named SQL Server 1.0 for [OS/2](http://en.wikipedia.org/wiki/OS/2) (about 1989) which was essentially the same as Sybase SQL Server 3.0 on [Unix](http://en.wikipedia.org/wiki/Unix) , [VMS](http://en.wikipedia.org/wiki/Virtual_Memory_System), etc.

Since the release of SQL Server , advances have been made in performance, the client IDE tools, and several complementary systems that are packaged with SQL Server . These include:

* an [extract-transform-load (ETL)](http://en.wikipedia.org/wiki/Extract,_transform,_load) tool (SQL Server Integration Services or [SSIS](http://en.wikipedia.org/wiki/SQL_Server_Integration_Services))
* a [Reporting Server](http://en.wikipedia.org/wiki/SQL_Server_Reporting_Services)
* an [OLAP](http://en.wikipedia.org/wiki/OLAP) and [data mining](http://en.wikipedia.org/wiki/Data_mining) server ([Analysis Services](http://en.wikipedia.org/wiki/Microsoft_Analysis_Services))

[Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR) integration was introduced with this version, enabling one to write SQL code as Managed Code by the CLR. For relational data, [T-SQL](http://en.wikipedia.org/wiki/T-SQL) has been augmented with error handling features (try/catch) and support for recursive queries with CTEs (Common Table Expressions). SQL Server 2005 has also been enhanced with new indexing algorithms, syntax and better error recovery systems.

**FEATURES SQL SERVER:**

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consist of six type of objects,

They are,

1. TABLE

2. QUERY

3. FORM

4. REPORT

5. MACRO

**TABLE:**

A database is a collection of data about a specific topic.

**VIEWS OF TABLE:**

We can work with a table in two types,

1. Design View

2. Datasheet View

**Design View**

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

**Datasheet View**

To add, edit or analyses the data itself we work in tables datasheet view mode.

**QUERY:**

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

MS SQL Server is a powerful database management system and the user can create application that requires little or no programming. It supports GUI features and an entire programming language, Visual Studio Application which can be used to develop richer and more developed application.

There are quite a few reasons, the first being that SQL is a feature rich program that can handle any database related task you have. You can create places to store your data build tools that make it easy to read and modify your database contents, and ask questions of your data. SQL is a relational database, a database that stores information about related objects. In MS SQL that database means a collection of tables that hold data. It collectively stores all the other related objects such as queries, forms and reports that are used to implement function effectively.

The MS SQL database can act as a back end database for .NET as a front end, MS SQL supports the user with its powerful database management functions. A beginner can create his/her own database very simply by some mouse clicks. Another good reason to use SQL as backend tool is that it is a component of the overwhelmingly popular Microsoft office software suite.

MS SQL however is a relational database, which means that you can define relationships among the data it contains. Relational database, are superior to flat file databases because you can store discrete information.

Microsoft SQL Server is a full-featured relational database management system (RDBMS) that offers a variety of administrative tools to ease the burdens of database development, maintenance and administration. In this article, we'll cover six of the more frequently used tools: Enterprise Manager, Query Analyzer, SQL Profiler, Service Manager, Data Transformation Services and Books Online. Let's take a brief look at each:

**Enterprise Manager** is the main administrative console for SQL Server installations. It provides you with a graphical "birds-eye" view of all of the SQL Server installations on your network. You can perform high-level administrative functions that affect one or more servers, schedule common maintenance tasks or create and modify the structure of individual databases.

**Query Analyzer** offers a quick and dirty method for performing queries against any of your SQL Server databases .It's a great way to quickly pull information out of a database in response to a user request, test queries before implementing them in other applications, create/modify stored procedures and execute administrative tasks.   
 **SQL Profiler** provides a window into the inner workings of your database. You can monitor many different event types and observe database performance in real time. SQL Profiler allows you to capture and replay system "traces" that log various activities. It's a great tool for optimizing databases with performance issues or troubleshooting particular problems.

**Service Manager** is used to control the MSSQL Server (the main SQL Server process), MSDTC (Microsoft Distributed Transaction Coordinator) and SQL Server Agent processes. An icon for this service normally resides in the system tray of machines running SQL Server. You can use Service Manager to start, stop or pause any one of these services.

**Data Transformation Services (DTS)** provide an extremely flexible method for importing and exporting data between a Microsoft SQL Server installation and a large variety of other formats. The most commonly used DTS application is the "Import and Export Data" wizard found in the SQL Server program group.

**Books Online** is an often overlooked resource provided with SQL Server that contains answers to a variety of administrative, development and installation issues. It's a great resource to consult before turning to the Internet or technical support. Hopefully, this article has provided you with a brief introduction to the various tools available to Microsoft SQL Server users. Now get out there and give them a whirl!

# 

# SQL Server Architecture

Microsoft® SQL Server data is stored in databases. The data in a database is organized into the logical components visible to users. A database is also physically implemented as two or more files on disk. When using a database, you work primarily with the logical components such as tables, views, procedures, and users. The physical implementation of files is largely transparent. Typically, only the database administrator needs to work with the physical implementation.

Each instance of SQL Server has four system databases (**master**, **model**, **tempdb**, and **msdb**) and one or more user databases. Some organizations have only one user database, containing all the data for their organization. Some organizations have different databases for each group in their organization, and sometimes a database used by a single application. For example, an organization could have one database for sales, one for payroll, one for a document management application, and so on. Sometimes an application uses only one database; other applications may access several databases.

It is not necessary to run multiple copies of the SQL Server database engine to allow multiple users to access the databases on a server. An instance of the SQL Server is capable of handling thousands of users working in multiple databases at the same time. Each instance of SQL Server makes all databases in the instance available to all users that connect to the instance, subject to the defined security permissions.

When connecting to an instance of SQL Server, your connection is associated with a particular database on the server. This database is called the **current database**. You are usually connected to a database defined as your default database by the system administrator.

SQL Server allows you to **detach** databases from an instance of SQL Server, then **reattach** them to another instance, or even attach the database back to the same instance. If you have a SQL Server database file, you can tell SQL Server when you connect to attach that database file with a specific database name.

**4. SYSTEM DESIGN**

**4.1 SYSTEM ARCHITECHTURE :**

View

Verify

View

Verify

Admin Panel

Beneficiaries

Login

**4.2 DATA FLOW DESIGN :**

**1.** **BENEFICIARIES:**

* Aadhar No
* Name
* Gender
* Mobile
* Date of birth
* Ref id
* Train image
* F’dose
* Next due
* F’vaccinateby
* F’vaccinateat
* F’batch
* S’dose
* S’vaccinateby
* S’vaccinateat
* S’batch

Beneficiaries

Administrator access only

Login

**2. ADMIN PANEL:**

User List

User Account

* Aadhar No
* Name
* Type
* Mobile No
* Username
* Password

Admin Panel

Administrator access only

Login

**3. VERIFY:**

Verify

* Detect
* View
* Detect
* View

Verify

Login

**4. VIEW :**

View

* Load
* View Certificate
* Load
* View Certificate

View

Login

**4.3 TABLE DESIGN :**

**Login Table :**

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data type** | **Key** |
| **Aadharno** | Varchar(100) | Primary key |
| **Name** | Varchar(100) | Null |
| **Type** | Varchar(100) | Null |
| **Mobile** | Varchar(100) | Null |
| **Uname** | Varchar(100) | Null |
| **Password** | Varchar(100) | Null |
| **Status** | Varchar(100) | Null |

**Beneficiaries Table :**

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Key** |
| **Aadhar** | Varchar(100) | Primary key |
| **Name** | Varchar(100) | Null |
| **Age** | Varchar(100) | Null |
| **Gender** | Varchar(100) | Null |
| **Mobile** | Varchar(100) | Null |
| **Bref** | Varchar(100) | Null |
| **Vaccinename** | Varchar(100) | Null |
| **Fdose** | Varchar(100) | Null |
| **Nxtdose** | Varchar(100) | Null |
| **Fvaccinateby** | Varchar(100) | Null |
| **Fvaccinateat** | Varchar(100) | Null |
| **Fbatch** | Varchar(100) | Null |
| **Sdose** | Varchar(100) | Null |
| **Svaccinateby** | Varchar(100) | Null |
| **Svaccinateat** | Varchar(100) | Null |
| **Sbatch** | Varchar(100) | Null |

**5. SYSTEM TESTING AND SYSTEM IMPLEMENTATION**

**5.1 SYSTEM TESTING :**

It is the stage of implementation, which ensures that system works accurately and effectively before the live operation Commences. It is a confirmation that all are correct and opportunity to show the users that the system must be tested with text data and show that the system will operate successfully and produce expected results under expected conditions.

Before implementation, the proposed system must be tested with raw data to ensure that the modules of the system work correctly and satisfactorily. The system must be tested with valid data to achieve its objective.

The purpose of system testing is to identify and correct errors in the candidate system. As important as this phase is, it is one that is frequently compromised. Typically, the project the schedule or the user is eager to go directly to conversion. Actually, testing is done to achieve the system goal.

Testing is vital to the parts of the system are correct; the goal will be successfully achieved. Inadequate testing or non-testing leads to errors that may not appear until months later Appearance of the problem.

**System testing** of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified [requirements](http://en.wikipedia.org/wiki/Requirements). System testing falls within the scope of [black box testing](http://en.wikipedia.org/wiki/Black_box_testing), and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed [integration testing](http://en.wikipedia.org/wiki/Integration_testing) and also the software system itself integrated with any applicable hardware system(s).

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

#### Unit Test

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

#### System Test

Several modules constitute a project. If the project is long-term project, several developers write the modules. Once all the modules are integrated, several errors may arise. The testing done at this stage is called system test.

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

#### Functional Test

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

#### Acceptance Testing

Testing the system with the intent of confirming readiness of the product and customer acceptance.

**5.2 SYSTEM MAINTENANCE**

All system is dynamic and subjects to constantly changing requirements. Effort must be devoted to adapting them and design must be flexible specified so that such changes can be easily implemented. This activity is called system maintains. It includes improvement of system functions and correction of errors.

Back up for the entire database files are taken and stored in secondary storage devices like magnetic tapes and disks so that is possible to restore the system at the earliest. If there is a breakdown or collapse, then the system gives provision to restore database files.

Storing data in a separate secondary device leads to an effective and efficient maintains of the system.

The master file has flags for maintains. After the mentioned period, the rejection suppliers, unused data in the files will be deleted in the master file. This method is the increasing the memory to store the data.

**5.3 Sample Coding :**

**Log in :**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Runtime.InteropServices;

using System.Configuration;

namespace MultiFaceRec

{

public partial class Login : Form

{

public string username;

SqlConnection con;

public Login()

{

InitializeComponent();

ConnectionStringSettings conc = ConfigurationManager.ConnectionStrings["connect"];

string constr = conc.ConnectionString;

con = new SqlConnection(constr);

}

[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]

private extern static void ReleaseCapture();

[DllImport("user32.DLL", EntryPoint = "SendMessage")]

private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam, int lParam);

private void button1\_Click(object sender, EventArgs e)

{

con.Close();

string querry = "select \* from Login where uname='"+textBox1.Text+"' and password='"+textBox3.Text+"'";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

dr.Close();

con.Close();

con.Open();

string state = "Y";

string s = "update Login set status='" + state.ToString() + "' where uname='" + textBox1.Text + "' and password='" + textBox3.Text + "'";

SqlCommand com = new SqlCommand(s, con);

com.ExecuteNonQuery();

con.Close();

MessageBox.Show("Login Successfully");

this.Close();

}

else

{

MessageBox.Show("Please Enter The Currect Username&Password ");

}

}

private void LoadTheme()

{

foreach (Control btns in this.Controls)

{

if (btns.GetType() == typeof(Button))

{

Button btn = (Button)btns;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

btn.Font = new System.Drawing.Font("Monotype Corsiva", 14.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

}

}

label1.ForeColor = ThemeColor.SecondaryColor;

}

private void Login\_Load(object sender, EventArgs e)

{

LoadTheme();

con.Open();

string state = "N";

string s = "update Login set status='" + state.ToString() + "'";

SqlCommand com = new SqlCommand(s, con);

com.ExecuteNonQuery();

con.Close();

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

ReleaseCapture();

SendMessage(this.Handle, 0x112, 0xf012, 0);

}

private void button2\_Click(object sender, EventArgs e)

{

textBox1.Clear();

textBox3.Clear();

}

private void textBox3\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

con.Close();

string querry = "select \* from Login where uname='" + textBox1.Text + "' and password='" + textBox3.Text + "'";

;

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

dr.Close();

con.Close();

con.Open();

string state = "Y";

string s = "update Login set status='" + state.ToString() + "' where uname='" + textBox1.Text + "' and password='" + textBox3.Text + "'";

SqlCommand com = new SqlCommand(s, con);

com.ExecuteNonQuery();

con.Close();

MessageBox.Show("Login Successfully");

this.Close();

}

else

{

MessageBox.Show("Please Enter The Currect Username&Password ");

}

}

}

}

}

**Dashboard :**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using MultiFaceRec.Forms;

using System.Runtime.InteropServices;

using System.Configuration;

namespace MultiFaceRec

{

public partial class Home : Form

{

SqlConnection con;

private Button currentButton;

private Random random;

private int tempIndex;

private Form activeForm;

public Home()

{

InitializeComponent();

ConnectionStringSettings conc = ConfigurationManager.ConnectionStrings["connect"];

string constr = conc.ConnectionString;

con = new SqlConnection(constr);

close.Visible = false;

random = new Random();

this.Text = string.Empty;

this.ControlBox = false;

this.MaximizedBounds = Screen.FromHandle(this.Handle).WorkingArea;

}

[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]

private extern static void ReleaseCapture();

[DllImport("user32.DLL", EntryPoint = "SendMessage")]

private extern static void SendMessage(System.IntPtr hWnd,int wMsg,int wParam,int lParam);

private Color SelectThemeColor()

{

int index = random.Next(ThemeColor.ColorList.Count);

while (tempIndex == index)

{

index=random.Next(ThemeColor.ColorList.Count);

}

tempIndex = index;

string color = ThemeColor.ColorList[index];

return ColorTranslator.FromHtml(color);

}

private void ActiveButton(object btnsender)

{

if (btnsender != null)

{

if (currentButton != (Button)btnsender)

{

DisableButton();

Color color = SelectThemeColor();

currentButton = (Button)btnsender;

currentButton.BackColor = color;

currentButton.ForeColor = Color.Gold;

currentButton.Font = new System.Drawing.Font("Monotype Corsiva", 14.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

panelTitleBar.BackColor = color;

ThemeColor.PrimaryColor = color;

close.Visible = true;

}

}

}

private void DisableButton()

{

foreach (Control previousBtn in panel2.Controls)

{

if (previousBtn.GetType() == typeof(Button))

{

previousBtn.BackColor = Color.LightBlue;

previousBtn.ForeColor = Color.Black;

previousBtn.Font = new System.Drawing.Font("Monotype Corsiva", 14.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

}

}

}

public void OpenChildForm(Form childForm, object btnSender)

{

if (activeForm != null)

{

activeForm.Close();

}

ActiveButton(btnSender);

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

this.panelDesktop.Controls.Add(childForm);

this.panelDesktop.Tag = childForm;

childForm.BringToFront();

childForm.Show();

label2.Text = childForm.Text;

}

private void Home\_Load(object sender, EventArgs e)

{

string ste="Y";

string querry = "select \* from Login where status='" + ste + "'";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

label3.Text=dr["name"].ToString();

}

dr.Close();

con.Close();

}

private void viwbtn\_Click(object sender, EventArgs e)

{

string querry = "select \* from Login where status='Y'";

con.Close();

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

OpenChildForm(new Viw(), sender);

dr.Close();

con.Close();

}

else

{

MessageBox.Show("Please Login");

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (activeForm != null)

{

activeForm.Close();

}

Reset();

}

private void button4\_Click(object sender, EventArgs e)

{

string querry = "select \* from Login where uname='Admin' and status='Y'";

con.Close();

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

OpenChildForm(new Members(), sender);

dr.Close();

con.Close();

}

else

{

MessageBox.Show("Administrator Access Only");

}

}

private void button5\_Click(object sender, EventArgs e)

{

string querry = "select \* from Login where uname='Admin' and status='Y'";

con.Close();

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

OpenChildForm(new Vaccinated(), sender);

dr.Close();

con.Close();

}

else

{

MessageBox.Show("Administrator Access Only");

}

}

private void button6\_Click(object sender, EventArgs e)

{

string querry = "select \* from Login where status='Y'";

con.Close();

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

OpenChildForm(new FrmPrincipal(), sender);

dr.Close();

con.Close();

}

else

{

MessageBox.Show("Please Login");

}

}

private void close\_Click(object sender, EventArgs e)

{

if (activeForm != null)

{

activeForm.Close();

}

Reset();

}

private void Reset()

{

DisableButton();

label2.Text = "Dashboard";

panelTitleBar.BackColor = Color.DodgerBlue;

currentButton = null;

close.Visible = false;

}

private void label3\_Click(object sender, EventArgs e)

{

if (label3.Text == "Login")

{

}

else

{

MessageBox.Show("Please Logout Current User");

}

}

private void label3\_Enter(object sender, EventArgs e)

{

label3.ForeColor = Color.Blue;

}

private void button1\_Click(object sender, EventArgs e)

{

if (label3.Text != "Login")

{

if (activeForm != null)

{

activeForm.Close();

}

Reset();

con.Close();

con.Open();

string state = "N";

string s = "update Login set status='" + state.ToString() + "'";

SqlCommand com = new SqlCommand(s, con);

com.ExecuteNonQuery();

con.Close();

label3.Text = "Login";

}

else

{

MessageBox.Show("Please Login");

}

}

private void panelTitleBar\_MouseDown(object sender, MouseEventArgs e)

{

ReleaseCapture();

SendMessage(this.Handle,0x112,0xf012,0);

}

private void button2\_Click\_1(object sender, EventArgs e)

{

DialogResult dialog = MessageBox.Show("Do you want Logout&Close this application", "Exit", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (dialog == DialogResult.Yes)

{

if (activeForm != null)

{

activeForm.Close();

}

Reset();

con.Close();

con.Open();

string state = "N";

string s = "update Login set status='" + state.ToString() + "'";

SqlCommand com = new SqlCommand(s, con);

com.ExecuteNonQuery();

con.Close();

Application.Exit();

}

else if (dialog == DialogResult.No)

{

}

}

private void button8\_Click(object sender, EventArgs e)

{

if (WindowState == FormWindowState.Normal)

{

this.WindowState = FormWindowState.Maximized;

}

else

{

this.WindowState = FormWindowState.Normal;

}

}

private void button9\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

private void button7\_Click(object sender, EventArgs e)

{

OpenChildForm(new Dashboard(), sender);

}

private void button10\_Click(object sender, EventArgs e)

{

if (label3.Text == "Login")

{

OpenChildForm(new Login(), sender);

}

else

{

MessageBox.Show("Please Logout Current User");

}

}

private void button11\_Click(object sender, EventArgs e)

{

string ste = "Y";

con.Close();

string querry = "select \* from Login where status='" + ste + "'";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

label3.Text = dr["name"].ToString();

}

dr.Close();

con.Close();

}

}

}

**Beneficiaries :**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using Emgu.CV;

using Emgu.CV.Structure;

using Emgu.CV.CvEnum;

using System.IO;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Diagnostics;

using System.Configuration;

namespace MultiFaceRec

{

public partial class Members : Form

{

Image<Bgr, Byte> currentFrame;

Capture grabber;

HaarCascade face;

string finalname;

HaarCascade eye;

MCvFont font = new MCvFont(FONT.CV\_FONT\_HERSHEY\_TRIPLEX, 0.5d, 0.5d);

Image<Gray, byte> result, TrainedFace = null;

Image<Gray, byte> gray = null;

List<Image<Gray, byte>> trainingImages = new List<Image<Gray, byte>>();

List<string> labels = new List<string>();

List<string> NamePersons = new List<string>();

int ContTrain, NumLabels, t;

string name, names = null;

SqlConnection con;

SqlCommand com;

public Members()

{

InitializeComponent();

ConnectionStringSettings conc = ConfigurationManager.ConnectionStrings["connect"];

string constr = conc.ConnectionString;

con = new SqlConnection(constr);

//Load haarcascades for face detection

face = new HaarCascade("haarcascade\_frontalface\_default.xml");

//eye = new HaarCascade("haarcascade\_eye.xml");

try

{

//Load of previus trainned faces and labels for each image

string Labelsinfo = File.ReadAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt");

string[] Labels = Labelsinfo.Split('%');

NumLabels = Convert.ToInt16(Labels[0]);

ContTrain = NumLabels;

string LoadFaces;

for (int tf = 1; tf < NumLabels + 1; tf++)

{

LoadFaces = "face" + tf + ".bmp";

trainingImages.Add(new Image<Gray, byte>(Application.StartupPath + "/TrainedFaces/" + LoadFaces));

labels.Add(Labels[tf]);

}

}

catch (Exception e)

{

//MessageBox.Show(e.ToString());

MessageBox.Show("Nothing in binary database, please add at least a face(Simply train the prototype with the Add Face Button).", "Triained faces load", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void Members\_Load(object sender, EventArgs e)

{

radioButton1.Checked = true;

radioButton4.Checked = true;

button4.Visible = false;

LoadTheme();

}

private void LoadTheme()

{

foreach (Control btns in this.Controls)

{

if (btns.GetType() == typeof(Button))

{

Button btn = (Button)btns;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

btn.Font = new System.Drawing.Font("Monotype Corsiva", 14.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox7.Text = "";

textBox8.Text = "";

textBox9.Text = "";

textBox10.Text = "";

comboBox1.Text = "";

comboBox2.Text = "";

comboBox3.Text = "";

} private void button2\_Click(object sender, EventArgs e)

{

con.Close();

if (radioButton1.Checked == true)

{

if (radioButton4.Checked == true)

{

if (((textBox1.Text != "" && textBox2.Text != "") && (comboBox3.Text != "" && textBox3.Text != "")) && ((comboBox3.Text != "" && textBox7.Text != "") && (textBox6.Text != "" && textBox4.Text != "")))

{

con.Open();

string querry = "INSERT INTO vaccineD(Aadhar,name,age,gender,mobile,bref,vaccinename,fdose,nxtdose,fvaccinateby,fvaccinateat,fbatch) VALUES('" + textBox1.Text + "','" + textBox2.Text + "','" + dateTimePicker1.Text + "','" + comboBox3.Text + "','" + textBox3.Text + "','" + textBox5.Text + "','" + comboBox2.Text + "','" + dateTimePicker2.Text + "','" + dateTimePicker3.Text + "','" + textBox6.Text + "','" + textBox4.Text + "','" + textBox7.Text + "')";

SqlCommand cmd = new SqlCommand(querry, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Inserted Successfully");

}

else

{

MessageBox.Show("Please Enter the Details");

}

}

else if (radioButton5.Checked == true)

{

if (((textBox1.Text != "" && textBox2.Text != "") && (comboBox3.Text != "" && textBox3.Text != "")) && ((comboBox3.Text != "" && textBox7.Text != "") && (textBox6.Text != "" && textBox4.Text != "")))

{

if ((textBox10.Text != "" && textBox9.Text != "") && (textBox8.Text != ""))

{

con.Open();

string querry = "INSERT INTO vaccineD(Aadhar,name,age,gender,mobile,bref,vaccinename,fdose,nxtdose,fvaccinateby,fvaccinateat,fbatch,sdose,svaccinateby,svaccinateat,sbatch) VALUES('" + textBox1.Text + "','" + textBox2.Text + "','" + dateTimePicker1.Text + "','" + comboBox3.Text + "','" + textBox3.Text + "','" + textBox5.Text + "','" + comboBox2.Text + "','" + dateTimePicker2.Text + "','" + dateTimePicker3.Text + "','" + textBox6.Text + "','" + textBox4.Text + "','" + textBox7.Text + "','" + dateTimePicker4.Text + "','" + textBox10.Text + "','" + textBox9.Text + "','" + textBox8.Text + "')";

SqlCommand cmd = new SqlCommand(querry, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Inserted Successfully");

}

else

{

MessageBox.Show("Please Enter the Second Dose Details");

}

}

else

{

MessageBox.Show("Please Enter the Details");

}

}

else

{

MessageBox.Show("Please Select Dose");

}

}

else if(radioButton2.Checked==true)

{

if (radioButton4.Checked == true)

{

if ((textBox7.Text != "" && textBox2.Text != "") && (comboBox3.Text != "" && textBox3.Text != ""))

{ con.Open();

string querry = "update vaccineD set name= '" + textBox2.Text + "',age='" + dateTimePicker1.Text + "',gender='" + comboBox3.Text + "',mobile='" + textBox3.Text + "',bref='" + textBox5.Text + "',vaccinename='" + comboBox2.Text + "',fdose='" + dateTimePicker2.Text + "',nxtdose='" + dateTimePicker3.Text + "',fvaccinateby='" + textBox6.Text + "',fvaccinateat='" + textBox4.Text + "',fbatch='" + textBox7.Text + "' where Aadhar='"+comboBox1.SelectedItem+"'";

SqlCommand cmd = new SqlCommand(querry, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Updated Successfully");

}

else

{

MessageBox.Show("Please Enter personal and fdose the Details");

}

}

else if (radioButton5.Checked == true)

{

if ((comboBox3.Text != "" && textBox7.Text != "") && (textBox6.Text != "" && textBox4.Text != ""))

{

if ((textBox10.Text != "" && textBox9.Text != "") && (textBox8.Text != ""))

{

con.Open();

string querry = "update vaccineD set name= '" + textBox2.Text + "',age='" + dateTimePicker1.Text + "',gender='" + comboBox3.Text + "',mobile='" + textBox3.Text + "',bref='" + textBox5.Text + "',vaccinename='" + comboBox2.Text + "',fdose='" + dateTimePicker2.Text + "',nxtdose='" + dateTimePicker3.Text + "',fvaccinateby='" + textBox6.Text + "',fvaccinateat='" + textBox4.Text + "',fbatch='" + textBox7.Text + "',sdose='" + dateTimePicker4.Text + "',svaccinateby='" + textBox10.Text + "',svaccinateat='" + textBox9.Text + "',sbatch='" + textBox8.Text + "' where Aadhar='" + comboBox1.SelectedItem + "'";

SqlCommand cmd = new SqlCommand(querry, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Updated Successfully");

}

else

{

MessageBox.Show("Please Enter personal and Second Dose Details");

}

}

else

{

MessageBox.Show("Please Enter the Details");

}

}

else

{

MessageBox.Show("Please Select Dose");

}

}

else if (radioButton3.Checked == true)

{

if (comboBox1.Text!="")

{

con.Open();

//string state = "N";

string s = "delete from vaccineD where Aadhar='" + comboBox1.SelectedItem + "'";

SqlCommand cmd = new SqlCommand(s, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Deleted Successfully");

comboBox1.Items.Remove(comboBox1.SelectedItem);

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox7.Text = "";

textBox8.Text = "";

textBox9.Text = "";

textBox10.Text = "";

comboBox1.Text = "";

comboBox2.Text = "";

comboBox3.Text = "";

}

else

{

MessageBox.Show("Please fill the currect details");

}

}

else

{

MessageBox.Show("Please select actions");

}

}

private void button5\_Click(object sender, EventArgs e)

{

label15.Visible = true;

label14.Visible = true;

//Initialize the capture device

grabber = new Capture();

grabber.QueryFrame();

//Initialize the FrameGraber event

Application.Idle += new EventHandler(FrameGrabber);

button1.Enabled = false;

}

private void button6\_Click(object sender, EventArgs e)

{

try

{

if (textBox2.Text != "")

{

//Trained face counter

ContTrain = ContTrain + 1;

//Get a gray frame from capture device

gray = grabber.QueryGrayFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//Face Detector

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

//Action for each element detected

foreach (MCvAvgComp f in facesDetected[0])

{

TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();

break;

}

//resize face detected image for force to compare the same size with the

//test image with cubic interpolation type method

TrainedFace = result.Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

trainingImages.Add(TrainedFace);

labels.Add(textBox2.Text);

//Show face added in gray scale

imageBox1.Image = TrainedFace;

//Write the number of triained faces in a file text for further load

File.WriteAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", trainingImages.ToArray().Length.ToString() + "%");

//Write the labels of triained faces in a file text for further load

for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)

{

trainingImages.ToArray()[i - 1].Save(Application.StartupPath + "/TrainedFaces/face" + i + ".bmp");

File.AppendAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");

}

MessageBox.Show(comboBox1.SelectedItem.ToString() + "´s face detected and added :)", "Training OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Please Enter the Name");

}

}

catch

{

MessageBox.Show("Enable the face detection first", "Training Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

void FrameGrabber(object sender, EventArgs e)

{

label14.Text = "0";

NamePersons.Add("");

//Get the current frame form capture device

currentFrame = grabber.QueryFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//Convert it to Grayscale

gray = currentFrame.Convert<Gray, Byte>();

//Face Detector

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

//Action for each element detected

foreach (MCvAvgComp f in facesDetected[0])

{

t = t + 1;

result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//draw the face detected in the 0th (gray) channel with blue color

currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);

if (trainingImages.ToArray().Length != 0)

{

//TermCriteria for face recognition with numbers of trained images like maxIteration

MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

//Eigen face recognizer

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(

trainingImages.ToArray(),

labels.ToArray(),

3000,

ref termCrit);

name = recognizer.Recognize(result);

finalname = name;

//Draw the label for each face detected and recognized

currentFrame.Draw(name, ref font, new Point(f.rect.X - 2, f.rect.Y - 2), new Bgr(Color.LightGreen));

}

NamePersons[t - 1] = name;

NamePersons.Add("");

//Set the number of faces detected on the scene

label14.Text = facesDetected[0].Length.ToString();

}

t = 0;

//Names concatenation of persons recognized

for (int nnn = 0; nnn < facesDetected[0].Length; nnn++)

{

names = names + NamePersons[nnn] + ", ";

}

//Show the faces procesed and recognized

imageBoxFrameGrabber.Image = currentFrame;

label13.Text = names;

//label4.Text = "Vaccinated";

names = "";

//Clear the list(vector) of names

NamePersons.Clear();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox7.Text = "";

textBox8.Text = "";

textBox9.Text = "";

textBox10.Text = "";

comboBox1.Text = "";

comboBox2.Text = "";

comboBox3.Text = "";

//comboBox4.Text = "";

button2.Text = "Insert";

textBox1.Visible = true;

textBox1.Enabled = true;

comboBox1.Visible = false;

dateTimePicker2.Enabled = true;

dateTimePicker3.Enabled = true;

textBox4.Enabled = true;

button5.Enabled = true;

button6.Enabled = true;

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

comboBox1.Items.Clear();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox7.Text = "";

textBox8.Text = "";

textBox9.Text = "";

textBox10.Text = "";

//textBox15.Text = "";

comboBox1.Text = "";

comboBox2.Text = "";

comboBox3.Text = "";

//comboBox4.Text = "";

comboBox1.Visible = true;

groupBox5.Enabled = true;

groupBox6.Enabled = true;

radioButton5.Checked = true;

textBox2.Enabled = true;

textBox3.Enabled = true;

comboBox3.Enabled = true;

dateTimePicker1.Enabled = true;

dateTimePicker2.Enabled = true;

dateTimePicker3.Enabled = true;

textBox5.Enabled = true;

textBox4.Enabled = true;

button2.Text = "Update";

string querry = " select \* from vaccineD";

con.Close();

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr["Aadhar"].ToString());

}

dr.Close();

con.Close();

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

comboBox1.Items.Clear();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox7.Text = "";

textBox8.Text = "";

textBox9.Text = "";

textBox10.Text = "";

comboBox1.Text = "";

comboBox2.Text = "";

comboBox3.Text = "";

groupBox5.Enabled = false;

groupBox6.Enabled = false;

radioButton5.Checked = true;

button2.Text = "Delete";

textBox2.Enabled = false;

textBox3.Enabled = false;

comboBox3.Enabled = false;

dateTimePicker1.Enabled = false;

textBox5.Enabled=false;

textBox1.Visible = false;

comboBox1.Visible = true;

dateTimePicker2.Enabled = false;

dateTimePicker3.Enabled = false;

textBox4.Enabled = false;

button5.Enabled = false;

button6.Enabled = false;

string querry = " select \* from vaccineD";

con.Close();

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr["Aadhar"].ToString());

}

dr.Close();

con.Close();

}

private void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

dateTimePicker3.Text = "" + dateTimePicker2.Value.Date.AddDays(84);

}

private void radioButton5\_CheckedChanged(object sender, EventArgs e)

{

groupBox7.Visible = true;

groupBox8.Visible = true;

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

groupBox7.Visible = true;

groupBox8.Visible = false;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string querry = "select \* from vaccineD where Aadhar='" + comboBox1.Text + "'";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

textBox2.Text = dr["name"].ToString();

dateTimePicker1.Text = dr["age"].ToString();

comboBox3.Text = dr["gender"].ToString();

textBox3.Text = dr["mobile"].ToString();

textBox5.Text = dr["bref"].ToString();

comboBox3.Text = dr["gender"].ToString();

comboBox2.Text = dr["vaccinename"].ToString();

dateTimePicker2.Text = dr["fdose"].ToString();

dateTimePicker3.Text = dr["nxtdose"].ToString();

textBox6.Text = dr["fvaccinateby"].ToString();

textBox4.Text = dr["fvaccinateat"].ToString();

textBox7.Text = dr["fbatch"].ToString();

dateTimePicker4.Text = dr["sdose"].ToString();

textBox10.Text = dr["svaccinateby"].ToString();

textBox9.Text = dr["svaccinateat"].ToString();

textBox8.Text = dr["sbatch"].ToString();

}

dr.Close();

con.Close();

}

}

}

**Admin :**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace MultiFaceRec.Forms

{

public partial class Vaccinated : Form

{

SqlConnection con;

SqlCommand com;

public Vaccinated()

{

InitializeComponent();

ConnectionStringSettings conc = ConfigurationManager.ConnectionStrings["connect"];

string constr = conc.ConnectionString;

con = new SqlConnection(constr);

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox5.Clear();

textBox11.Clear();

comboBox3.Text = "";

comboBox1.Text = "";

textBox2.Enabled = true;

textBox3.Enabled = true;

textBox5.Enabled = true;

textBox11.Enabled = true;

comboBox3.Enabled = true;

//textBox1.Visible = false;

textBox2.Visible = true;

textBox3.Visible = true;

textBox5.Visible = true;

textBox11.Visible = true;

comboBox3.Visible = true;

//comboBox1.Visible = true;

button1.Text = "Create Account";

comboBox1.Visible = false;

textBox1.Visible = true;

}

private void Vaccinated\_Load(object sender, EventArgs e)

{

radioButton1.Checked = true;

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox5.Clear();

textBox11.Clear();

comboBox3.Text = "";

comboBox1.Text = "";

comboBox1.Text = "";

comboBox1.Items.Clear();

string querry = "select aadhar from Login";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr["aadhar"].ToString());

}

dr.Close();

con.Close();

button1.Text = "Update Account";

textBox1.Visible = false;

comboBox1.Visible = true;

textBox2.Visible = true;

textBox3.Visible = true;

textBox5.Visible = true;

textBox11.Visible = true;

comboBox3.Visible = true;

textBox2.Enabled = true;

textBox3.Enabled = true;

textBox5.Enabled = true;

textBox11.Enabled = true;

comboBox3.Enabled = true;

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox5.Clear();

textBox11.Clear();

comboBox3.Text = "";

comboBox1.Text = "";

textBox1.Visible = false;

textBox2.Enabled = false;

textBox3.Enabled = false;

textBox5.Enabled = false;

textBox11.Enabled = false;

comboBox3.Enabled = false;

comboBox1.Visible = true;

comboBox1.Text = "";

comboBox1.Items.Clear();

string querry = "select aadhar from Login";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr["aadhar"].ToString());

}

dr.Close();

con.Close();

button1.Text = "Delete Account";

textBox1.Visible = false;

comboBox1.Visible = true;

}

private void button1\_Click(object sender, EventArgs e)

{

if (radioButton1.Checked == true)

{

if ((textBox1.Text != "") && (textBox3.Text != ""))

{

if((textBox2.Text != "") && (textBox5.Text != ""))

{

con.Open();

string status="N";

string querry = "insert into Login(aadhar,name,type,mobile,uname,password,status) values('"+textBox1.Text+"','"+textBox3.Text+"','"+comboBox3.Text+"','"+textBox11.Text+"','"+textBox2.Text+"','"+textBox5.Text+"','"+status.ToString()+"')";

SqlCommand cmd = new SqlCommand(querry, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Insert Successfully");

}

else{

MessageBox.Show("Please Fill Username & Password");

}

}

else

{

MessageBox.Show("Please Fill The Details");

}

}

else if (radioButton2.Checked == true)

{

if ((comboBox1.Text != "") && (textBox3.Text != ""))

{

if ((textBox2.Text != "") && (textBox5.Text != ""))

{

con.Open();

string state = "N";

string s="update Login set aadhar='"+comboBox1.Text+"',name='"+textBox3.Text+"',type='"+comboBox3.Text+"',mobile='"+textBox11.Text+"',uname='"+textBox2.Text+"',password='"+textBox5.Text+"',status='"+state.ToString()+"' where aadhar='"+comboBox1.Text+"'";

SqlCommand cmd = new SqlCommand(s, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Update Successfully");

}

else

{

MessageBox.Show("Please Fill Username & Password");

}

}

else

{

MessageBox.Show("Please Select The Details");

}

}

else if (radioButton3.Checked == true)

{

if (comboBox1.Text != "")

{

con.Open();

//string state = "N";

string s = "delete from Login where aadhar='" + comboBox1.SelectedItem + "'";

SqlCommand cmd = new SqlCommand(s, con);

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Deleted Successfully");

comboBox1.Items.Remove(comboBox1.SelectedItem);

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox5.Clear();

textBox11.Clear();

comboBox3.Text = "";

comboBox1.Text = "";

}

else

{

MessageBox.Show("Please Select The Aadhar No");

}

}

else

{

MessageBox.Show("Please Select Action");

}

}

private void button2\_Click(object sender, EventArgs e)

{

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox5.Clear();

textBox11.Clear();

comboBox3.Text = "";

comboBox1.Text = "";

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string querry = "select \* from Login where aadhar='"+comboBox1.Text+"'";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

// comboBox1.Items.Add(dr["aadhar"].ToString());

textBox3.Text=dr["name"].ToString();

comboBox3.Text=dr["type"].ToString();

textBox11.Text=dr["mobile"].ToString();

textBox2.Text = dr["uname"].ToString();

textBox5.Text = dr["password"].ToString();

}

dr.Close();

con.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

//con.Open();

SqlDataAdapter da = new SqlDataAdapter("SELECT \* FROM Login", con);

DataSet ds = new DataSet();

da.Fill(ds, "Login");

dataGridView1.DataSource = ds.Tables["Login"].DefaultView;

}

}

}

**Verify :**

/Multiple face detection and recognition in real time

//Using EmguCV cross platform .Net wrapper to the Intel OpenCV image processing library for C#.Net

//Writed by Sergio Andrés Guitérrez Rojas

//"Serg3ant" for the delveloper comunity

// Sergiogut1805@hotmail.com

//Regards from Bucaramanga-Colombia ;)

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using Emgu.CV;

using Emgu.CV.Structure;

using Emgu.CV.CvEnum;

using System.IO;

using System.Diagnostics;

using System.Data.SqlClient;

using System.Configuration;

namespace MultiFaceRec

{

public partial class FrmPrincipal : Form

{

//Declararation of all variables, vectors and haarcascades

Image<Bgr, Byte> currentFrame;

Capture grabber;

HaarCascade face;

string finalname;

HaarCascade eye;

MCvFont font = new MCvFont(FONT.CV\_FONT\_HERSHEY\_TRIPLEX, 0.5d, 0.5d);

Image<Gray, byte> result, TrainedFace = null;

Image<Gray, byte> gray = null;

List<Image<Gray, byte>> trainingImages = new List<Image<Gray, byte>>();

List<string> labels= new List<string>();

List<string> NamePersons = new List<string>();

int ContTrain, NumLabels, t;

string name, names = null;

SqlConnection con;

SqlCommand com;

public FrmPrincipal()

{

InitializeComponent();

con = new SqlConnection();

com = new SqlCommand();

con.ConnectionString = "server=(local);Initial Catalog=vac;Integrated Security=True";

com.Connection = con;

//Load haarcascades for face detection

face = new HaarCascade("haarcascade\_frontalface\_default.xml");

//eye = new HaarCascade("haarcascade\_eye.xml");

try

{

//Load of previus trainned faces and labels for each image

string Labelsinfo = File.ReadAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt");

string[] Labels = Labelsinfo.Split('%');

NumLabels = Convert.ToInt16(Labels[0]);

ContTrain = NumLabels;

string LoadFaces;

for (int tf = 1; tf < NumLabels+1; tf++)

{

LoadFaces = "face" + tf + ".bmp";

trainingImages.Add(new Image<Gray, byte>(Application.StartupPath + "/TrainedFaces/" + LoadFaces));

labels.Add(Labels[tf]);

}

}

catch(Exception e)

{

//MessageBox.Show(e.ToString());

MessageBox.Show("Nothing in binary database, please add at least a face(Simply train the prototype with the Add Face Button).", "Triained faces load", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void button1\_Click(object sender, EventArgs e)

{

//Initialize the capture device

grabber = new Capture();

grabber.QueryFrame();

//Initialize the FrameGraber event

Application.Idle += new EventHandler(FrameGrabber);

button1.Enabled = false;

}

private void button2\_Click(object sender, System.EventArgs e)

{

try

{

//Trained face counter

ContTrain = ContTrain + 1;

//Get a gray frame from capture device

gray = grabber.QueryGrayFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//Face Detector

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

//Action for each element detected

foreach (MCvAvgComp f in facesDetected[0])

{

TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();

break;

}

//resize face detected image for force to compare the same size with the

//test image with cubic interpolation type method

TrainedFace = result.Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

trainingImages.Add(TrainedFace);

labels.Add(comboBox1.SelectedItem.ToString());

//Show face added in gray scale

imageBox1.Image = TrainedFace;

//Write the number of triained faces in a file text for further load

File.WriteAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", trainingImages.ToArray().Length.ToString() + "%");

//Write the labels of triained faces in a file text for further load

for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)

{

trainingImages.ToArray()[i - 1].Save(Application.StartupPath + "/TrainedFaces/face" + i + ".bmp");

File.AppendAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");

}

MessageBox.Show(comboBox1.SelectedItem.ToString() + "´s face detected and added :)", "Training OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch

{

MessageBox.Show("Enable the face detection first", "Training Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

void FrameGrabber(object sender, EventArgs e)

{

label3.Text = "0";

//label4.Text = "";

NamePersons.Add("");

//Get the current frame form capture device

currentFrame = grabber.QueryFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//Convert it to Grayscale

gray = currentFrame.Convert<Gray, Byte>();

//Face Detector

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

//Action for each element detected

foreach (MCvAvgComp f in facesDetected[0])

{

t = t + 1;

result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//draw the face detected in the 0th (gray) channel with blue color

currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);

if (trainingImages.ToArray().Length != 0)

{

//TermCriteria for face recognition with numbers of trained images like maxIteration

MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

//Eigen face recognizer

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(

trainingImages.ToArray(),

labels.ToArray(),

3000,

ref termCrit);

name = recognizer.Recognize(result);

// MessageBox.Show("" + name);

// textBox2.Text = name;

finalname = name;

//Draw the label for each face detected and recognized

currentFrame.Draw(name, ref font, new Point(f.rect.X - 2, f.rect.Y - 2), new Bgr(Color.LightGreen));

}

NamePersons[t-1] = name;

NamePersons.Add("");

//Set the number of faces detected on the scene

label3.Text = facesDetected[0].Length.ToString();

}

t = 0;

//Names concatenation of persons recognized

for (int nnn = 0; nnn < facesDetected[0].Length; nnn++)

{

names = names + NamePersons[nnn] + ", ";

label9.ForeColor = Color.Black;

label9.Text = "No";

if (label4.Text!= ", ")

{

label9.ForeColor = Color.Green;

label9.Text = "Vaccinated";

button4.Enabled = true;

}

else if (label4.Text == ", ")

{

label9.ForeColor = Color.Red;

label9.Text = "Not Vaccinated";

button4.Enabled = false;

}

else

{

label9.Text = "No";

}

}

//label9.Text = "No";

//Show the faces procesed and recognized

imageBoxFrameGrabber.Image = currentFrame;

label4.Text = names;

//label4.Text = "Vaccinated";

//label9.Text = "";

names = "";

//Clear the list(vector) of names

NamePersons.Clear();

}

private void button4\_Click(object sender, EventArgs e)

{

textBox2.Text = finalname;

if (textBox2.Text != "")

{

if (label9.Text != "No")

{

ViewV vv = new ViewV(this);

vv.ShowDialog();

}

else if(label9.Text == "No")

{

MessageBox.Show("No Data");

}

}

else

{

MessageBox.Show("No Data Detection");

}

}

private void LoadTheme()

{

foreach (Control btns in this.Controls)

{

if (btns.GetType() == typeof(Button))

{

Button btn = (Button)btns;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

btn.Font = new System.Drawing.Font("Monotype Corsiva", 14.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

}

}

}

private void FrmPrincipal\_Load(object sender, EventArgs e)

{

LoadTheme();

label9.Visible = true;

label10.Visible = true;

label6.Visible = false;

label7.Visible = false;

label8.Visible = false;

textBox2.Visible = false;

textBox3.Visible = false;

textBox4.Visible = false;

button3.Visible = false;

button2.Visible = false;

groupBox1.Visible = false;

comboBox1.Visible = false;

ConnectionStringSettings conc = ConfigurationManager.ConnectionStrings["connect"];

string constr = conc.ConnectionString;

con = new SqlConnection(constr);

}

private void label4\_TextChanged(object sender, EventArgs e)

{

if (label4.Text == "")

{

label9.ForeColor = Color.Black;

label9.Text = "No";

}

else if (label4.Text == ", ")

{

label9.ForeColor = Color.Black;

label9.Text = "No";

}

}

}

}

**ViewRecord :**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

namespace MultiFaceRec

{

public partial class ViewV : Form

{

SqlConnection con;

SqlDataAdapter da;

FrmPrincipal fp;

public ViewV(FrmPrincipal verify)

{

InitializeComponent();

this.fp = verify;

ConnectionStringSettings conc = ConfigurationManager.ConnectionStrings["connect"];

string constr = conc.ConnectionString;

con = new SqlConnection(constr);

}

private void ViewV\_Load(object sender, EventArgs e)

{

checkBox1.Checked = false;

label2.Visible = false;

textBox2.Visible = false;

button1.Size.Height.ToString("31");

button1.Size.Width.ToString("74");

textBox1.Visible = true;

button1.Visible = true;

//string Aadharname;

//string name = "kabila";

//Aadharname = fp.textBox2.Text;

textBox1.Text = fp.textBox2.Text;

// fp.textBox2.Text;

MessageBox.Show("" + textBox1.Text);

con.Close();

//con.Open();

SqlCommand cmd = new SqlCommand("select \* from vaccineD where name='" + textBox1.Text + "'", con);

SqlDataAdapter da = new SqlDataAdapter();

//da.SelectCommand=cmd

da.SelectCommand = cmd;

DataTable dt = new DataTable();

da.Fill(dt);

BindingSource bsource = new BindingSource();

bsource.DataSource = dt;

dataGridView1.DataSource = bsource;

con.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

if ((checkBox1.Checked == false)&&(textBox1.Text!=""))

{

string querry = "select \* from vaccineD where name='" + textBox1.Text + "' ";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

da = new SqlDataAdapter();

da.SelectCommand = cmd;

DataTable dt = new DataTable();

da.Fill(dt);

dataGridView1.DataSource = dt;

con.Close();

}

else if ((checkBox1.Checked == true) && (textBox1.Text != "") && (textBox2.Text != ""))

{

string querry = "select \* from vaccineD where name='" + textBox1.Text + "' and Aadhar='" + textBox2.Text + "'";

SqlCommand cmd = new SqlCommand(querry, con);

//SqlCommand cmd = new SqlCommand("select \* from vaccineD where name='" + textBox1.Text + "'", con);

con.Open();

da = new SqlDataAdapter();

da.SelectCommand = cmd;

DataTable dt = new DataTable();

da.Fill(dt);

dataGridView1.DataSource = dt;

con.Close();

}

else

{

MessageBox.Show("Please Enter The Name or Aadhar no");

}

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked == true)

{

label2.Visible = true;

textBox2.Visible = true;

button1.Size.Height.ToString("31");

button1.Size.Width.ToString("74");

}

else

{

label2.Visible = false;

textBox2.Visible = false;

button1.Size.Height.ToString("57");

button1.Size.Width.ToString("74");

}

}

}

}

**View:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

namespace MultiFaceRec

{

public partial class Viw : Form

{

SqlConnection con;

public Viw()

{

InitializeComponent();

ConnectionStringSettings conc = ConfigurationManager.ConnectionStrings["connect"];

string constr = conc.ConnectionString;

con = new SqlConnection(constr);

}

private void LoadTheme()

{

foreach (Control btns in this.Controls)

{

if (btns.GetType() == typeof(Button))

{

Button btn = (Button)btns;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

btn.Font = new System.Drawing.Font("Monotype Corsiva", 14.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

}

}

//label1.ForeColor = ThemeColor.SecondaryColor;

}

private void Viw\_Load(object sender, EventArgs e)

{

LoadTheme();

groupBox4.Enabled = false;

groupBox5.Enabled = false;

}

private void button2\_Click(object sender, EventArgs e)

{

if (textBox7.Text != "" && textBox8.Text == "")

{

//fdose report

Report frpt = new Report();

Fdose fd=new Fdose();

TextObject Name = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["NameText"];

TextObject dob = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["AgeText"];

TextObject Gender = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["GenderText"];

TextObject Aadhar = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["AadharText"];

TextObject Bref = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["brefText"];

TextObject Vaccinename = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["VaccineText"];

TextObject Fdose = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["FdoseText"];

TextObject Nxtdose = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["NxtText"];

TextObject Fvaccinatedby = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["FvaccinatedbyText"];

TextObject Fvaccinatedat = (TextObject)fd.ReportDefinition.Sections["Section3"].ReportObjects["FvaccinatedatText"];

string aAA= textBox2.Text;

string aadhar = aAA.Substring(aAA.Length - 4);

Name.Text = textBox3.Text;

dob.Text = dateTimePicker1.Text;

Gender.Text = comboBox3.Text;

Aadhar.Text = aadhar.ToString();

Bref.Text = textBox5.Text;

Vaccinename.Text = comboBox2.Text;

Fdose.Text = dateTimePicker2.Text;

Nxtdose.Text = dateTimePicker3.Text;

Fvaccinatedby.Text = textBox1.Text;

Fvaccinatedat.Text = textBox4.Text;

frpt.crystalReportViewer1.ReportSource = fd;

frpt.ShowDialog();

}

else if (textBox8.Text != "" && textBox7.Text != "")

{

//sdose report

Report srpt = new Report();

Sdose sd = new Sdose();

TextObject Name = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["NameText"];

TextObject dob = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["AgeText"];

TextObject Gender = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["GenderText"];

TextObject Aadhar = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["AadharText"];

TextObject Bref = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["brefText"];

TextObject Vaccinename = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["VaccineText"];

TextObject Fdose = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["FdoseText"];

TextObject Fbatch = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["FbatchText"];

TextObject Sdose = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["SdoseText"];

TextObject Svaccinatedby = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["SvaccinatedbyText"];

TextObject Svaccinatedat = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["SvaccinatedatText"];

TextObject Sbatch = (TextObject)sd.ReportDefinition.Sections["Section3"].ReportObjects["SbatchText"];

//aadhar

string aa = textBox2.Text;

string aadhar = aa.Substring(aa.Length - 4);

Name.Text = textBox3.Text;

dob.Text = dateTimePicker1.Text;

Gender.Text = comboBox3.Text;

Aadhar.Text = aadhar.ToString();

Bref.Text = textBox5.Text;

Vaccinename.Text = comboBox2.Text;

Fdose.Text = dateTimePicker2.Text;

Fbatch.Text = textBox7.Text;

Sdose.Text = dateTimePicker4.Text;

Sbatch.Text = textBox8.Text;

Svaccinatedby.Text = textBox10.Text;

Svaccinatedat.Text = textBox9.Text;

srpt.crystalReportViewer1.ReportSource = sd;

srpt.ShowDialog();

}

else

{

MessageBox.Show("Please Enter The Aadhar No");

}

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

string querry = "select \* from vaccineD where Aadhar='" + textBox6.Text + "' or name='" + textBox6.Text + "' ";

SqlCommand cmd = new SqlCommand(querry, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

// comboBox1.Items.Add(dr["aadhar"].ToString());

textBox2.Text = dr["Aadhar"].ToString();

textBox3.Text = dr["name"].ToString();

dateTimePicker1.Text = dr["age"].ToString();

comboBox3.Text = dr["gender"].ToString();

textBox11.Text = dr["mobile"].ToString();

textBox5.Text = dr["bref"].ToString();

//comboBox3.Text = dr["gender"].ToString();

comboBox2.Text = dr["vaccinename"].ToString();

dateTimePicker2.Text = dr["fdose"].ToString();

dateTimePicker3.Text = dr["nxtdose"].ToString();

textBox1.Text = dr["fvaccinateby"].ToString();

textBox4.Text = dr["fvaccinateat"].ToString();

textBox7.Text = dr["fbatch"].ToString();

dateTimePicker4.Text = dr["sdose"].ToString();

textBox10.Text = dr["svaccinateby"].ToString();

textBox9.Text = dr["svaccinateat"].ToString();

textBox8.Text = dr["sbatch"].ToString();

}

dr.Close();

con.Close();

}

catch (Exception ex)

{

MessageBox.Show(""+ex);

}

if (textBox7.Text != "" && textBox8.Text == "")

{

radioButton4.Checked = true;

}

else if (textBox8.Text != "" && textBox7.Text != "")

{

radioButton5.Checked = true;

}

else

{

MessageBox.Show("Please Contact Admin");

}

}

}

}

**appconfig:**

<?xml version="1.0"?>

<configuration>

<configSections>

</configSections>

<connectionStrings>

<add name="connect"

connectionString="Data Source=(local);Initial Catalog=vac;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup></configuration>

**Themecolor :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

namespace MultiFaceRec

{

public static class ThemeColor

{

public static Color PrimaryColor { get; set; }

public static Color SecondaryColor { get; set; }

public static List<string> ColorList = new List<string>() { "#3F51B5",

"#009688",

"#FF5722",

"#607D8B",

"#FF9800",

"#9C27B0",

"#2196F3",

"#EA676C",

"#E41A4A",

"#5978BB",

"#018790",

"#0E3441",

"#00B0AD",

"#721D47",

"#EA4833",

"#EF937E",

"#F37521",

"#A12059",

"#126881",

"#8BC240",

"#364D5B",

"#C7DC5B",

"#0094BC",

"#E4126B",

"#43B76E",

"#7BCFE9",

"#B71C46"};

public static Color ChangeColorBrightness(Color color, double correctionFactor)

{

double red = color.R;

double green = color.G;

double blue = color.B;

//If correction factor is less than 0, darken color.

if (correctionFactor < 0)

{

correctionFactor = 1 + correctionFactor;

red \*= correctionFactor;

green \*= correctionFactor;

blue \*= correctionFactor;

}

//If correction factor is greater than zero, lighten color.

else

{

red = (255 - red) \* correctionFactor + red;

green = (255 - green) \* correctionFactor + green;

blue = (255 - blue) \* correctionFactor + blue;

}

return Color.FromArgb(color.A, (byte)red, (byte)green, (byte)blue);

}

}

}

**EigenObjectRecognizer :**

using System;

using System.Diagnostics;

using Emgu.CV.Structure;

namespace Emgu.CV

{

/// <summary>

/// An object recognizer using PCA (Principle Components Analysis)

/// </summary>

[Serializable]

public class EigenObjectRecognizer

{

private Image<Gray, Single>[] \_eigenImages;

private Image<Gray, Single> \_avgImage;

private Matrix<float>[] \_eigenValues;

private string[] \_labels;

private double \_eigenDistanceThreshold;

/// <summary>

/// Get the eigen vectors that form the eigen space

/// </summary>

/// <remarks>The set method is primary used for deserialization, do not attemps to set it unless you know what you are doing</remarks>

public Image<Gray, Single>[] EigenImages

{

get { return \_eigenImages; }

set { \_eigenImages = value; }

}

/// <summary>

/// Get or set the labels for the corresponding training image

/// </summary>

public String[] Labels

{

get { return \_labels; }

set { \_labels = value; }

}

/// <summary>

/// Get or set the eigen distance threshold.

/// The smaller the number, the more likely an examined image will be treated as unrecognized object.

/// Set it to a huge number (e.g. 5000) and the recognizer will always treated the examined image as one of the known object.

/// </summary>

public double EigenDistanceThreshold

{

get { return \_eigenDistanceThreshold; }

set { \_eigenDistanceThreshold = value; }

}

/// <summary>

/// Get the average Image.

/// </summary>

/// <remarks>The set method is primary used for deserialization, do not attemps to set it unless you know what you are doing</remarks>

public Image<Gray, Single> AverageImage

{

get { return \_avgImage; }

set { \_avgImage = value; }

}

/// <summary>

/// Get the eigen values of each of the training image

/// </summary>

/// <remarks>The set method is primary used for deserialization, do not attemps to set it unless you know what you are doing</remarks>

public Matrix<float>[] EigenValues

{

get { return \_eigenValues; }

set { \_eigenValues = value; }

}

private EigenObjectRecognizer()

{

}

/// <summary>

/// Create an object recognizer using the specific tranning data and parameters, it will always return the most similar object

/// </summary>

/// <param name="images">The images used for training, each of them should be the same size. It's recommended the images are histogram normalized</param>

/// <param name="termCrit">The criteria for recognizer training</param>

public EigenObjectRecognizer(Image<Gray, Byte>[] images, ref MCvTermCriteria termCrit)

: this(images, GenerateLabels(images.Length), ref termCrit)

{

}

private static String[] GenerateLabels(int size)

{

String[] labels = new string[size];

for (int i = 0; i < size; i++)

labels[i] = i.ToString();

return labels;

}

/// <summary>

/// Create an object recognizer using the specific tranning data and parameters, it will always return the most similar object

/// </summary>

/// <param name="images">The images used for training, each of them should be the same size. It's recommended the images are histogram normalized</param>

/// <param name="labels">The labels corresponding to the images</param>

/// <param name="termCrit">The criteria for recognizer training</param>

public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, ref MCvTermCriteria termCrit)

: this(images, labels, 0, ref termCrit)

{

}

/// <summary>

/// Create an object recognizer using the specific tranning data and parameters

/// </summary>

/// <param name="images">The images used for training, each of them should be the same size. It's recommended the images are histogram normalized</param>

/// <param name="labels">The labels corresponding to the images</param>

/// <param name="eigenDistanceThreshold">

/// The eigen distance threshold, (0, ~1000].

/// The smaller the number, the more likely an examined image will be treated as unrecognized object.

/// If the threshold is &lt; 0, the recognizer will always treated the examined image as one of the known object.

/// </param>

/// <param name="termCrit">The criteria for recognizer training</param>

public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, double eigenDistanceThreshold, ref MCvTermCriteria termCrit)

{

Debug.Assert(images.Length == labels.Length, "The number of images should equals the number of labels");

Debug.Assert(eigenDistanceThreshold >= 0.0, "Eigen-distance threshold should always >= 0.0");

CalcEigenObjects(images, ref termCrit, out \_eigenImages, out \_avgImage);

/\*

\_avgImage.SerializationCompressionRatio = 9;

foreach (Image<Gray, Single> img in \_eigenImages)

//Set the compression ration to best compression. The serialized object can therefore save spaces

img.SerializationCompressionRatio = 9;

\*/

\_eigenValues = Array.ConvertAll<Image<Gray, Byte>, Matrix<float>>(images,

delegate(Image<Gray, Byte> img)

{

return new Matrix<float>(EigenDecomposite(img, \_eigenImages, \_avgImage));

});

\_labels = labels;

\_eigenDistanceThreshold = eigenDistanceThreshold;

}

#region static methods

/// <summary>

/// Caculate the eigen images for the specific traning image

/// </summary>

/// <param name="trainingImages">The images used for training </param>

/// <param name="termCrit">The criteria for tranning</param>

/// <param name="eigenImages">The resulting eigen images</param>

/// <param name="avg">The resulting average image</param>

public static void CalcEigenObjects(Image<Gray, Byte>[] trainingImages, ref MCvTermCriteria termCrit, out Image<Gray, Single>[] eigenImages, out Image<Gray, Single> avg)

{

int width = trainingImages[0].Width;

int height = trainingImages[0].Height;

IntPtr[] inObjs = Array.ConvertAll<Image<Gray, Byte>, IntPtr>(trainingImages, delegate(Image<Gray, Byte> img) { return img.Ptr; });

if (termCrit.max\_iter <= 0 || termCrit.max\_iter > trainingImages.Length)

termCrit.max\_iter = trainingImages.Length;

int maxEigenObjs = termCrit.max\_iter;

#region initialize eigen images

eigenImages = new Image<Gray, float>[maxEigenObjs];

for (int i = 0; i < eigenImages.Length; i++)

eigenImages[i] = new Image<Gray, float>(width, height);

IntPtr[] eigObjs = Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr; });

#endregion

avg = new Image<Gray, Single>(width, height);

CvInvoke.cvCalcEigenObjects(

inObjs,

ref termCrit,

eigObjs,

null,

avg.Ptr);

}

/// <summary>

/// Decompose the image as eigen values, using the specific eigen vectors

/// </summary>

/// <param name="src">The image to be decomposed</param>

/// <param name="eigenImages">The eigen images</param>

/// <param name="avg">The average images</param>

/// <returns>Eigen values of the decomposed image</returns>

public static float[] EigenDecomposite(Image<Gray, Byte> src, Image<Gray, Single>[] eigenImages, Image<Gray, Single> avg)

{

return CvInvoke.cvEigenDecomposite(

src.Ptr,

Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr; }),

avg.Ptr);

}

#endregion

/// <summary>

/// Given the eigen value, reconstruct the projected image

/// </summary>

/// <param name="eigenValue">The eigen values</param>

/// <returns>The projected image</returns>

public Image<Gray, Byte> EigenProjection(float[] eigenValue)

{

Image<Gray, Byte> res = new Image<Gray, byte>(\_avgImage.Width, \_avgImage.Height);

CvInvoke.cvEigenProjection(

Array.ConvertAll<Image<Gray, Single>, IntPtr>(\_eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr; }),

eigenValue,

\_avgImage.Ptr,

res.Ptr);

return res;

}

/// <summary>

/// Get the Euclidean eigen-distance between <paramref name="image"/> and every other image in the database

/// </summary>

/// <param name="image">The image to be compared from the training images</param>

/// <returns>An array of eigen distance from every image in the training images</returns>

public float[] GetEigenDistances(Image<Gray, Byte> image)

{

using (Matrix<float> eigenValue = new Matrix<float>(EigenDecomposite(image, \_eigenImages, \_avgImage)))

return Array.ConvertAll<Matrix<float>, float>(\_eigenValues,

delegate(Matrix<float> eigenValueI)

{

return (float)CvInvoke.cvNorm(eigenValue.Ptr, eigenValueI.Ptr, Emgu.CV.CvEnum.NORM\_TYPE.CV\_L2, IntPtr.Zero);

});

}

/// <summary>

/// Given the <paramref name="image"/> to be examined, find in the database the most similar object, return the index and the eigen distance

/// </summary>

/// <param name="image">The image to be searched from the database</param>

/// <param name="index">The index of the most similar object</param>

/// <param name="eigenDistance">The eigen distance of the most similar object</param>

/// <param name="label">The label of the specific image</param>

public void FindMostSimilarObject(Image<Gray, Byte> image, out int index, out float eigenDistance, out String label)

{

float[] dist = GetEigenDistances(image);

index = 0;

eigenDistance = dist[0];

for (int i = 1; i < dist.Length; i++)

{

if (dist[i] < eigenDistance)

{

index = i;

eigenDistance = dist[i];

}

}

label = Labels[index];

}

/// <summary>

/// Try to recognize the image and return its label

/// </summary>

/// <param name="image">The image to be recognized</param>

/// <returns>

/// String.Empty, if not recognized;

/// Label of the corresponding image, otherwise

/// </returns>

public String Recognize(Image<Gray, Byte> image)

{

int index;

float eigenDistance;

String label;

FindMostSimilarObject(image, out index, out eigenDistance, out label);

return (\_eigenDistanceThreshold <= 0 || eigenDistance < \_eigenDistanceThreshold ) ? \_labels[index] : String.Empty;

}

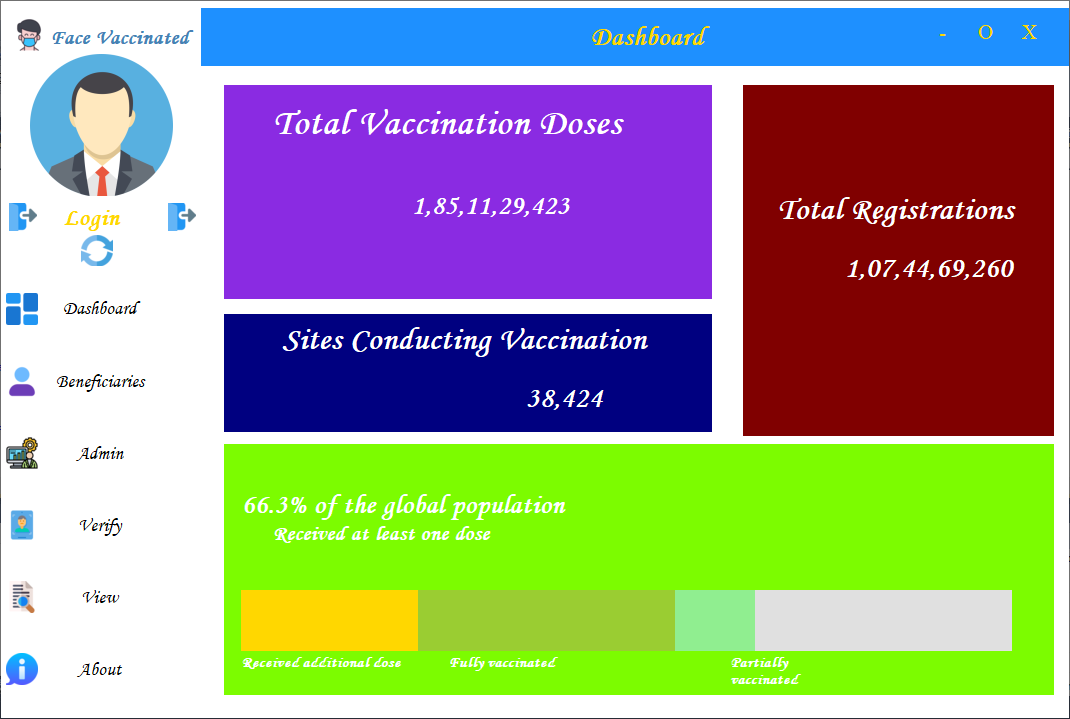
}

}

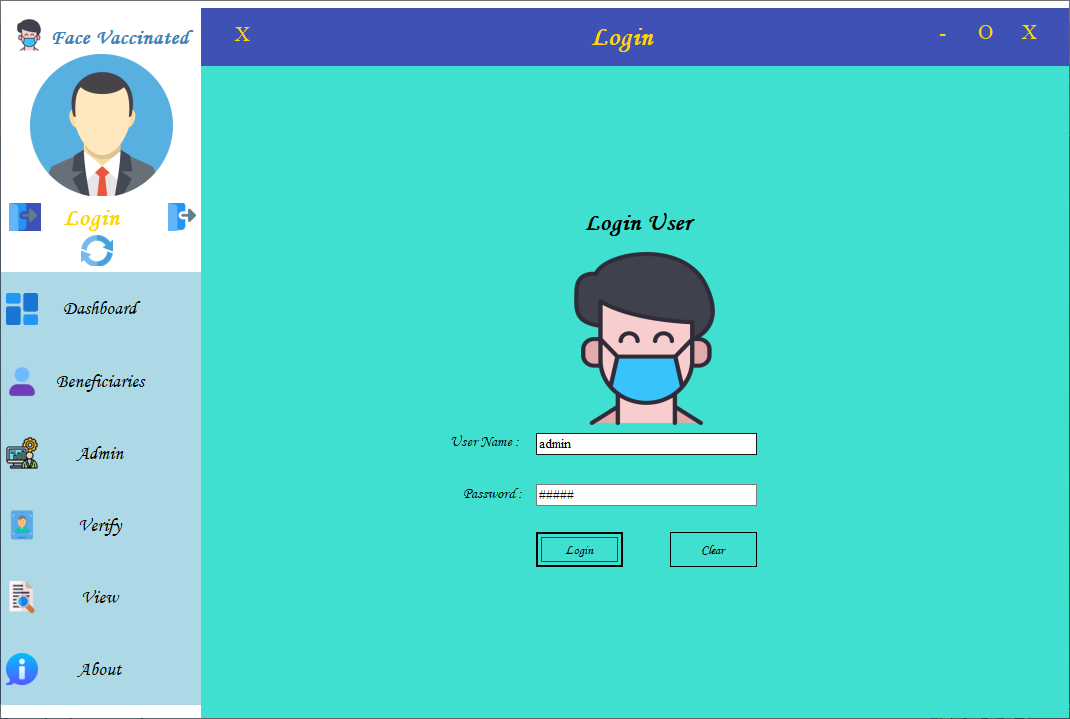
1. **SCREEN SHOTS:**

**6.1 FORM DESIGN :**

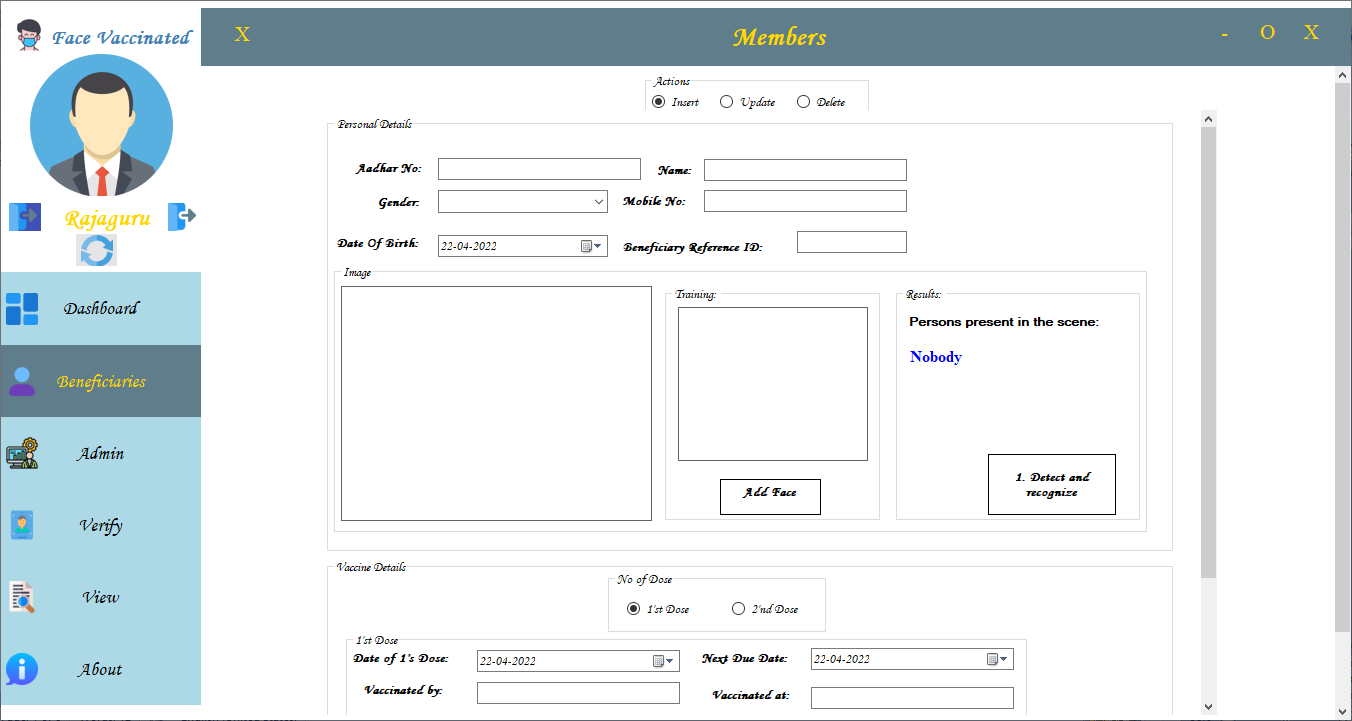
Dashboard:



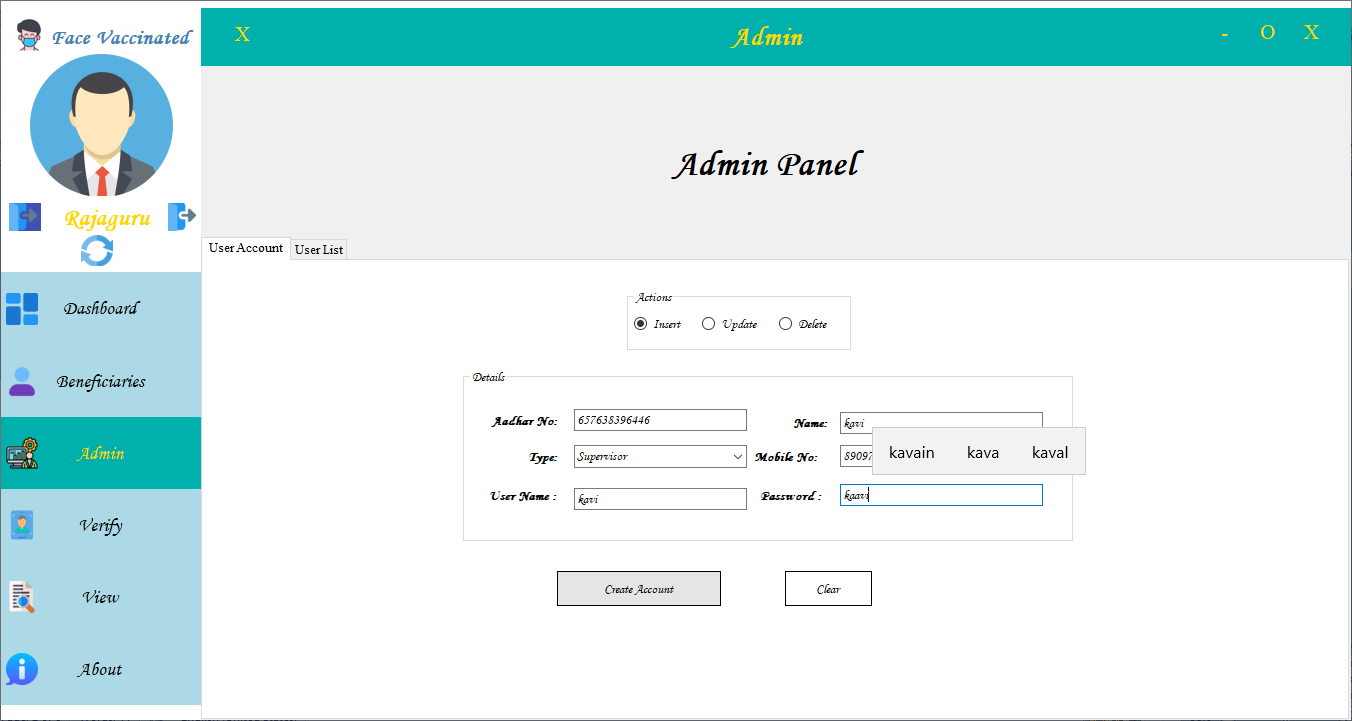
Login:

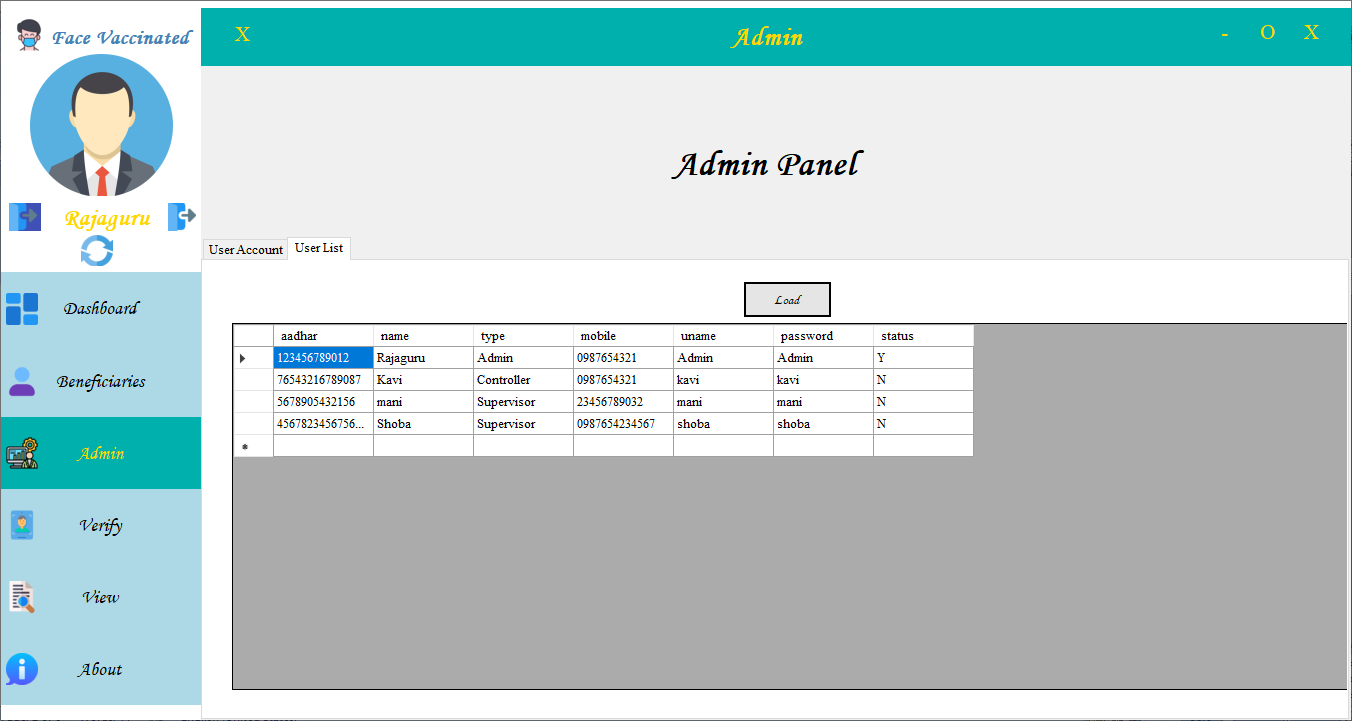


Beneficiaries:

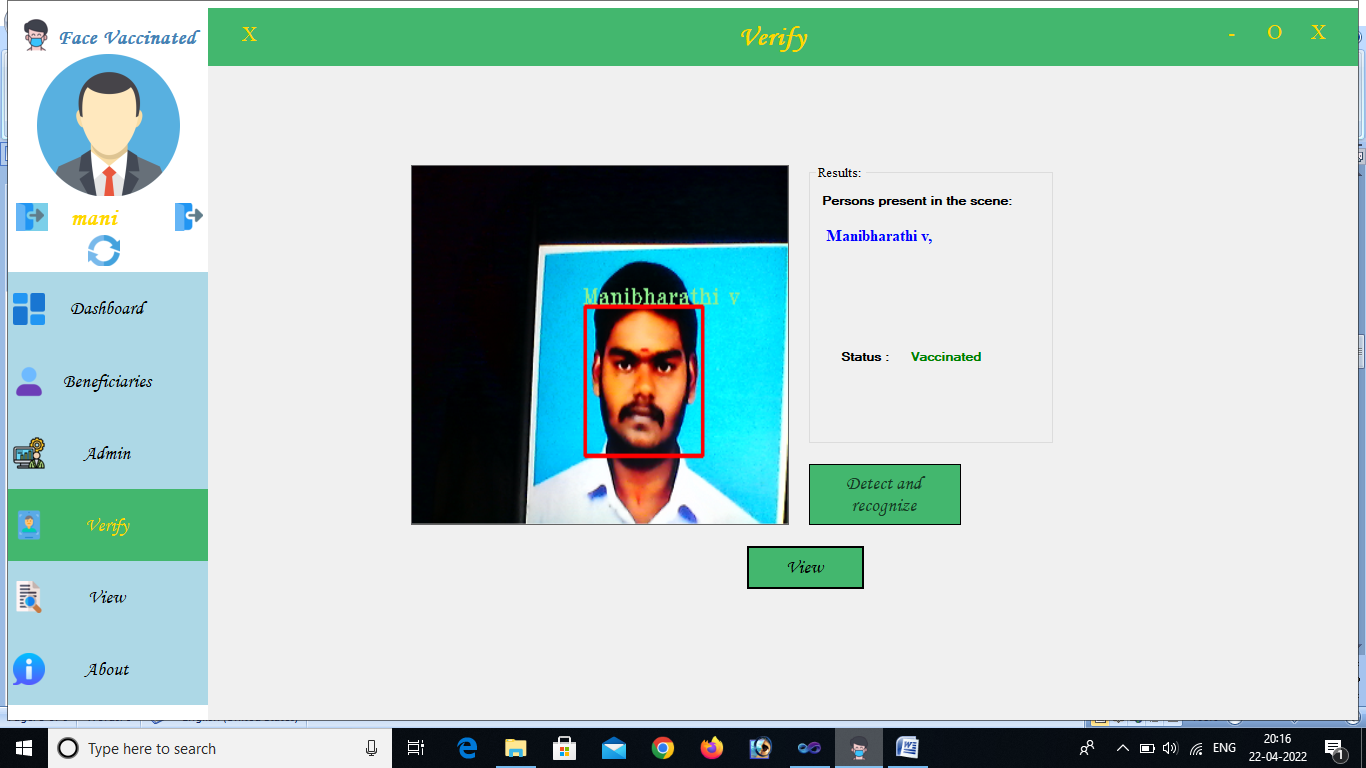


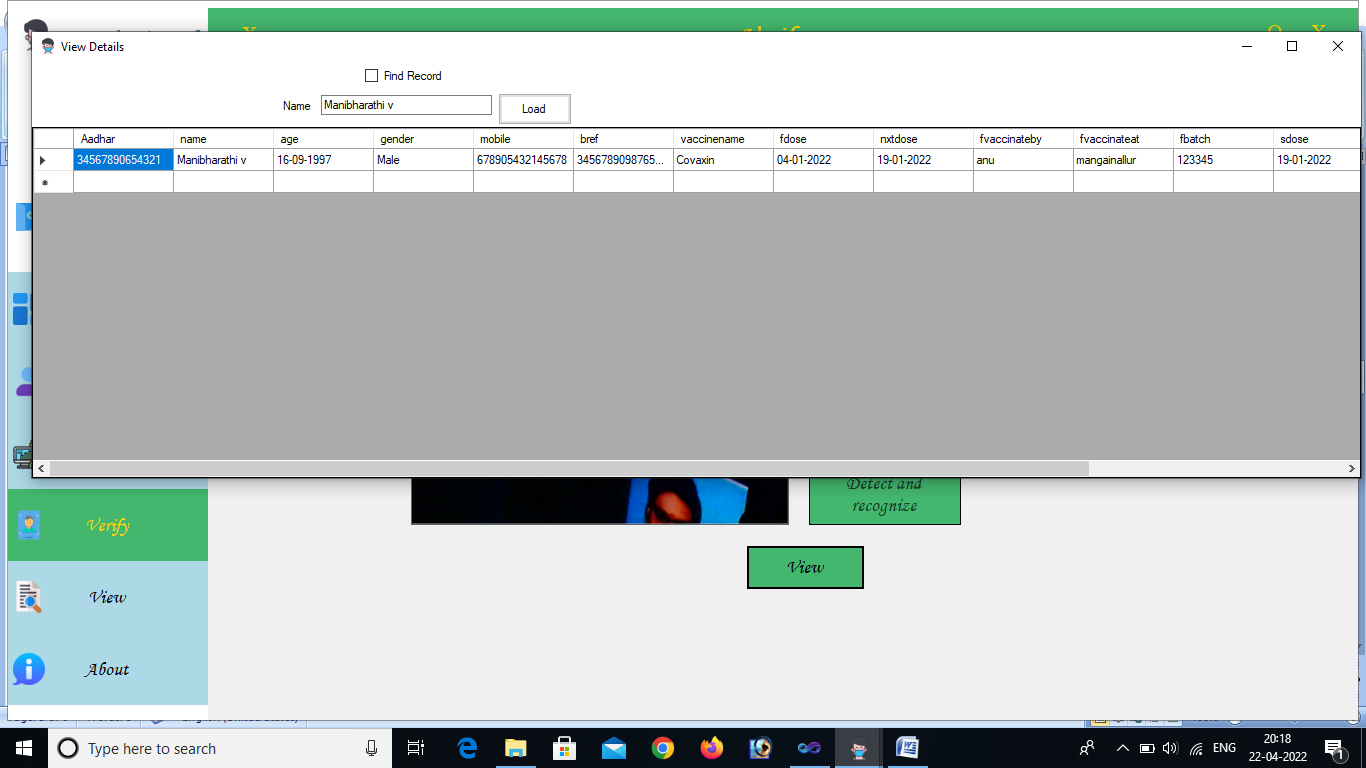
Admin:



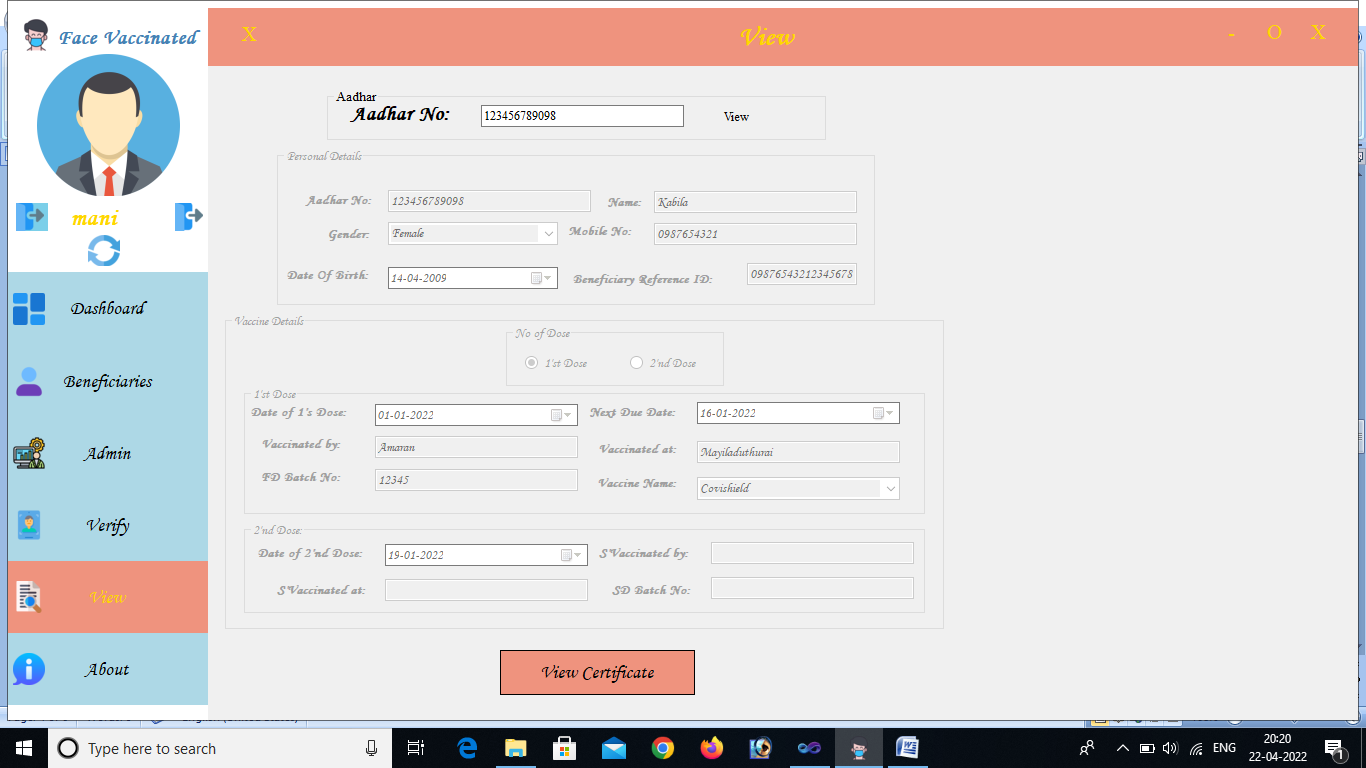


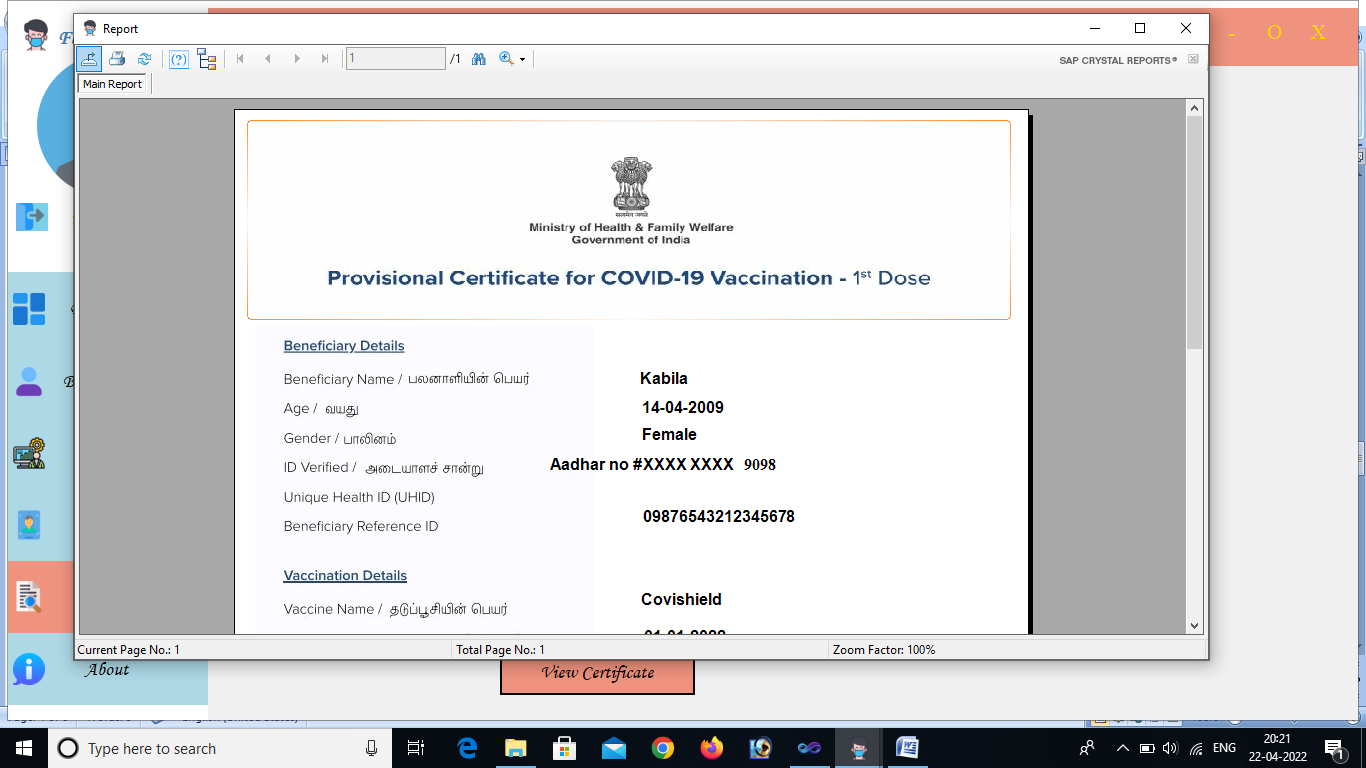
Verify:





View:





1. **CONCLUTION**

Our project is only a humble venture to satisfy the needs in a library. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the organization.

The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made with in a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

Last but not least it is no the work that played the ways to success but **ALMIGHTY.**

**8.REFERENCE**

Decision Support Systems, (ed., W. House), Petro celli, New York, 1983.

Building Decision Support Systems, (ed., J. L. Bennett), Addison-Wesley, Reading, MA, 1983.

Knowledge Representation for Decision Support Systems, (eds., L. Meth lie and R. Sprague), North-Holland, Amsterdam, 1985.

Decision Support Systems: A Decade in Perspective, (eds., E. R. McLean and H. G. Sol), North-Holland, Amsterdam, 1986.

Manager's Guide to Expert Systems Using Guru, C. W. Hollsopple and A. B. Winston, Dow Jones-Irwin, Homewood, IL 1986.

Business Expert Systems, C. W. Hollsopple and A. B. Winston, Irwin, Homewood, IL, 1987.

Decision Support Systems: Theory and Applications, (eds., C. W. Hollsopple and A. B. Winston), Springer-Verlag, Berlin, 1987.

Expert Systems for Business, (ed., B. Silverman), Addison-Wesley, Reading, MA, 1987.

The Information Jungle: A Quasi Novel Approach to Managing Corporate Knowledge, C. W. Hollsopple and A. B. Winston, Dow Jones-Irwin, Homewood, IL, 1988.