1. INTRODUCTION

1.1 General

Steganography is a technique for covert communication, which aims to hide secret messages into ordinary digital media without drawing suspicion. Designing steganography algorithms for various cover sources is challenging due to the fundamental lack of accurate models. Currently, the most successful approach for designing content adaptive steganography is based on minimizing the distortion between the cover and the corresponding steno object. The distortion is obtained by assigning a cost to each modified cover element (e.g., pixel in the spatial domain image), and the messages are embedded while minimizing the total distortion which is the sum of costs of all modified elements.

1.2 Objective

The cost reassignment scheme where extensive experiments are done on several kinds of steno algorithms, steganalysis features and cover databases that demonstrate the CPP rule which can improve the security of state-of-the-art steganography algorithms for spatial images.

1.3 Existing System

Most modern secure image steganographic schemes define distortion functions for constraining the embedding changes to those parts of the image that are difficult to model such as textured or noisy regions. A two-player zero-sum game is conducted between steganographer and attacker related to the security of practical steganography.

1.3.1 Existing System Disadvantages

- Moderate distortion
- Noise is high

1.4 Proposed System

CPP Rule

The steganographic methods have a very similar Security performance while defining distortions in very different ways. Among these methods, there is a distinguishment on the costs assignment for some pixels; in other words, the costs assigned on some pixels are large in one method but small in another. They are defined as "controversial pixels" Because they are assigned with very different costs in different algorithms. Even with such a discrepancy, some of these algorithms can still provide the same level of security. This phenomenon implies that modifications on such controversial pixels have little effect on features of steganalysis.

1.4.1 Proposed System Advantages

- Noise reduction
- Improved distortion

2. LITERATURE REVIEW

Title 1 : A Short Survey on Image Steganography and Steganalysis Techniques

Author : Yam bern Jina Chanu, ThemrichonTuithung, Kh. Manglem Singh

Year : 2014

Description

The paper describes a short survey on different types of steganography techniques for image in spatial and transform domains and steganalysis techniques for the detection of secret message in the image. The strong and weak points of these techniques are mentioned briefly so that researchers who work in steganography and steganalysis gain prior knowledge in designing these techniques and their variants. One can develop a better steganography technique by analyzing the contemporary steganalysis techniques.

Title 2 : Defining Cost Functions for Adaptive Steganography at the Micro scale

Author : Kejiang Chen, Weiming Zhang, Hang Zhou, Nenghai Yu.

Year : 2015.

Description

In the framework of minimizing embedding distortion steganography, the definition of cost function almost determines the security of the method. Generally speaking, texture areas would be assigned low cost, while smooth areas with high cost. However, the prior methods are still not precise enough to capture image details. In this paper, a novel scheme of defining cost function for adaptive steganography at the micro scale is used. The proposed scheme is designed by using a "microscope" to highlight fine details in an image so that distortion definition can be more refined. Experiments show that by adopting our scheme, the current steganographic methods (WOW, UNIWARD, HILL) will achieve better performances on resisting the state-of-the-art steganalysis.

Title 3 : Gibbs Construction in Steganography

Author : Tomáˇs Filler, Jessica Fridrich

Year : 2013

Description

A connection is made between steganography design by minimizing embedding distortion and statistical physics. The unique aspect of this work and one that distinguishes it from prior art and that allows the distortion function to be arbitrary, which permits us to consider spatially dependent embedding changes. It provides complete theoretical framework and describes practical tools, such as the thermodynamic integration for computing the rate-distortion bound and the Gibbs sampler for simulating the impact of optimal embedding schemes and constructing practical algorithms. The proposed framework reduces the design of secure steganography in empirical covers to the problem of finding local potentials for the distortion function that correlate with statistical delectability in practice. By working experimentally validated the approach and discuss various options available to the steganographer in practice.

Title 4 : Rich Models for Steganalysis of Digital Images

Author : Jessica Fridrich, Jan Kodovský

Year : 2016.

Description

The novel general strategy for building steganography detectors for digital images are described. The process starts with assembling a rich model of the noise component as a union of many diverse sub models formed by joint distributions of neighboring samples from quantized image noise residuals obtained using linear and nonlinear high-pass filters. In contrast to previous approaches, the model assembles a part of the training process driven by samples drawn from the corresponding cover- and stego-sources. Ensemble classifiers are used to assemble the model as well as the final steganalyzer due to their low computational complexity and ability to efficiently work with high-dimensional feature spaces and large training sets. The proposed framework is demonstrated on three steganographic algorithms designed to hide messages in images represented in the spatial domain: HUGO, edge-adaptive algorithm by Luo, and optimally coded ternary 1 embedding. For each algorithm, a simple sub model-selection technique is applied to increase the detection accuracy per model dimensionality and show how the detection saturates with increasing complexity of the rich model. By observing the differences between how different sub models engage in detection, an interesting interplay between the embedding and detection is revealed. Steganalysis built around rich image models combined with ensemble classifiers is a promising direction towards automat zing steganalysis for a wide spectrum of steganographic schemes.

Title 5 **:** Designing steganographic distortion using directional filters

Author **:** Vojtˇech Holub and Jessica Fridrich

Year **:** 2012.

Description

This paper presents a new approach to defining additive steganographic distortion in the spatial domain. The change in the output of directional high-pass filters after changing one pixel is weighted and then aggregated using the reciprocal Holder norm to define the individual pixel costs. In contrast to other adaptive embedding schemes, the aggregation rule is designed to force the embedding changes to highly textured or noisy regions and to avoid clean edges. Consequently, the new embedding scheme appears markedly more resistant to steganalysis using rich models. The actual embedding algorithmic realized using syndrome-trellis codes to minimize the expected distortion for a given payload.

3. DESIGN

3.1 Requirements

Using this requirement, our application provides high service with efficiently. Software requirements deal with defining software resource requirements and pre-requisites that need to be installed on a server that provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed. The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

3.1.1 Hardware Requirements

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

➢ Processor	: Pentium Dual Core 2.00GHZ
➢ Hard disk	: 140 GB
➢ Mouse	: Logitech.
➢ RAM	: 4GB(minimum)
➢ Keyboard	: 110 keys enhanced.

3.1.2 Software Requirements

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

➢ Operating system	: Windows7 SP1,8,8.1
➢ IDE	: Microsoft Visual Studio .Net 2013
➢ Front End	: ASP.NET
➢ Coding Language	: C#
➢ Backend	: SQL Server 2012

3.1.3 Functional Requirements

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, and outputs. Our system requires minimum one system to achieve this concept.

3.1.4 Non-Functional Requirements

Effciency

The EOB efficiently supports the range queries and preserving high-level security compared to existing methods. EOB supports any version of IND-OCPA

3.2 System Architecture

Architecture diagram shows the relationship between different components of system. This diagram is very important to understand the overall concept of system. Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in the engineering world in hardware design, electronic design, software design, and process flow diagrams.

.

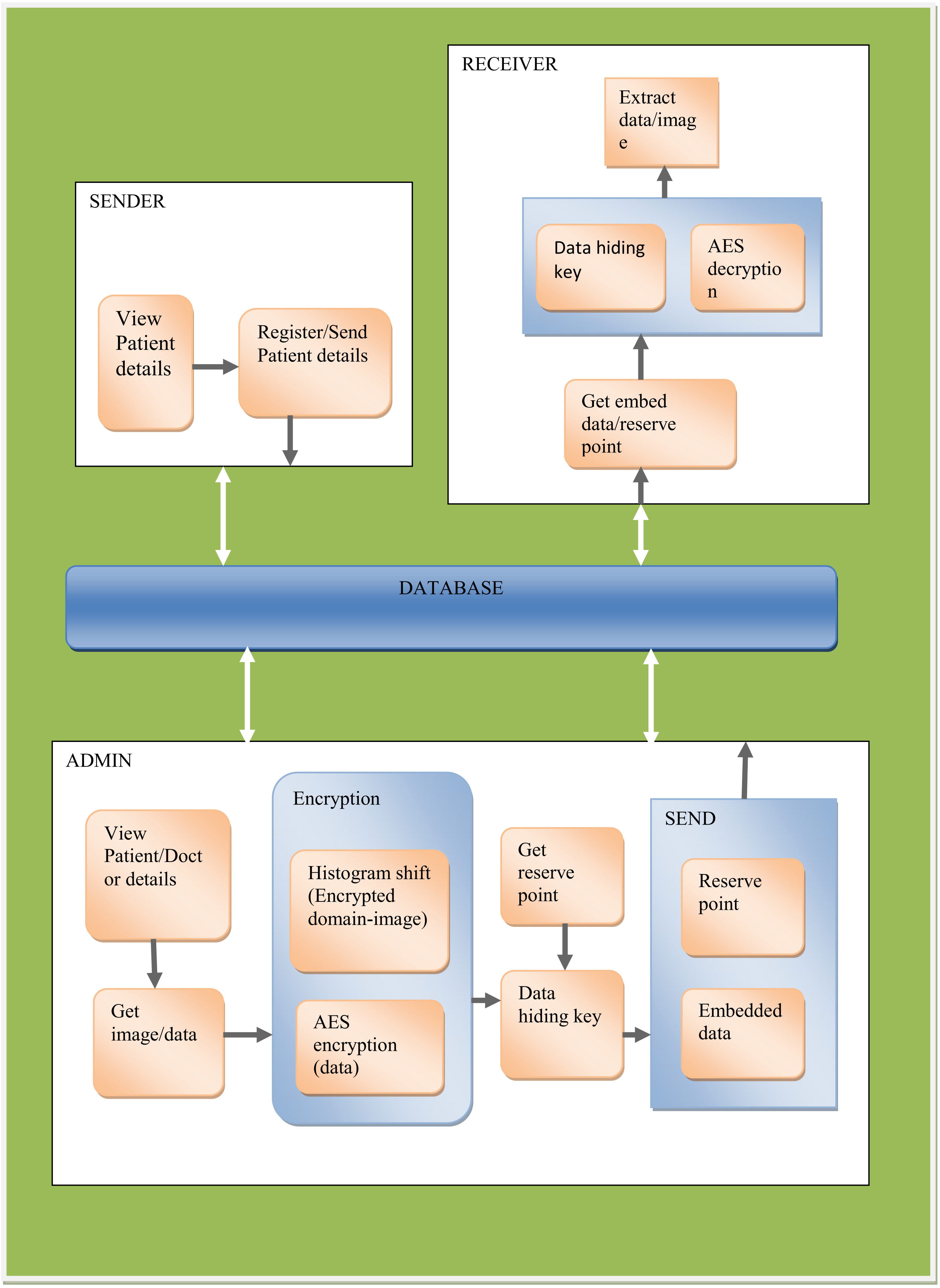


Fig 3.1 System Architecture

3.3 Modules

Patient

- Authentication
- Register Patient Details

Admin

- Authentication
- Encrypt Image
- Get Patient Details
- Get Reserve Point/Data/Send
- View Patient/Doctor Details

Receiver

- Extract Original Data

3.3.1 Module Description

Patient-Authentication

Registration

If you are the new user going to login into the application then you have to register first by providing necessary details. After successful completion of sign up process, the user has to login into the application by providing username and exact password.

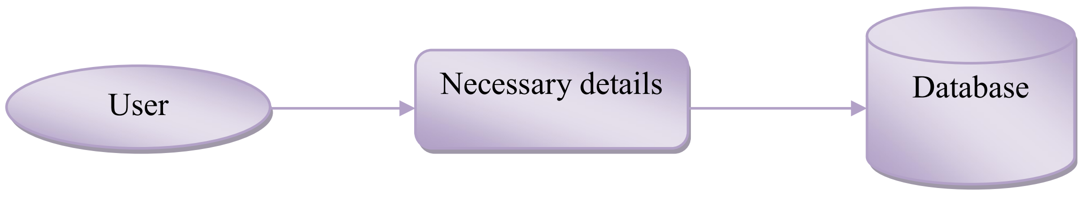


Fig 3.2 Patient-Authentication(Registration)

Login

The user needs to enter exact username and password. If login success means it will take up to upload page else it will remain in the login page itself.

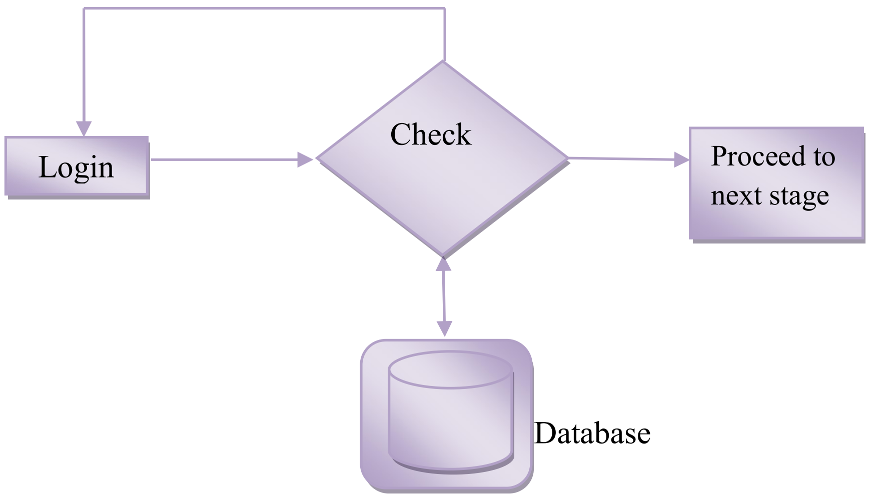


Fig 3.3 Patient-Authentication(Login)

Register patient details

Authentication success then doctor registers the patient details.

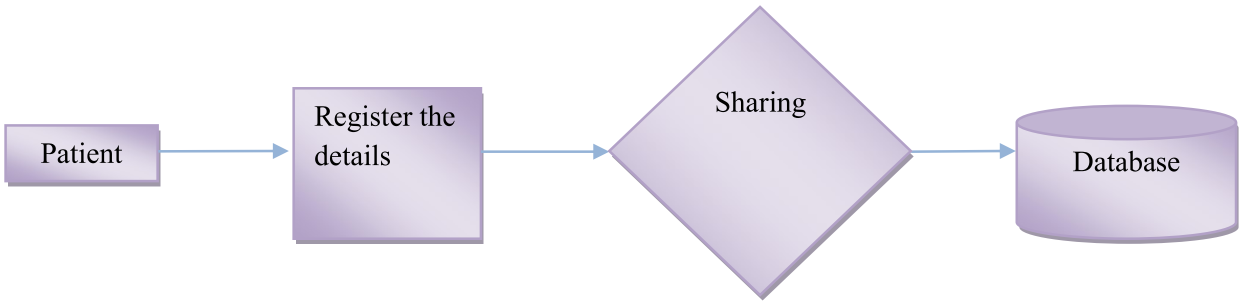


Fig 3.4 Register Patient Details

Admin

Authentication

The Admin needs to enter exact username and password. If login success means it will take up to upload page else it will remain in the login page itself.

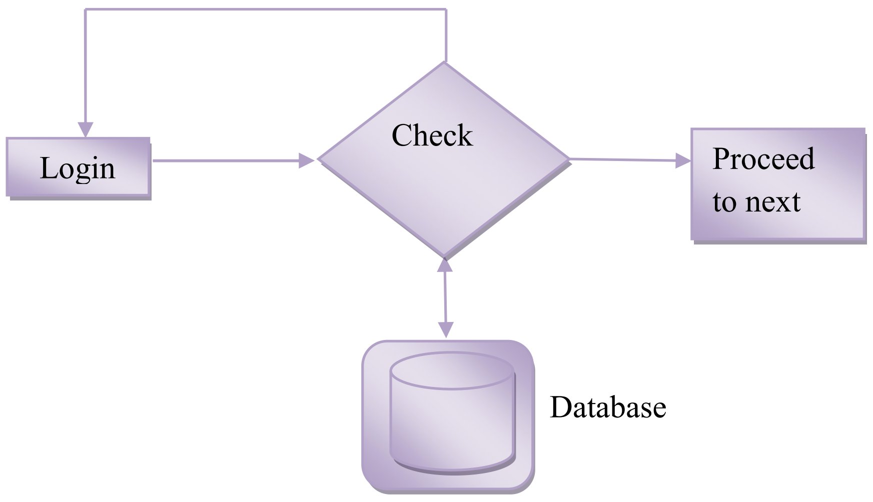


Fig 3.5 Admin Authentication

Encrypt cover medium

After login success admin can encrypt image.

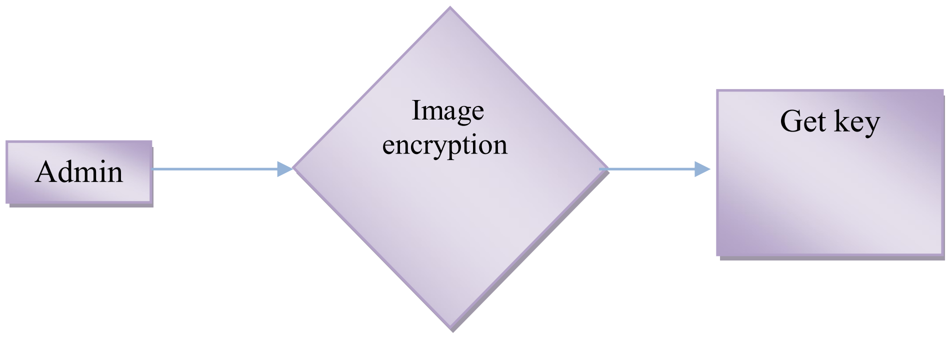


Fig 3.6 Encrypt Cover Medium

Embed the data

After login success admin can get patient data, after that encrypted data embed into image and send to receiver side.

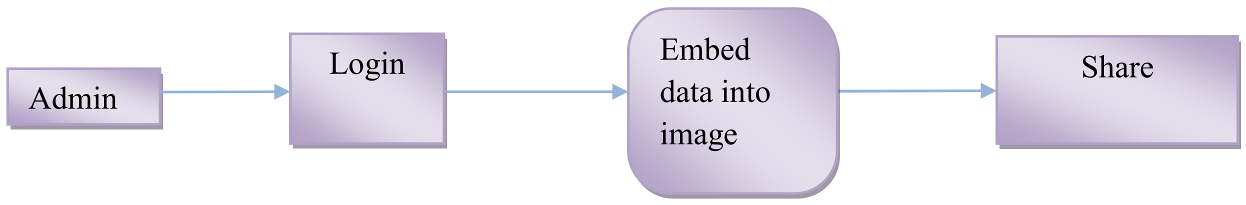


Fig 3.7 Embed the data

Reserve Point

Admin send the reserve point of image to receiver.

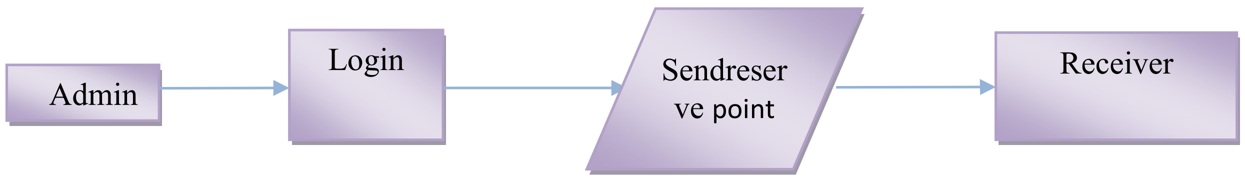


Fig 3.8 Reserve point

View patient/doctor details

Admin can view the patient /doctor details.

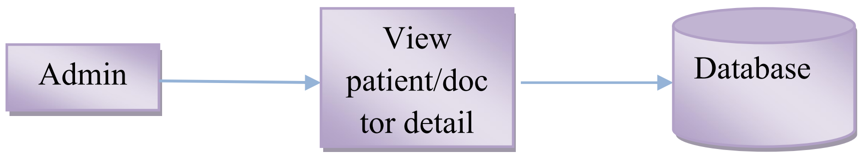


Fig 3.9 View patient/doctor details

Receiver

Extract data

The receiver extracts the original data from embedded image.

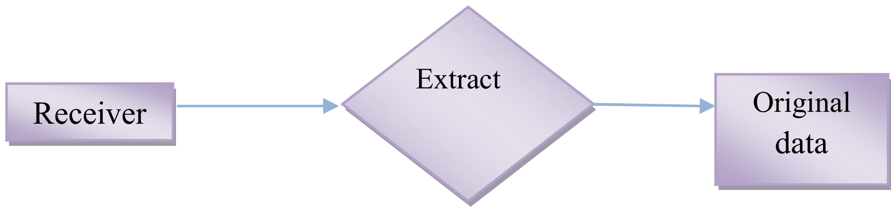


Fig 3.10 Extract data

3.4 UML Introduction

A diagram is the symbolic representation of information according to some visualization technique. In other words, it is the graphical presentation of a set of elements most often referred to connected graph of vertices(things) and arcs(relationships)

The Unified Modelling language is a standard general purpose modelling language in the field of object-oriented software engineering. UML includes a set of graphic notation techniques to create visual models of object-oriented software systems. UML combines techniques of data modelling, business modelling, object modelling and component modelling and can be used throughout the software development life-cycle and across different implementation technologies.

3.4.1 UML Specification

There are four parts to the UML 2.x specification:

1. The Superstructure that defines the notation and semantics for diagrams and their model elements.
2. The Infrastructure that defines the core met model on which the Superstructure is based.
3. The Object Constraint Language(OCL) for defining rules for model elements.
4. The UML Diagram Interchange that defines how UML 2 diagram layouts are exchanged.

The current versions of these standards follow

UML Superstructure version 2.4.1

UML Infrastructure version 2.4.1

OCL version 2.3.1

UML Interchange version 1.0

Design

UML offers a way to visualize a system's architectural blueprints in a diagram, including elements such as

- Any activities (jobs)
- Individual component of the system
- And how they interact with other software components
- How the system will run
- How entities interact with others
- External interface

3.4.2 Modelling

It is important to distinguish between the UML model and the set of diagrams of the system. A diagram is a partial graphic representation of a system's model.

The model may also contain documentation that drives the model elements and diagrams

UML diagrams represent two different views of a system model.

- Static (or Structural) view : Emphasizes the static structure of the system using objects , attributes ,operations and relationships.

  It includes

  Class Diagrams

  Composite Structure diagrams

  Object Diagrams

  Deployment Diagrams

  Component Diagrams
- Dynamic (or Behavioural) view: Emphasizes the dynamic behaviour of the system by showing collaborations among objects and changes to the internal states of objects.

  It includes

Sequence diagrams

Activity diagrams

State machine diagrams

Use case diagrams

Interaction Diagram

UML Diagrams Description

Class Diagram

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature.

Active class is used in a class diagram to represent the concurrency of the system. Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

Object Diagram

Object diagrams can be described as an instance of class diagram. Thus, these diagrams are more close to real-life scenarios where we implement a system. Object diagrams are a set of objects and their relationship is just like class diagrams. They also represent the static view of the system. The usage of object diagrams is similar to class diagrams but they are used to build prototype of a system from a practical perspective.

Component Diagram

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system. During the design phase, software artifacts (classes, interfaces, etc.) of a system are arranged in different groups depending upon their relationship. Now, these groups are known as components. Finally, it can be said component diagrams are used to visualize the implementation.

Deployment Diagram

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

Sequence Diagram

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another. Interaction among the components of a system is very important from implementation and execution perspective. Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

Collaboration Diagram

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links. The purpose of collaboration diagram is similar to sequence diagram. However, the specific purpose of collaboration diagram is to visualize the organization of objects and their interaction.

State chart Diagram

Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. State chart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface, etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.

Activity Diagram

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

UML Diagrams

Class Diagram

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.

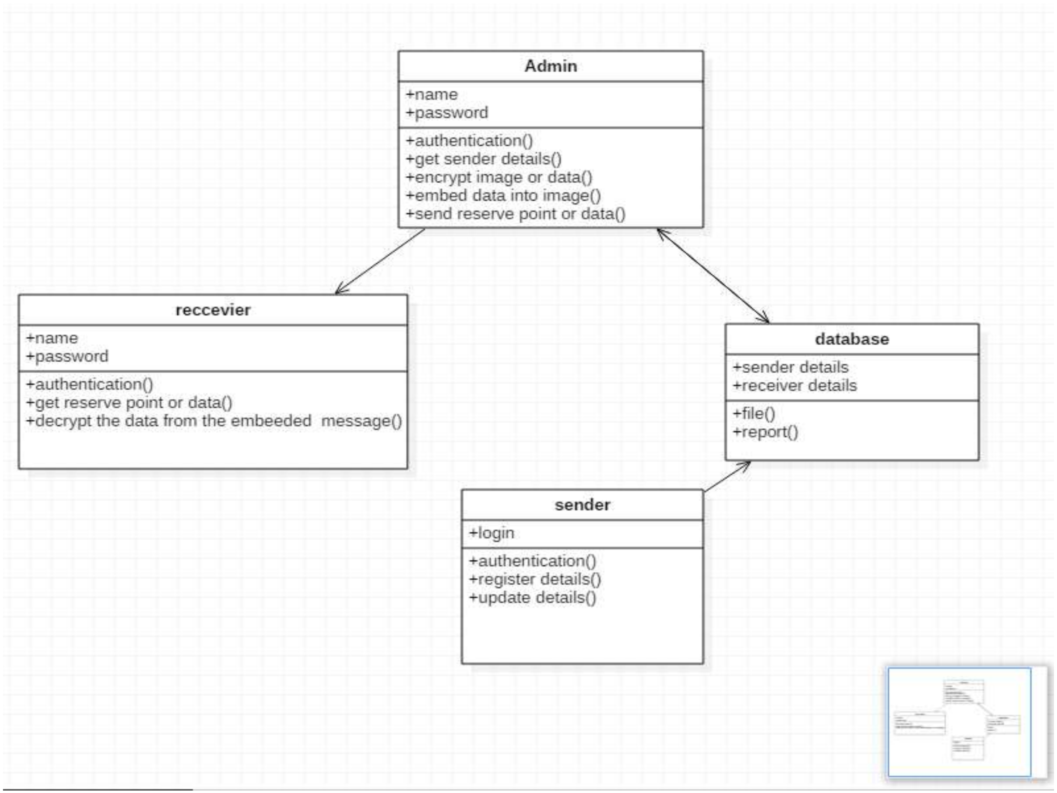


Fig 3.11 Class Diagram

Use Case Diagram

A use case diagram is a type of behavioral diagram created from a Use-case analysis. The purpose of use case is to present overview of the functionality provided by the system in terms of actors, their goals and any dependencies between those use cases.

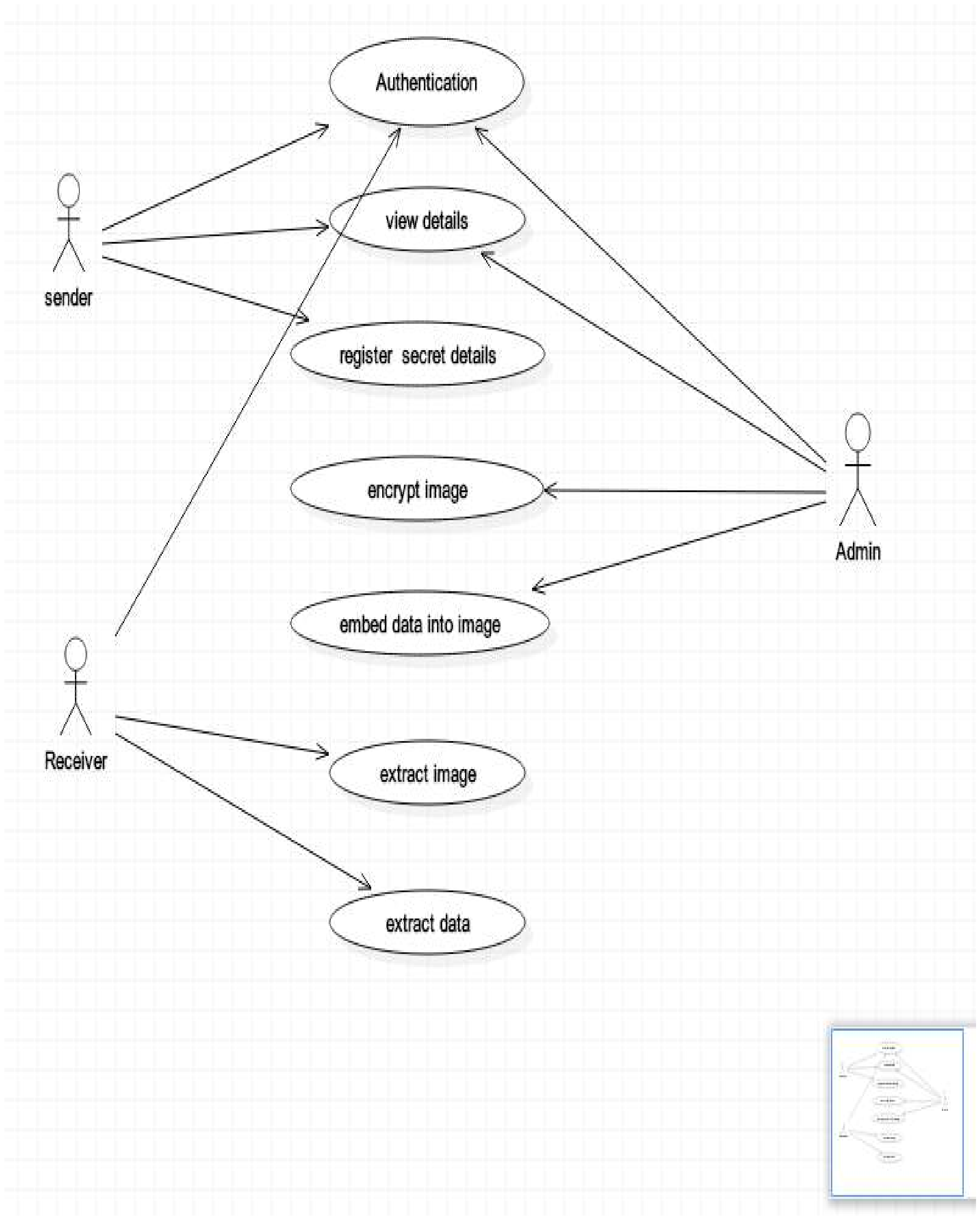


Fig 3.12 Use case Diagram

State chart

State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction**.**

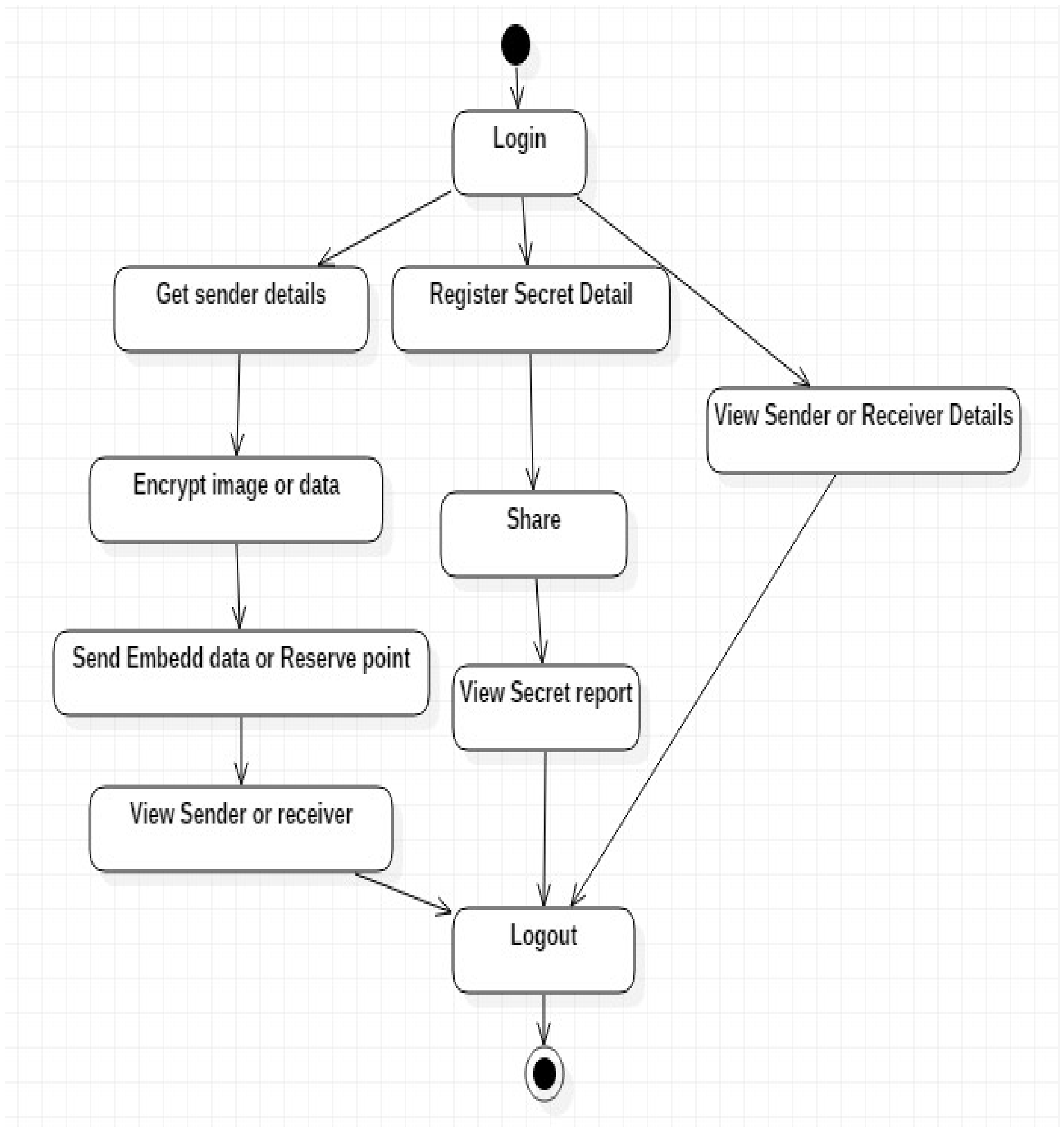


Fig 3.13 statechart Diagram

Sequence Diagram

A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order.It is a construct of a message sequence chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.

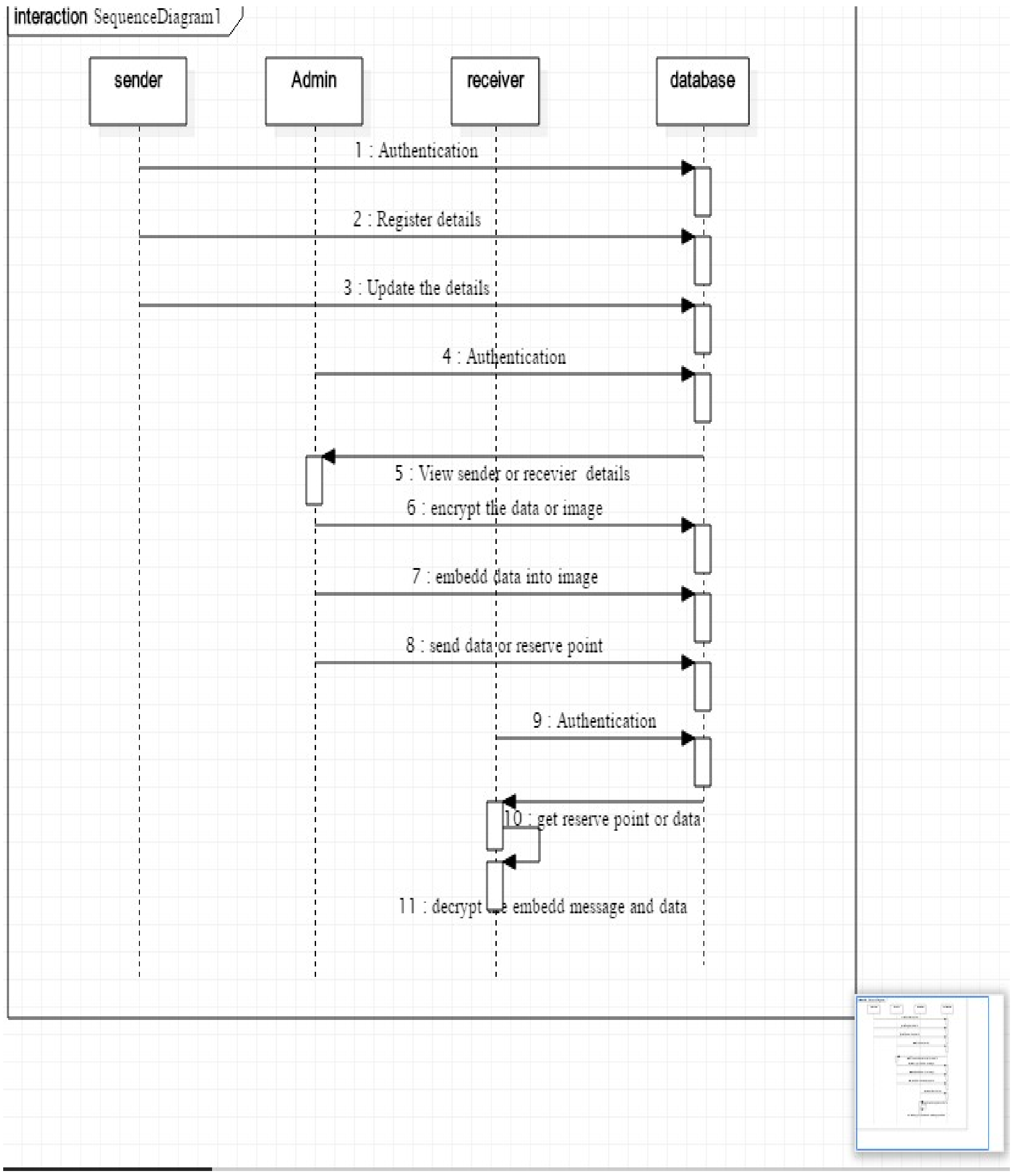

interaction SequenceDiagram1
sender
Admin
receiver
database
1 : Authentication
2 : Register details
3 : Update the details
4 : Authentication
5 : View sender or recevier details
6 : encrypt the data or image
7 : embedd data into image
8 : send data or reserve point
9 : Authentication
10 : get reserve point or data
11 : decrypt the embedd message and data

Fig 3.14 Sequence Diagram

Collaboration Diagram

A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram. It can be the decomposition of a use case, use case diagram, or part of a use case diagram.

The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

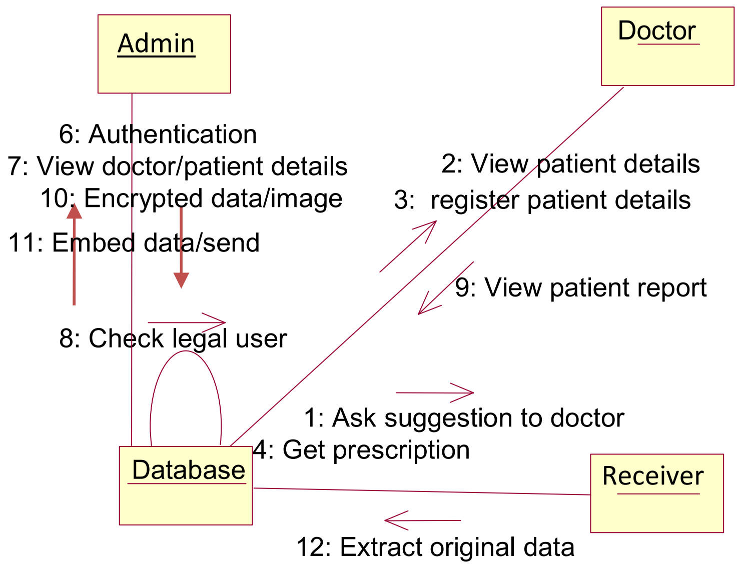


Fig 3.15 collaboration Diagram

Component Diagram

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

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

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

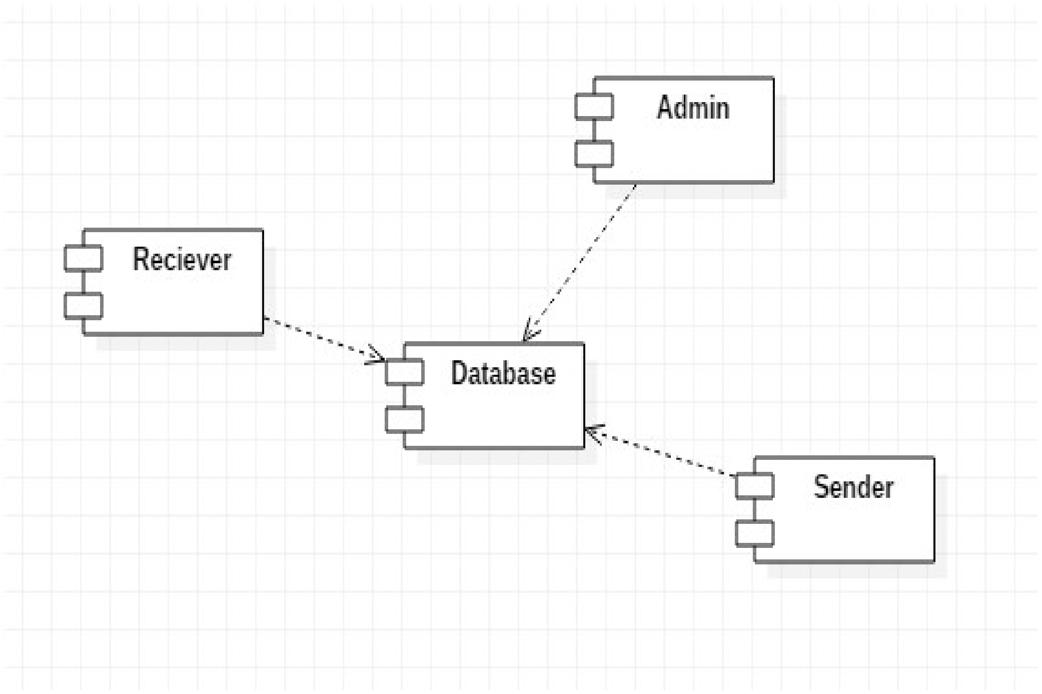


Fig 3.16 component Diagram

Object Diagram

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

An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time.

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

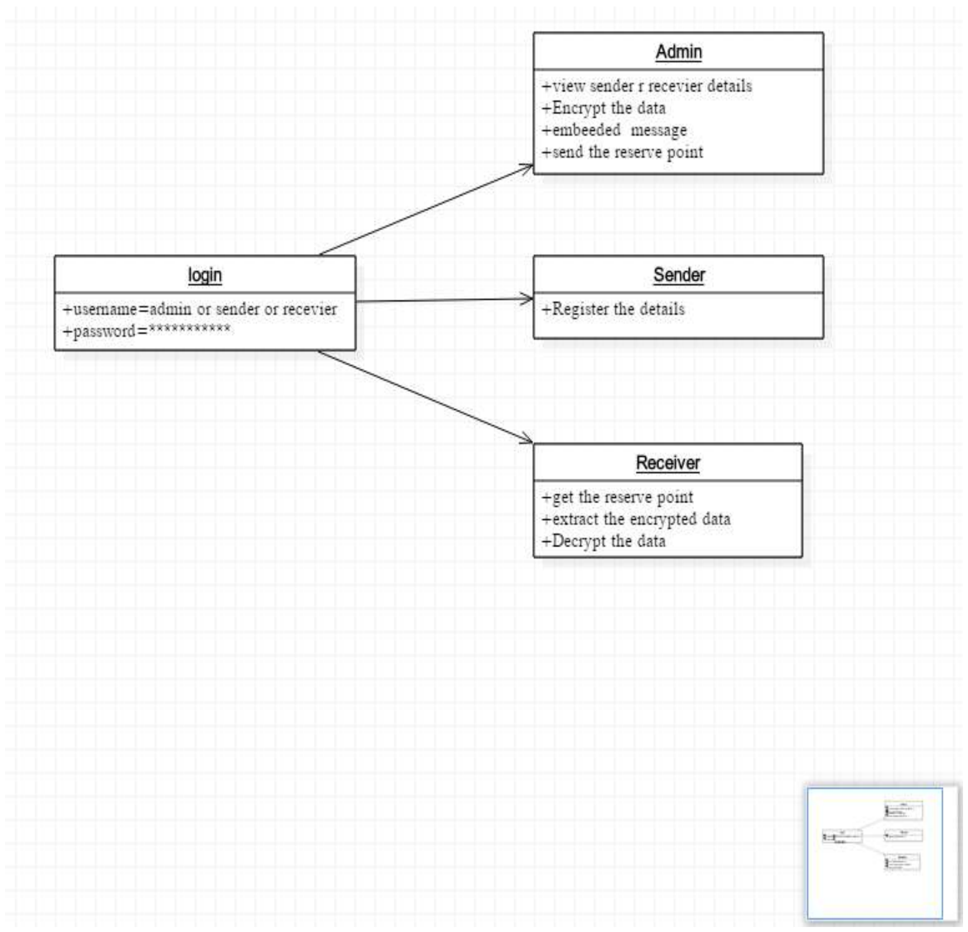


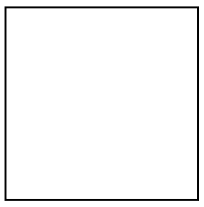
Fig 3.17 Object Diagram

Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts.

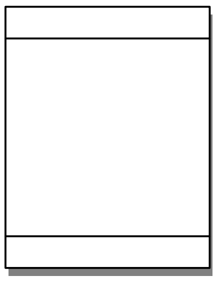
There are four symbols:

1. Squares representing external entities, which are sources or destination



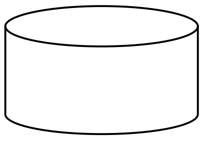
External entity

2. Rounded rectangles representing process, which take data as input



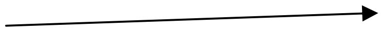
Process

3. Data store: manual or computer storage of data



Data store

4. Data flow: Data transfer in the direction indicated by the arrow. Each arrow should be labelled to indicate what data is being transferred.

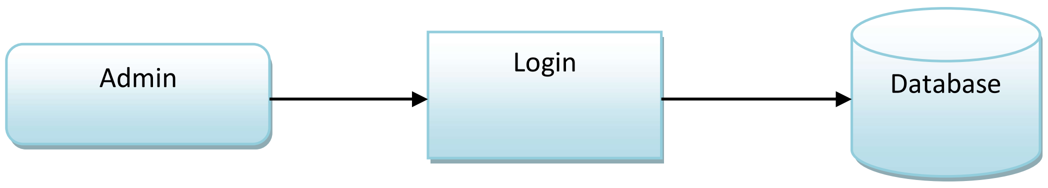


3.4.3 DFD layers

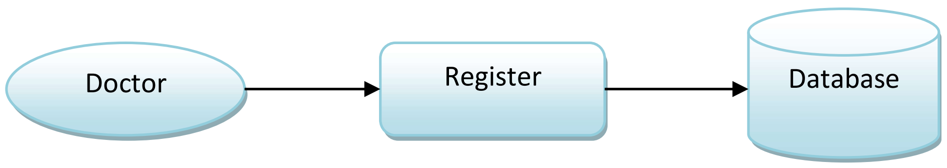
The first level DFD shows the main processes within the system.

Each of these processes can be broken into further processes until you reach pseudo code.

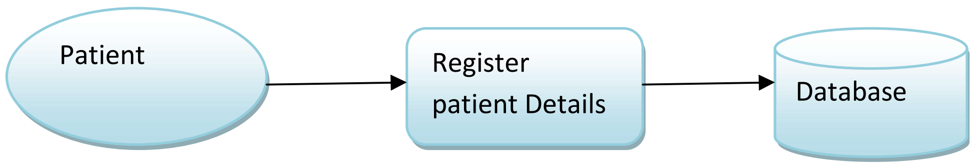
Level 0



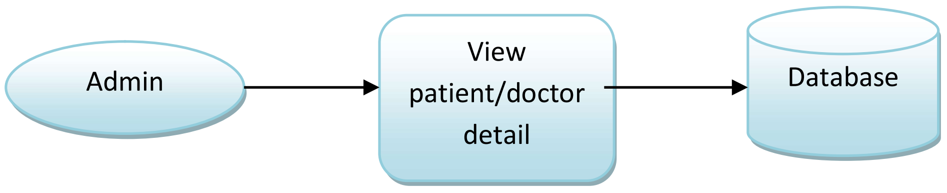
Level 1



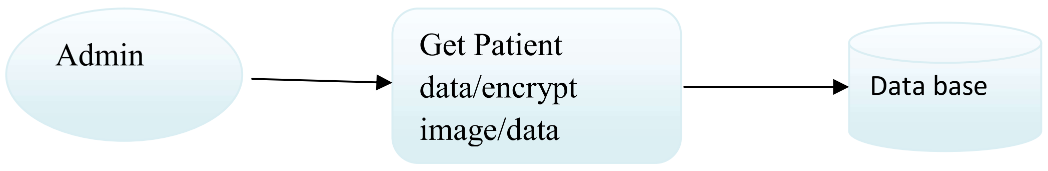
Level 2



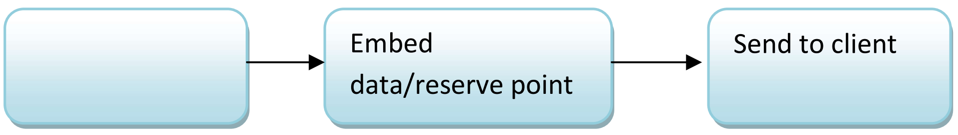
Level 3



Level 4



Level 5



Level 6

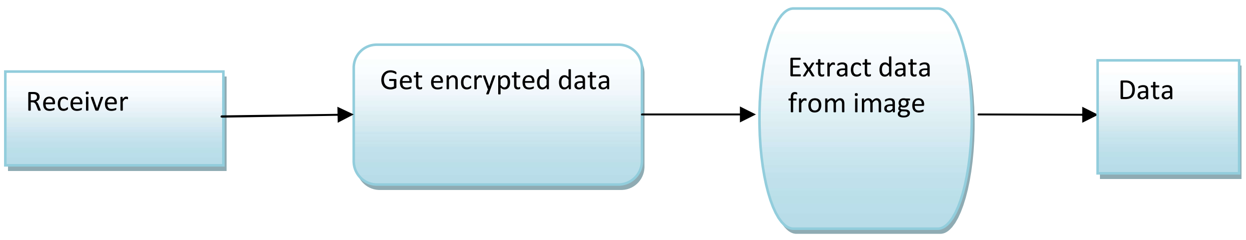


Fig 3.18 DFD(0-6) layers

All Level DFD

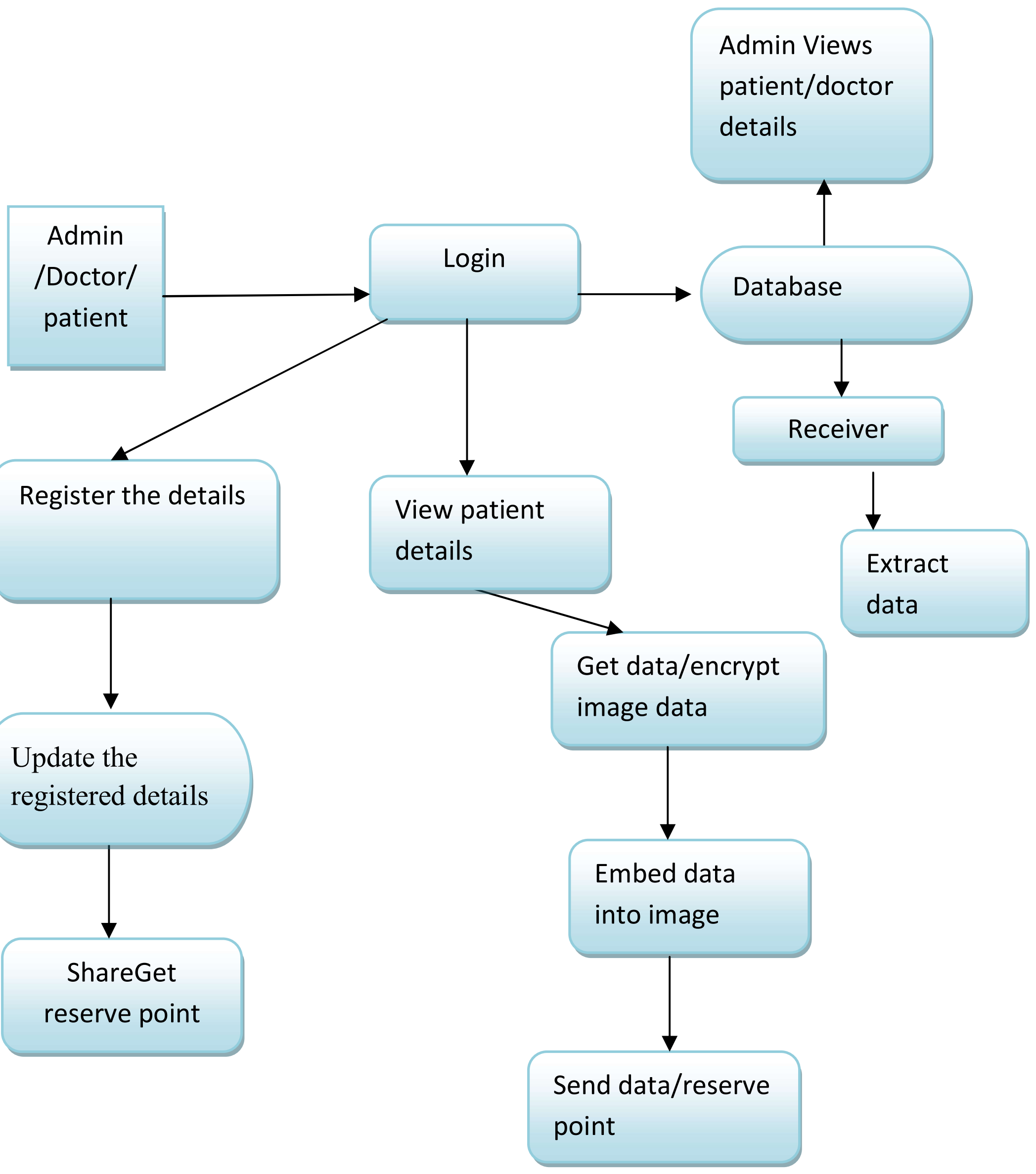


Fig 3.19 All DFD layers

3.4.4 E-R Diagram

In software engineering, an Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called Entity-Relationship Diagrams, ER diagrams, or ERDs.

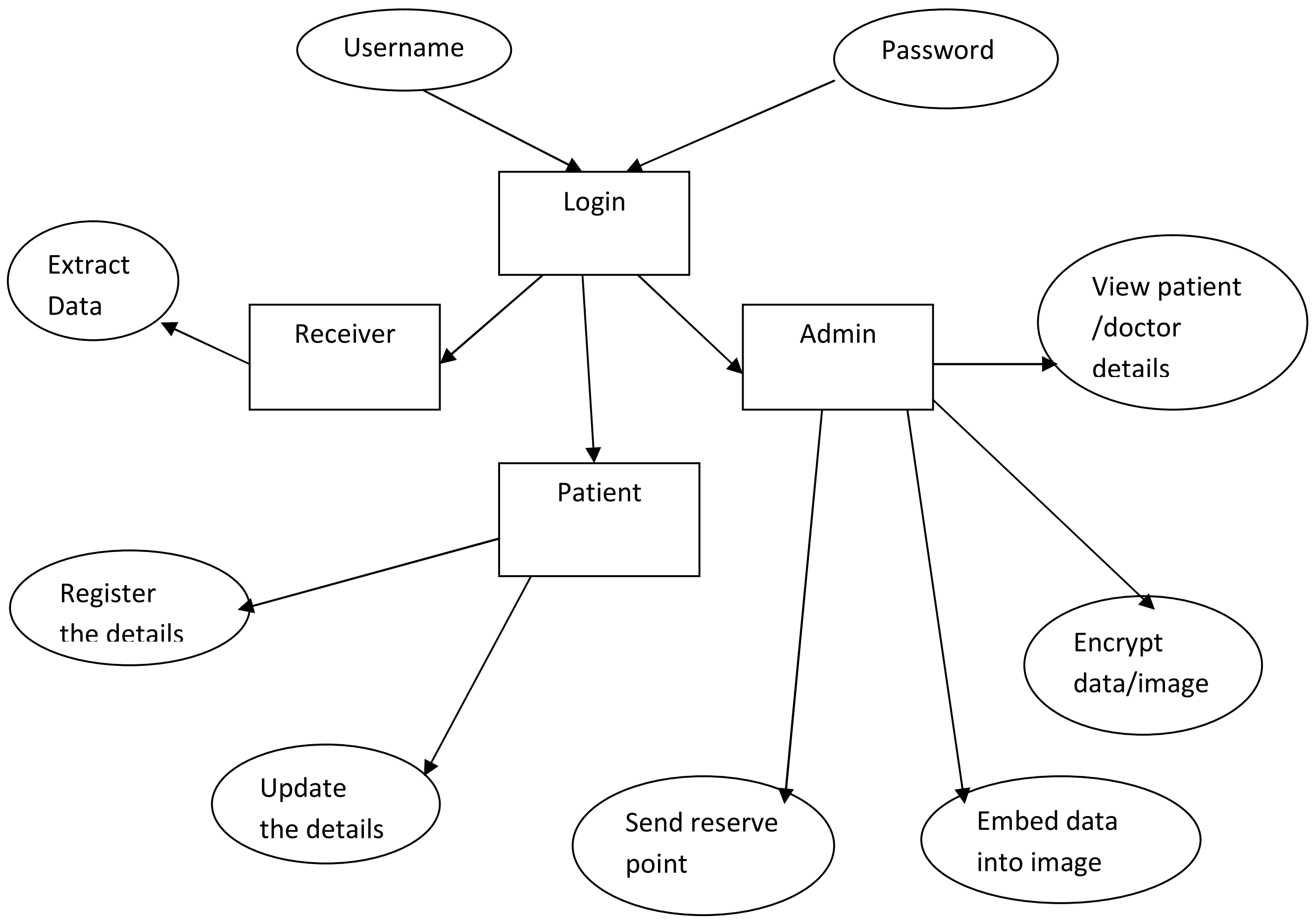
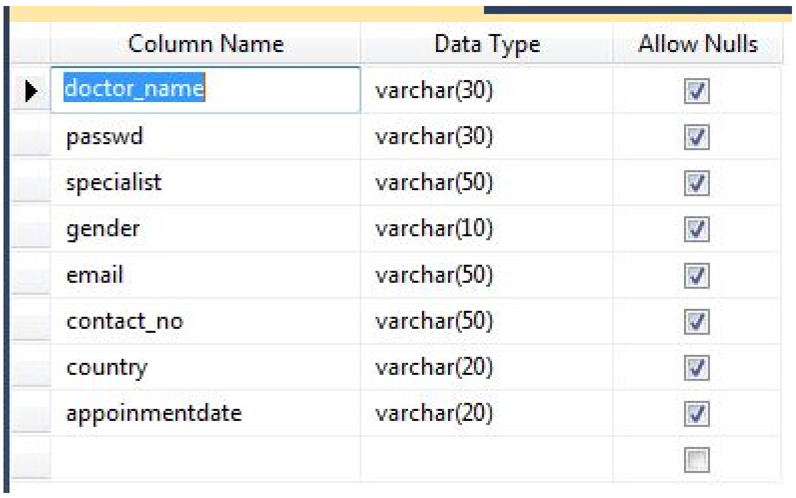


Fig 3.20 E-R Diagram

3.5 Database Design Structure

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

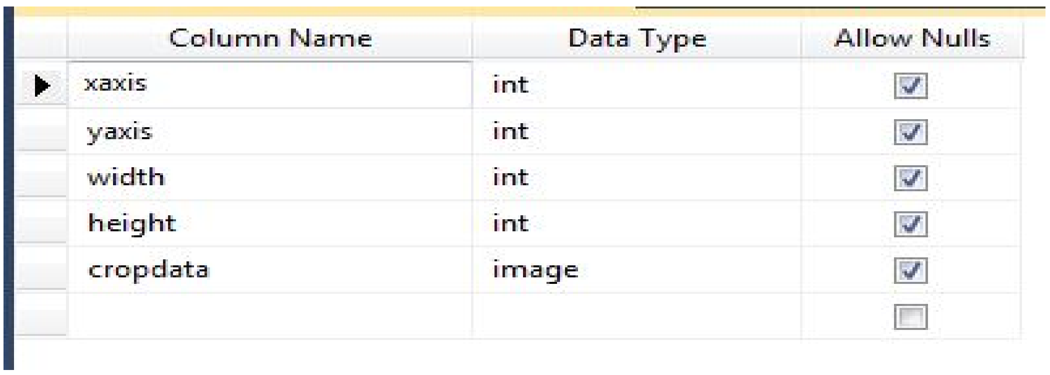
Table - 3.1 Doctor Table



Column Name	Data Type	Allow Nulls
doctor_name	varchar(30)	✓
passwd	varchar(30)	✓
specialist	varchar(50)	✓
gender	varchar(10)	✓
email	varchar(50)	✓
contact_no	varchar(50)	✓
country	varchar(20)	✓
appoinmentdate	varchar(20)	✓
		☐

This table is used for storing the details of doctor.

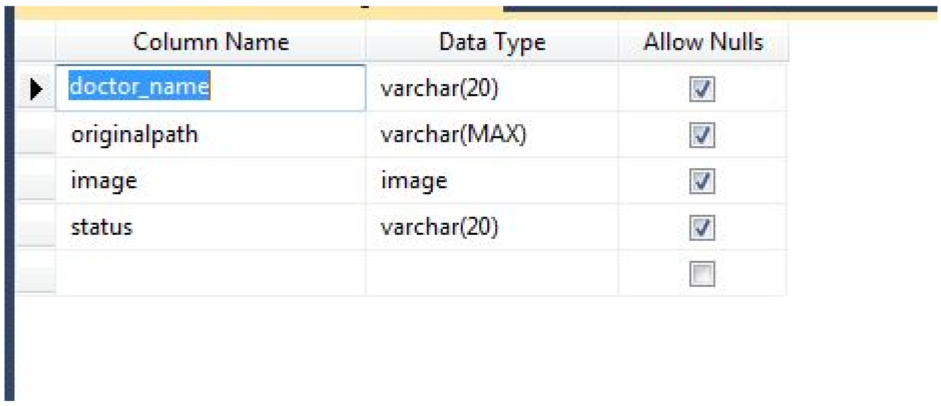
Table - 3.2 Crop Image Table



Column Name	Data Type	Allow Nulls
xaxis	int	✓
yaxis	int	✓
width	int	✓
height	int	✓
cropdata	image	✓
		☐

This table is used for storing the crop Image.

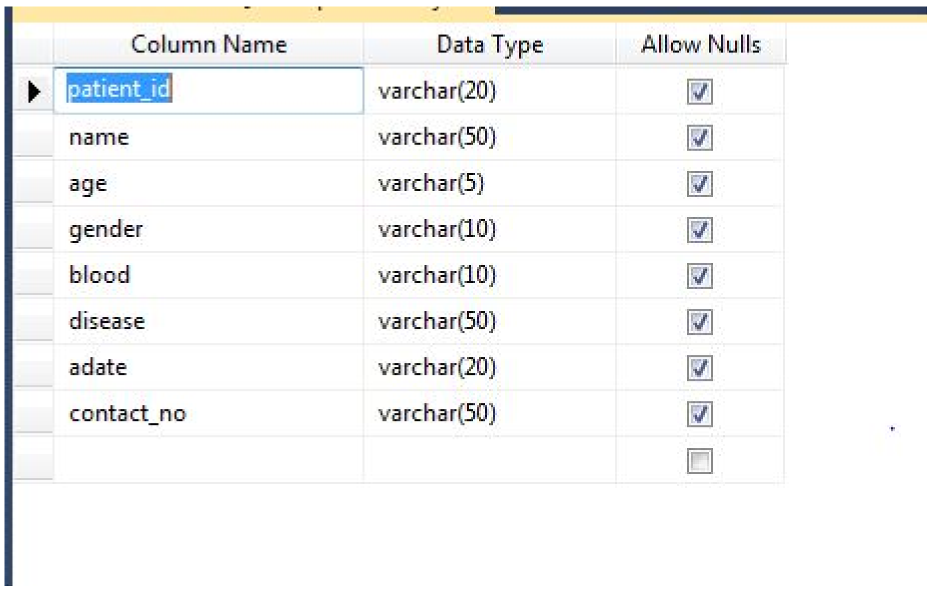
Table - 3.3 Image Details



Column Name	Data Type	Allow Nulls
doctor_name	varchar(20)	✓
originalpath	varchar(MAX)	✓
image	image	✓
status	varchar(20)	✓
		☐

This table is used for storing the image details.

Table - 3.4 Patient Entry Details



Column Name	Data Type	Allow Nulls
patient_id	varchar(20)	✓
name	varchar(50)	✓
age	varchar(5)	✓
gender	varchar(10)	✓
blood	varchar(10)	✓
disease	varchar(50)	✓
adate	varchar(20)	✓
contact_no	varchar(50)	✓
		☐

This table is used for storing the patient details.

4. ANALYSIS

4.1 Introduction to Dotnet

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

".NET" is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

4.1.1 The .NET Framework

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is described as the "execution engine" of .NET. It provides the environment within which programs run.

The most important features are

- ♦ Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- ♦ Memory management, notably including garbage collection.
- ♦ Checking and enforcing security restrictions on the running code.
- ♦ Loading and executing programs, with version control and other such features.
- ♦ The following features of the .NET framework are also worth description:

Manage Code

The code that targets .NET, and which contains certain extra Information - "metadata" - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

Manage Data

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you're using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn't get garbage collected but instead is looked after by unmanaged code.

Common Type System(CTS)

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.

Common Language Specification

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

The Class Library

.NET 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.

4.1.2 Languages Supported By .NET

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft's new language. It's a C-style language that is essentially "C++ for Rapid Application Development". Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State's Perl Dev Kit.

ASP.NET XML WEB SERVICES	Windows Forms
Base Class Libraries	
Common Language Runtime	
Operating System	

Fig 4.1 .Net Framework

4.2 Features of C#

1. C# is a simple, modern, object oriented language derived from C++ and Java.
2. It aims to combine the high productivity of Visual Basic and the raw power of C++.
3. It is a part of Microsoft Visual Studio7.0.
4. Visual studio supports Vb, VC++, C++, Vbscript, Jscript. All of these languages provide access to the Microsoft .NET platform.
5. .NET includes a Common Execution engine and a rich class library.
6. Microsoft's JVM equiv. is Common language run time (CLR).
7. CLR accommodates more than one languages such as C#, VB.NET, Jscript, ASP.NET, C++.
8. Source code --->Intermediate Language code (IL) ---> (JIT Compiler) Native code.
9. The classes and data types are common to all of the .NET languages.
10. We may develop Console application, Windows application, and Web application using C#.
11. In C# Microsoft has taken care of C++ problems such as Memory management, pointers etc.
12.It supports garbage collection, automatic memory management and a lot.

4.2.1 Main Features Of C#

1. Pointers are missing in C#.
2. Unsafe operations such as direct memory manipulation are not allowed.
3. In C# there is no usage of "::" or "->" operators.
4. Since it`s on .NET, it inherits the features of automatic memory management and garbage collection.
5. Varying ranges of the primitive types like Integer, Floats etc.
6. Integer values of 0 and 1 are no longer accepted as Boolean values. Boolean values are pure true or false values in C# so no more errors of "="operator and "=="operator. "==" is used for comparison operation and "=" is used for assignment operation.

4.3 Objectives of .NET

The .net framework is one of the tools provided by the .net platform. It provides an Environment for building, deploying and running web services and other applications like Console applications; Windows based applications, Web sites. It is a Common architecture for all .net programming languages.

The Main Objectives of .NET Framework

1) Platform dependent
2) Language Independent
3) Language Interoperability
4) Security
5) Database Connectivity
6) Globalization of Application

Platform Independent : As dll or exe files are executable in any operating system with the help of the CLR (common language runtime), hence .net is called as platform independent.

Language Independent : As .net application logic can be developed in any .net framework compatible languages, hence it is called as Language Independent.
Specification in ASP.net
It provides set of rules to be followed while integrating with the language.

Language Interoperability: The code written in one language should be used from the application developed using other language.

Security : The .net applications attain high level of security.

Database Connectivity : A new Database connectivity model to connect Database.

Globalization of Application : Designing the applications for supporting multiple languages and cultures.

4.4 Componets of.NET Framework

The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable ASP.NET applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and isolated file storage.

4.5 Common Language Runtime

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions; even if it is being used in the same active application.The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally feature rich.

The runtime also enforces code robustness by implementing a strict type-and-code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

4.5.1 Features Of The Common Language Runtime

Common Language Runtime is a heart of the .net framework. It actually manages the code during Execution. The Code that runs under the CLR is called "Managed Code". The code that is executed under .net runtime gets benefits like cross language inheritance, cross language exception handling, enhanced Security, Versioning and development support, a simplified model for component interaction, debugging and Profiling services.

Automatic memory management: - The CLR provides the Garbage Collection feature for managing the life time of object. This relives a programmer from memory management task.

Standard Type System: - The CLR Implement a formal Specification called the Common Type System (CTS). CTS is important part of rules that ensures that objects written in different language can interact with each other.

Language interoperability: - It is the ability of an application to interact with another application written in a different programming language. Language interoperability helps maximum code reuse. The CLR provides support for language interoperability by specifying and enforcing CTS and by providing metadata.

Platform Independence: - The Compiler compiles code language, which is CPU-independent. This means that the code can be executed from any platform that supports the .Net CLR.

Security Management: - In .net platform, Security is achieved through the code access Security (CAS) model. In the model, CLR enforces the restriction an managed code through the object called "permissions". The CLR allows the code to perform only that task for which it has permissions. In other words, the CAS model specifies what the code can access instead of specifies who can access resources.

Type Safety: - This feature ensures that object is always accessed in compatible ways. Therefore the CLR will prohibit a code from assign a 10-byte value to an object that occupies 8 bytes.

4.5.2 Benefits of CLR

- Performance improvement
- The ability to easily use components developed in other languages.
- Extensible types provided by library.
- New Language features such as inheritance, interfaces etc.
- Complete Object-Oriented design.
- Very Strong Type Safety.
- A good blend of Visual Basic simplicity and c++ power.
- Syntax and keywords similar to c and c++.
- Use of delegates rather than function pointers for increased type safety and security.

4.6 WPF – An Overview

Developed by Microsoft, the Windows Presentation Foundation (or WPF) is a computer-software graphical subsystem for rendering user interfaces in Windows-based applications. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0. Rather than relying on the older GDI subsystem, WPF utilizes DirectX. WPF attempts to provide a consistent programming model for building applications and provides a separation between the user interface and the business logic. It resembles similar XML-oriented object models, such as those implemented in XUL and SVG.

4.6.1 Evolution of WPF

The user interface is an important part of nearly every application. Yet what users expect from those interfaces has advanced significantly. Traditional menu-driven GUIs are still required, of course, but applications may also need to display video, run animations, use two- and three-dimensional graphics, and work with various kinds of documents. And all of this must be possible whether the application is accessed from a standalone desktop client or via a Web browser.

A primary goal of Windows Presentation Foundation (WPF), originally released with the .NET Framework 3.0, is to address this challenge. By offering a consistent technical underpinning for all of these user interface aspects, WPF makes life simpler for developers. By taking a more modern approach, including support for video, animation, two- and three-dimensional graphics, and various kinds of documents, WPF can let users work with information in new ways. And by providing a common foundation for desktop clients and browser clients, WPF makes it easier to build applications that address both.

One challenge that has long faced the creators of user interfaces stems from the different roles required for building effective interfaces. Software developers are needed to create the logic behind the interface, but they're rarely the best people to define the interface's look and feel. Designers, specialists in human/machine interaction, are typically a much better choice for this role. Yet older technologies such as Windows Forms are focused entirely on the developer. There's no truly effective way for developers and designers to collaborate. To address this problem, WPF relies on the Extensible Application Markup Language (XAML). An XML-based language, XAML allows specifying a user interface declaratively rather than in code. This makes it much easier for tools to generate and work with an interface specification based on the visual representation created by a designer. In fact, Microsoft provides Expression Blend to do exactly this. Designers can use this tool (and others provided by third parties) to create the look of an interface, and then have a XAML definition of that interface generated for them. The developer reads this definition into Visual Studio, then creates the logic the interface requires.

When a developer creates a standalone WPF application, one that runs directly on Windows, she has access to everything WPF provides. To create a client that runs inside a Web browser, however, a developer can build a XAML browser application, commonly referred to as an XBAP. Built on the same foundation as a standalone WPF application, an XBAP allows presenting the same style of user interface within a downloadable browser application. The same code can potentially be used for both kinds of applications, which means that developers no longer need different skill sets for desktop and browser clients. As is typical for this kind of rich Internet application, an XBAP downloaded from the Internet runs in a secure sandbox, which limits what the application can do. Still, a large subset of the user interface functionality available to a standalone WPF application can also be used in an XBAP.

4.6.2 Benefits of WPF

First off, WPF is not just for applications which simply require “eye candy.” That is the most common and frustrating misperception about WPF which I’ve encountered. Sure, WPF has a lot of support for flashy visuals and animations. But that’s not all it’s good for. If you’ve worked with WPF for any substantial period of time you are probably well aware of this fact, so I won’t keep harping on the issue.

WPF is an especially great platform to use if your applications involve various media types. For example, if you need to incorporate video, or documents, or 3D content, or animated transitions between a sequence of images, or a combination of any of the above. WPF is also great if you need to create a skinned user interface, or if you need to bind to XML data, or dynamically load portions of a user interface from a Web service, or want to create a desktop application with a Web-like navigation style. Another great reason to use WPF is if you have a team of developers who are bored with WinForms and are itching to get into something new and cool. Of course this is not as powerful and compelling a reason from a business perspective, but nothing promotes employee retention better than keeping the employees interested in their jobs.

WinForms definitely still has a role to play, despite the fact that WPF has hit the scene. If you are building applications with no need for the extensive modern functionality in WPF, then there is no compelling reason to leave behind a time-tested developer-approved platform. WinForms certainly has more 3rd party controls available, online resources, developer communities, etc. than WPF currently does. It’s much easier to find WinForms developers than WPF developers. Also, WinForms currently has a much better design-time experience in Visual Studio than WPF. That fact alone is a very compelling reason to stick with WinForms for a while.

4.7 XAML

XAML stands for Extensible Application Markup Language. It’s a simple language based on XML to create and initialize .NET objects with hierarchical relations. Although it was originally invented for WPF it can be used to create any kind of object trees.

Today XAML is used to create user interfaces in WPF, Silver light, declare workflows in WF and for electronic paper in the XPS standard.

All classes in WPF have parameter less constructors and make excessive usage of properties. That is done to make it perfectly fit for XML languages like XAML.

4.7.1 Advantages of XAML

All you can do in XAML can also be done in code. XAML ist just another way to create and initialize objects. You can use WPF without using XAML. It's up to you if you want to declare it in XAML or write it in code. Declare your UI in XAML has some advantages

- XAML code is short and clear to read
- Separation of designer code and logic
- Graphical design tools like Expression Blend require XAML as source.
- The separation of XAML and UI logic allows it to clearly separate the roles of designer and developer.

What is XAML?

XAML is a declarative markup language. As applied to the .NET Framework programming model, XAML simplifies creating a UI for a .NET Framework application. You can create visible UI elements in the declarative XAML markup, and then separate the UI definition from the run-time logic by using code-behind files, joined to the markup through partial class definitions. XAML directly represents the instantiation of objects in a specific set of backing types defined in assemblies. This is unlike most other markup languages, which are typically an interpreted language without such a direct tie to a backing type system. XAML enables a workflow where separate parties can work on the UI and the logic of an application, using potentially different tools.

When represented as text, XAML files are XML files that generally have the .XAML extension. The files can be encoded by any XML encoding, but encoding as UTF-8 is typical.

4.8 Features Of SQL Server

Microsoft SQL Server 2012

The following is a list of the new features provided in SQL Server 2012:

- Database mirroring
- Database snapshots
- CLR integration
- Service Broker
- Database Mail
- User-defined functions
- Indexed views
- Distributed partitioned views
- INSTEAD OF and AFTER triggers
- New data types
- Cascading RI constraints
- Multiple SQL Server instances
- XML support
- Log shipping

Database mirroring

Database mirroring is a new high-availability feature in SQL Server 2012. It's similar to server clustering in that failover is achieved by the use of a stand-by server; the difference is that the failover is at the database level rather than the server level. The primary database continuously sends transaction logs to the backup database on a separate SQL Server instance. A third SQL Server instance is then used as a witness database to monitor the interaction between the primary and the mirror databases.

Database snapshots

A database snapshot is essentially an instant read-only copy of a database, and it is a great candidate for any type of reporting solution for your company. In addition to being a great reporting tool, you can revert control from your primary database to your snapshot database in the event of an error. The only data loss would be from the point of creation of the database snapshot to the event of failure.

CLR integration

With SQL Server 2008, you now have the ability to create custom .NET objects with the database engine. For example, stored procedures, triggers, and functions can now be created using familiar .NET languages such as VB and C#. Exposing this functionality gives you tools that you never had access to before such as regular expressions.

Service Broker

This feature gives you the ability to create asynchronous, message-based applications in the database entirely through TSQL. The database engine guarantees message delivery, message order consistency, and handles message grouping. In addition, Service Broker gives you the ability to send messages between different SQL Server instances. Server Broker is also used in several other features in SQL Server 2008. For example, you can define Event Nonfictions in the database to send a message to a Queue in the database when someone attempts to alter a table structure, of if there is a string of login failures.

Database Mail

Database Mail, the eventual successor to SQL Mail, is greatly enhanced e-mail solution available in the database engine. With Database Mail, there is no longer a dependency on Microsoft Outlook or MAPI e-mail clients. Database Mail uses standard SMTP to send e-mail messages. These messages may contain query results, attachments (which can be governed by the DBA), and is fully cluster aware. In addition, the e-mail process runs outside of the database engine space, which means that messages can continue to be queued even when the database engine has stopped.

User-Defined Functions

SQL Server has always provided the ability to store and execute SQL code routines via stored procedures. In addition, SQL Server has always supplied a number of built-in functions. Functions can be used almost anywhere an expression can be specified in a query. This was one of the shortcomings of stored procedures—they couldn't be used in line in queries in select lists, where clauses, and so on. Perhaps you want to write a routine to calculate the last business day of the month. With a stored procedure, you have to exec the procedure, passing in the current month as a parameter and returning the value into an output variable, and then use the variable in your queries. If only you could write your own function that you could use directly in the query just like a system function? In SQL Server 2012, you can.

SQL Server 2008 introduces the long-awaited support for user-defined functions. User-defined functions can take zero or more input parameters and return a single value—either a scalar value like the system-defined functions, or a table result. Table-valued functions can be used anywhere table or view expressions can be used in queries, and they can perform more complex logic than is allowed in a view.

Indexed Views

Views are often used to simplify complex queries, and they can contain joins and aggregate functions. However, in the past, queries against views were resolved to queries against the underlying base tables, and any aggregates were recalculated each time you ran a query against the view. In SQL Server 2008 Enterprise or Developer Edition, you can define indexes on views to improve query performance against the view. When creating an index on a view, the result set of the view is stored and indexed in the database. Existing applications can take advantage of the performance improvements without needing to be modified.
Indexed views can improve performance for the following types of queries:

- Joins and aggregations that process many rows
- Join and aggregation operations that are performed frequently within many queries
- Decision support queries that rely on summarized, aggregated data that is infrequently updated

Distributed Partitioned Views

SQL Server 7.0 provided the ability to create partitioned views using the UNION ALL statement in a view definition. It was limited, however, in that all the tables had to reside within the same SQL Server where the view was defined. SQL Server 2012 expands the ability to create partitioned views by allowing you to horizontally partition tables across multiple SQL Servers. The feature helps you scale out one database server to multiple database servers, while making the data appear as if it comes from a single table on a single SQL Server. In addition, partitioned views are now able to be updated.

INSTEAD OF and AFTER Triggers

In versions of SQL Server prior to 7.0, a table could not have more than one trigger defined for INSERT, UPDATE, and DELETE. These triggers only fired *after* the data modification took place. SQL Server 7.0 introduced the ability to define multiple AFTER triggers for the same operation on a table. SQL Server 2008 extends this capability by providing the ability to define which AFTER trigger fires first and which fires last.

SQL Server 2012 also introduces the ability to define INSTEAD OF triggers. INSTEAD OF triggers can be specified on both tables and views. (AFTER triggers can still only be specified on tables.) If an INSTEAD OF trigger is defined on a table or view, the trigger will be executed in place of the data modification action for which it is defined. The data modification is not executed unless the SQL code to perform it is included in the trigger definition.

New Data types

SQL Server 2012 introduces three new data types. Two of these can be used as datatypes for local variables, stored procedure parameters and return values, user-defined function parameters and return values, or table columns:

- Bigint— an 8-byte integer that can store values from $–2^{63}$ (–9223372036854775808) through 2^{63-1} (9223372036854775807).
- sql_variant— a variable-sized column that can store values of various SQL Server-supported data types, with the exception of text, ntext, timestamp, and sql_variant.

The third new data type, the table data type, can be used only as a local variable datatype within functions, stored procedures, and SQL batches. The table datatype cannot be passed as a parameter to functions or stored procedures, nor can it be used as a column datatype. A variable defined with the table data type can be used to store a result set for later processing. A table variable can be used in queries anywhere a table can be specified.

Text in Row Data

In previous versions of SQL Server, text and image data was always stored on a separate page chain from where the actual data row resided. The data row contained only a pointer to the text or image page chain, regardless of the size of the text or image data. SQL Server 2008 provides a new text in row table option that allows small text and image data values to be placed directly in the data row, instead of requiring a separate data page. This can reduce the amount of space required to store small text and image data values, as well as reduce the amount of I/O required to retrieve rows containing small text and image data values.

Cascading RI Constraints

In previous versions of SQL Server, referential integrity (RI) constraints were restrictive only. If an insert, update, or delete operation violated referential integrity, it was aborted with an error message. SQL Server 2008 provides the ability to specify the action to take when a column referenced by a foreign key constraint is updated or deleted. You can still abort the update or delete if related foreign key records exist by specifying the NO ACTION option, or you can specify the new CASCADE option, which will cascade the update or delete operation to the related foreign key records.

Multiple SQL Server Instances

Previous versions of SQL Server supported the running of only a single instance of SQL Server at a time on a computer. Running multiple instances or multiple versions of SQL Server required switching back and forth between the different instances, requiring changes in the Windows registry. (The SQL Server Switch provided with 7.0 to switch between 7.0 and 6.5 performed the registry changes for you.)

SQL Server 2005 provides support for running multiple instances of SQL Server on the same system. This allows you to simultaneously run one instance of SQL Server 6.5 or 7.0 along with one or more instances of SQL Server 2008. Each SQL Server instance runs independently of the others and has its own set of system and user databases, security configuration, and so on. Applications can connect to the different instances in the same way they connect to different SQL Servers on different machines.

XML Support

Extensible Markup Language has become a standard in Web-related programming to describe the contents of a set of data and how the data should be output or displayed on a Web page. XML, like HTML, is derived from the Standard

Generalize Markup Language (SGML). When linking a Web application to SQL Server, a translation needs to take place from the result set returned from SQL Server to a format that can be understood and displayed by a Web application. Previously, this translation needed to be done in a client application.
SQL Server 2008 provides native support for XML. This new feature provides the ability to do the following:

- Return query result sets directly in XML format.
- Retrieve data from an XML document as if it were a SQL Server table.
- Access SQL Server through a URL using HTTP. Through Internet Information Services (IIS), you can define a virtual root that gives you HTTP access to the data and XML functionality of SQL Server 2008.

Log Shipping

The Enterprise Edition of SQL Server 2012 now supports log shipping, which you can use to copy and load transaction log backups from one database to one or more databases on a constant basis. This allows you to have a primary read/write database with one or more read-only copies of the database that are kept synchronized by restoring the logs from the primary database. The destination database can be used as a warm standby for the primary database, for which you can switch users over in the event of a primary database failure. Additionally, log shipping provides a way to offload read-only query processing from the primary database to the destination database.

This capability was available in previous versions of SQL Server, but it required the DBA to manually set up the process and schedule the jobs to copy and restore the log backups. SQL Server 2012 officially supports log shipping and has made it easier to set up via the Database Maintenance Plan Wizard. This greatly simplifies the process by automatically generating the jobs and configuring the databases to support log shipping.

DDL triggers

In previous articles, I outlined how you can use data definition language (DDL) triggers in SQL Server 2012 to implement custom database and server auditing solutions for Sarbanes-Oxley compliance (here are part one and part two of my SOX articles). DDL triggers are defined at the server or database level and fire when DDL statements occur. This gives you the ability to audit when new tables, stored procedures, or logins are created.

Ranking functions

SQL Server 2012 provides you with the ability to rank result sets returned from the database engine. This allows you to customize the manner in which result sets are returned, such as creating customized paging functions for Web site data.

Row versioning-based isolation levels

This new database engine feature improves database read concurrency by reducing the amount of locks being used in your database. There are two versions of this feature (both of which must be enabled at the database level):

- **Read Committed Isolation Using Row Versioning** is used at the individual statement level, and guarantees that the data is consistent for the duration of the statement.
- **Snapshot Isolation** is used at the transaction level, and guarantees that the data is consistent for the duration of the transaction.

5. IMPLEMENTATION AND TESTING

5.1 Sample code

DAL

```
namespace DAL
{
public class DalProcess
{
SqlConnection connect = new SqlConnection("Data Source=.;Initial Catalog=Data_Hiding;Integrated Security=True");
SqlCommand cmd = new SqlCommand();
SqlDataReader dr;
SqlDataAdapter da;
BO.PatientRegister PR = new BO.PatientRegister();
BO.DoctorRegister DR = new BO.DoctorRegister();
BO.ReserveRoom RR = new BO.ReserveRoom();
public string DalPatientCode()
{
int a;
try
{
connect.Open();
cmd = new SqlCommand("patient_code", connect);
cmd.CommandType = CommandType.StoredProcedure;
a = Convert.ToInt32(cmd.ExecuteScalar());
PR.Patient_Code = a.ToString();
return PR.Patient_Code;
}
catch (Exception)
{
throw;
 }
Finally{
```

```
connect.Close();
}
}
public bool DalPatientDetails(BO.PatientRegister PR)
{
int a = 0;
try
{
connect.Open();
cmd = new SqlCommand("patient_details", connect);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@patient_id", PR.Patient_ID);
cmd.Parameters.AddWithValue("@name", PR.Patient_Name);
cmd.Parameters.AddWithValue("@age", PR.Age);
cmd.Parameters.AddWithValue("@gender", PR.Gender);
cmd.Parameters.AddWithValue("@blood", PR.Blood_Group);
cmd.Parameters.AddWithValue("@disease", PR.Disease);
cmd.Parameters.AddWithValue("@adate", PR.Admit_Date);
cmd.Parameters.AddWithValue("@contact_no", PR.Contect_No);
a = cmd.ExecuteNonQuery();
}
catch(Exception)
{
throw;
}
finally
{
connect.Close();
}
return a > 0 ? true : false;
}
public bool DalDoctorDetails(BO.DoctorRegister DR)
{
int a = 0;
```

```
try
{
connect.Open();
cmd = new SqlCommand("doctor_details", connect);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@doctor_name",DR.Doctor_Name);
cmd.Parameters.AddWithValue("@passwd", DR.Password);
cmd.Parameters.AddWithValue("@specialist",DR.Specialist);
cmd.Parameters.AddWithValue("@gender",DR.Gender);
cmd.Parameters.AddWithValue("@email",DR.Email);
cmd.Parameters.AddWithValue("@contact_no",DR.Contact_No);
cmd.Parameters.AddWithValue("@country",DR.Country);
cmd.Parameters.AddWithValue("@appoinmentdate", DR.Appoinment_Date);
cmd.Parameters.AddWithValue("@mode", "register");
a = cmd.ExecuteNonQuery();
}
catch (Exception)
{
throw;
}
finally
{
connect.Close();
}
return a > 0 ? true : false;
}
public ArrayList DalGetReserveDetail()
{
byte[] empty=new byte[1024*5000];
int a = 0;
ArrayList objar = new ArrayList();
try
{
connect.Open();
```

```
cmd = new SqlCommand("cropimagevalues", connect);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@mode", "select");
cmd.Parameters.AddWithValue("@xaxis", "");
cmd.Parameters.AddWithValue("@yaxis", "");
cmd.Parameters.AddWithValue("@width", "");
cmd.Parameters.AddWithValue("@height", "");
cmd.Parameters.AddWithValue("@cropdata",empty);
dr = cmd.ExecuteReader();
if (dr.Read())
{
objar.Add(dr[0]);
objar.Add(dr[1]);
objar.Add(dr[2]);
 objar.Add(dr[3]);
}
return objar;
}
catch (Exception)
{
throw;
}
finally
{
connect.Close();
}
public bool DalImageSend(BO.DetailsSend DS)
{
int a = 0;
try
{
connect.Open();
cmd = new SqlCommand("senddetails", connect);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@mode", "insert");
cmd.Parameters.AddWithValue("@doctor_name", DS.Doctor_Name);
cmd.Parameters.AddWithValue("@originalpath", DS.Original);
cmd.Parameters.AddWithValue("@image",DS.Image);
cmd.Parameters.AddWithValue("@status", "Pending");
a = cmd.ExecuteNonQuery();
return a > 0 ? true : false;
}
catch (Exception)
{
throw;
}
finally
{
connect.Close();
}
}
```

Recieve data

```
namespace ITDIP01Receiver
{
public partial class ReceiveData : Form
{
public ReceiveData()
{
InitializeComponent();
}
public string DoctorName;
BO.ReceiveData RD = new BO.ReceiveData();
BAL.BalProcess ObjBal = new BAL.BalProcess();
server objserv = new server();
string x, y, width, height;
public ReceiveData(string name)
{
InitializeComponent();
```

```
DoctorName = name;
}
private void ReceiveData_Load(object sender, EventArgs e)
{
}
DataSet ds = new DataSet();
private void btnListen_Click(object sender, EventArgs e)
{
RD.Doctor_Name = DoctorName;
byte[] image = ObjBal.BalGetImage(RD);
TypeConverter tc = TypeDescriptor.GetConverter(typeof(Bitmap));
Bitmap img = (Bitmap)tc.ConvertFrom(image);
pbox_receivedata.BackgroundImage = img;
objserv.receive();
}
public Bitmap receiveimage;
private void btn_extract_Click(object sender, EventArgs e)
{
RD.Doctor_Name = DoctorName;
receiveimage=(Bitmap)pbox_receivedata.BackgroundImage;
txt_extract.Text = DataExtract.ExtractText(receiveimage);
ImageDecrypt(250, 250);
string original = ObjBal.BalGetValue(RD);
Bitmap image = new Bitmap(original);
pbox_original.BackgroundImage = image;
}
public Bitmap ImageDecrypt(int width, int height)
{
Bitmap fBmp = new Bitmap(width, height);
Random r = new Random(123);
for (int x = 0; x < width; x++)
{
for (int y = 0; y < height; y++)
{
 int num = r.Next(0, 256);
fBmp.SetPixel(x, y, Color.FromArgb(255, num, num, num));
}
}
return fBmp;
}
public class server
{
IPEndPoint IPEnd;
Socket Sock;
public int Count = 0;
public string[] da = new string[4];
public server()
{
try
{
PEnd = new IPEndPoint(IPAddress.Any, 8687);
Sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.IP);
Sock.Bind(IPEnd);
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
}
public byte[] receive()
{
try
{
Sock.Listen(1000);
Socket clientsock = Sock.Accept();
byte[] ClientData = new byte[1024 * 500000];
int receivedbyte = clientsock.Receive(ClientData);
int datalen = BitConverter.ToInt32(ClientData, 0);
```

```
 if(datalen!=0)
 {
 MessageBox.Show("Image Received");
 }
 else
 {
 MessageBox.Show("You Dont Have Message");
 }
```

Data hiding

```
namespace ITDIP01
{
public class DataHiding
{
enum State
{
hiding,
filling_with_zeros
};
public static int reverseBits(int n)
{
int result = 0;
for (int i = 0; i < 8; i++)
{
result = result * 2 + n % 2;
n /= 2;
}
return result;
}
```

Data Send

```
namespace ITDIP01
{
public partial class DataSend : Form
{
//Declaration
```

```
public readonly string ss;
BAL.BalProcess ObjBal = new BAL.BalProcess();
BO.DetailsSend DS = new BO.DetailsSend();
BO.ReserveRoom RR = new BO.ReserveRoom();
DAL.DalProcess obj_dal = new DAL.DalProcess();
string clientsystem;
Bitmap embed = null;
string originalpath;
public DataSend()
{
InitializeComponent();
}
public DataSend(string s)
{
InitializeComponent();
ss = s;
}
Activenodeclass objnode = new Activenodeclass();
private void DataSend_Load(object sender, EventArgs e)
{
lbl_detail.Visible = false;
gridview_bind.Visible = false;
lbl_doctorname.Visible = false;
cbox_dname.Visible = false;
btn_browse.Visible = false;
btn_send.Visible = false;
lbl_filename.Visible = false;
pbox_choose.Visible = false;
lbl_client.Visible = false;
cbox_name.Visible = false;
pnl_reserve.Visible = false;
private void llbl_reservepoint_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
```

```
lbl_detail.Visible = false;
gridview_bind.Visible = false;
lbl_doctorname.Visible = false;
cbox_dname.Visible = false;
btn_browse.Visible = false;
btn_send.Visible = false;
lbl_filename.Visible = false;
pbox_choose.Visible = false;
lbl_client.Visible = true;
cbox_name.Visible = true;
pnl_reserve.Visible = true;
}
```

Encryption

```
public static class Decryption
{
public static string Decrypt(string input, string key)
{
byte[] inputArray = Convert.FromBase64String(input);
TripleDESCryptoServiceProvider tripleDES = new
TripleDESCryptoServiceProvider();
tripleDES.Key = UTF8Encoding.UTF8.GetBytes(key);
tripleDES.Mode = CipherMode.ECB;
tripleDES.Padding = PaddingMode.PKCS7;
ICryptoTransform cTransform = tripleDES.CreateDecryptor();
byte[] resultArray = cTransform.TransformFinalBlock(inputArray, 0,
inputArray.Length);
tripleDES.Clear();
return UTF8Encoding.UTF8.GetString(resultArray);
```

Doctor registration

```
namespace ITDIP01
{
public partial class DoctorRegister : Form
{
public DoctorRegister()
```

```
{
InitializeComponent();
}
BO.DoctorRegister DR = new BO.DoctorRegister();
BAL.BalProcess ObjBal = new BAL.BalProcess();
private void DoctorLogin_Load(object sender, EventArgs e)
{
}
private void llbl_home_LinkClicked(object sender, LinkLabelLinkClickedEventArgs
e)
{
this.Hide();
HomePage HP = new HomePage();
HP.Show();
}
```

Embed data

```
namespace ITDIP01
{
public partial class EmbedData : Form
{
BAL.BalProcess ObjBal = new BAL.BalProcess();
BO.PatientRegister PR = new BO.PatientRegister();
BO.ReserveRoom RR = new BO.ReserveRoom();
public string f2;
public EmbedData(string f1)
{
InitializeComponent();
f2 = f1;
}
public EmbedData()
{
InitializeComponent();
}
```

Data Extract

```
namespace ITDIP01Receiver
{
public class DataExtract
{
enum State
{
hiding,
filling_with_zeros
};
public static string ExtractText(Bitmap bmp)
{
int colorUnitIndex = 0;
int charValue = 0;
string extractedText = string.Empty;
```

Reserve Point

```
namespace ITDIP01
{
public partial class ReceiveData : Form
{
string embedpath, originalpath;
BO.DetailsSend DS = new BO.DetailsSend();
BAL.BalProcess ObjBal = new BAL.BalProcess();
public ReceiveData()
{
InitializeComponent();
}
private void ReceiveData_Load(object sender, EventArgs e)
{
lbl_getvalue.Visible = false;
lbl_getvalue.Text = DoctorsLogin.passvalue;
DS.Doctor_Name = lbl_getvalue.Text;
DataSet ds=new DataSet();
try
```

```
{
ds= ObjBal.BalGetImage(DS);
if (ds.Tables[0].Rows.Count!= null && ds.Tables[0].Rows.Count!=0)
{
embedpath = ds.Tables[0].Rows[0].ItemArray.GetValue(0).ToString();
originalpath = ds.Tables[0].Rows[0].ItemArray.GetValue(1).ToString();
}
else
{
MessageBox.Show("You Dont have Details");
}
}
catch (Exception)
```

Patient details

```
namespace ITDIP01
{
public partial class PatientDetails : Form
{
public PatientDetails()
{
InitializeComponent();
}
BAL.BalProcess ObjBal = new BAL.BalProcess();
BO.PatientRegister PR = new BO.PatientRegister();
public string patient_id;
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs
e)
.{
 Environment.Exit(0);
 }
```

5.2 Testing

5.2.1 General

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

5.2.2 Developing Methodologies

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

5.2.3 Types of Tests

1. Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

3. System Testing

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

4. Performance Testing

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

5. Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

6. Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updating process

7. Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

5.3 Test Cases

Table – 5.1 Patient Registration

Test Scenario	**Test Steps**	**Test Data**	**Expected Result**	**Success/Failure**
Check patient Registration with valid input	Enter Name Age Gender Blood Group Disease Admit Date Contact No	 Aaaa 23 Female O+ General 28/03/2018 987654123	Successful registration on valid input	Success
Check patient Registration with invalid input	Enter Name Age Gender Blood Group Disease Admit Date Contact No	 Qwe123 Abc Male A+ General 23/09/2017 23gty45667	Invalid entries	Fail

Table – 5.2 Patient login

Test scenario	**Test steps**	**Test Data**	**Expected Result**	**Success/failure**
Check patient login with valid input	Enter Patient_ID	Patient-0012	Successful login on valid input	Success
Check patient login with invalid input	Enter Patient_ID	Aaaa123	Enter valid entries	Fail

Table -5.3 Doctor Registration

Test Scenario	**Test Steps**	**Test Data**	**Expected Result**	**Success/Failure**
Check Doctor Registration with valid input	Enter Doctor Name Password Specialist Gender Email ID Contact No Country Appointment Date	 Sri 23 Cardio Male Sri@gmail.com 9876541235 India 29/03/2018	 Successful registration on valid input	 success
Check Doctor Registration with invalid input	Enter Doctor Name Password Specialist Gender Email ID Contact No Country Appointment Date	 abc123 Abc cardio Male Abc.com 9872@56767 2uf 23gty45667	 Invalid entries	 Fail

Table - 5.4 Doctor login

Test scenario	**Test steps**	**Test Data**	**Expected Result**	**Success/failure**
Check Doctor login with valid input	Enter Name Password	 Sri 12345678	Successful login on valid input	Success
Check Doctor login with invalid input	Enter Name Password	 Sri123 Ad123	 Enter valid entries	 Fail

Table – 5.5 Admin Login

Test scenario	**Test steps**	**Test Data**	**Expected Result**	**Success/failure**
Check Admin login with valid input	Enter Name Password	 Admin Admin	Successful login on valid input	Success
Check Admin login with invalid input	Enter Name Password	 Sri123 Ad123	Enter valid entries	Fail

6. RESULTS

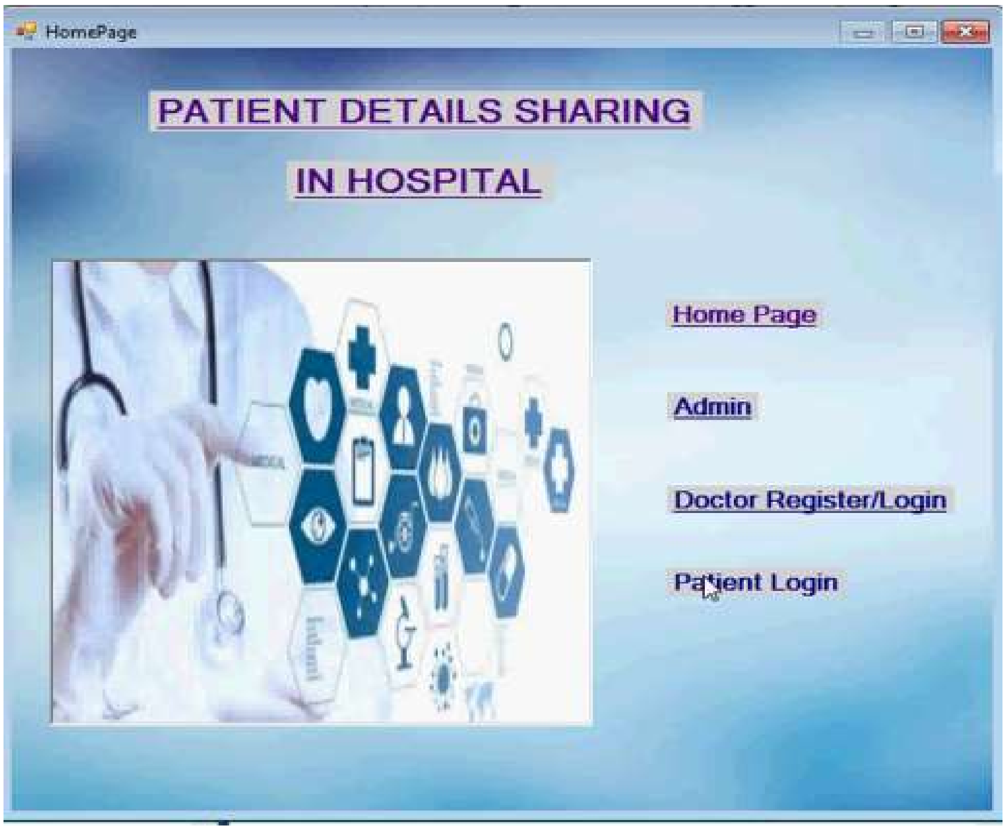
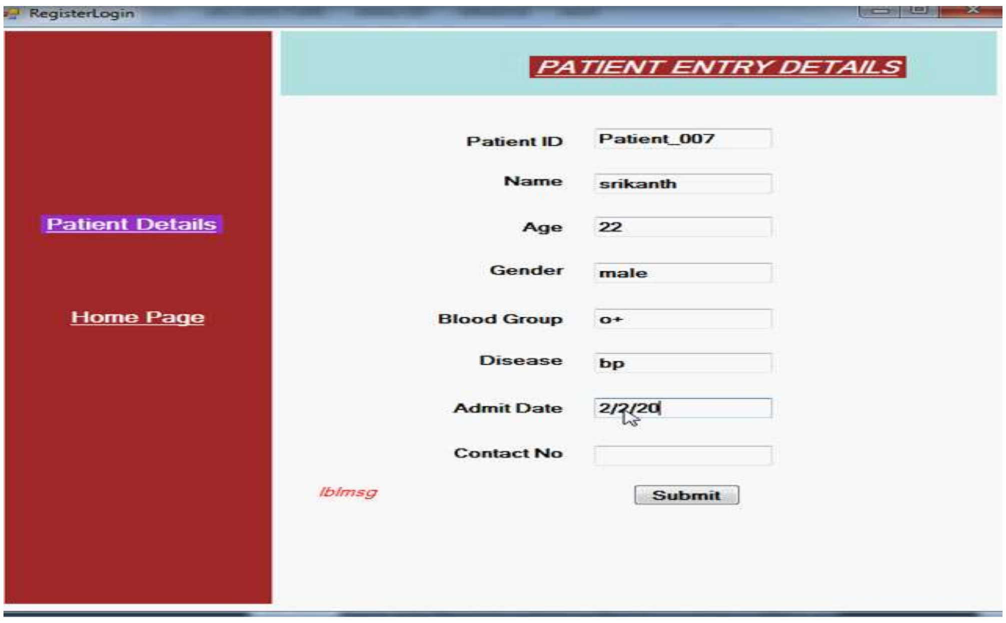


Fig 6.1 Home page

Description

This is an home page where a new patient can register himself. If the patient is authorized person, he can login and update details. Admin can login and perform necessary actions. Even a new doctor can register and the doctor who already exists can login.


RegisterLogin

PATIENT ENTRY DETAILS

Patient Details

Home Page

Patient ID	Patient_007
Name	srikanth
Age	22
Gender	male
Blood Group	o+
Disease	bp
Admit Date	2/2/20
Contact No	

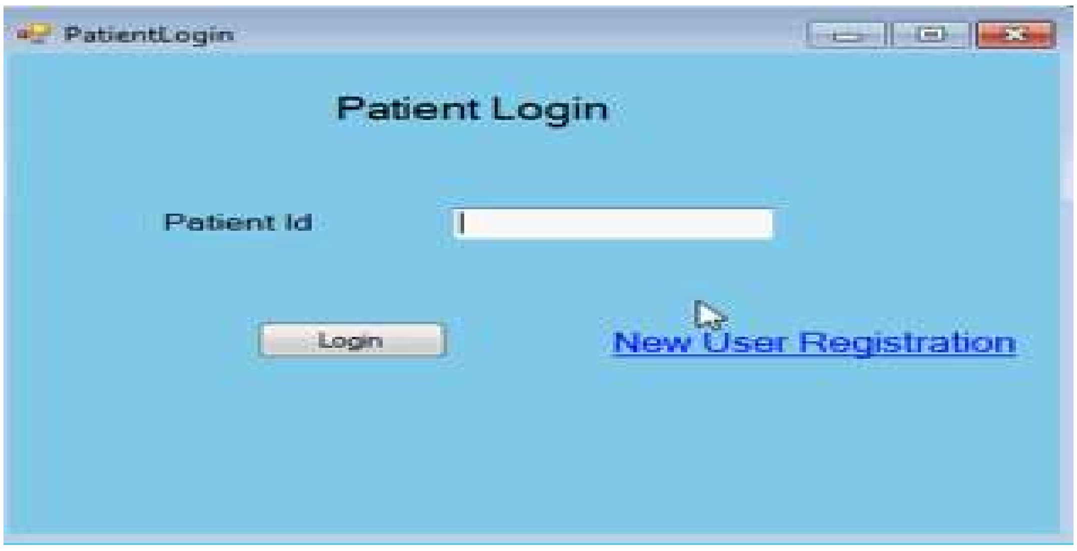
lblmsg

Submit

Fig 6.2 Patient Registration

Description

If an patient is new one ,he has to enter all the necessary details and get registered. Once the registration is completed with all valid details the patient can then login


PatientLogin

Patient Login

Patient Id

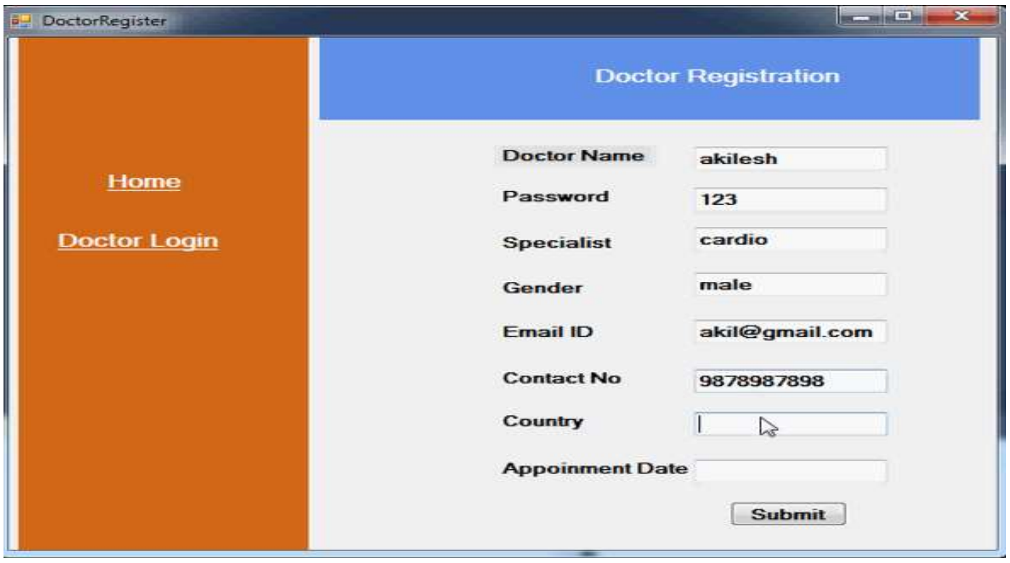
Login

New User Registration

Fig 6.3 patient login

Description

If the registration is successful then the patient has access to login in and perform the task. Otherwise patient has to first register.

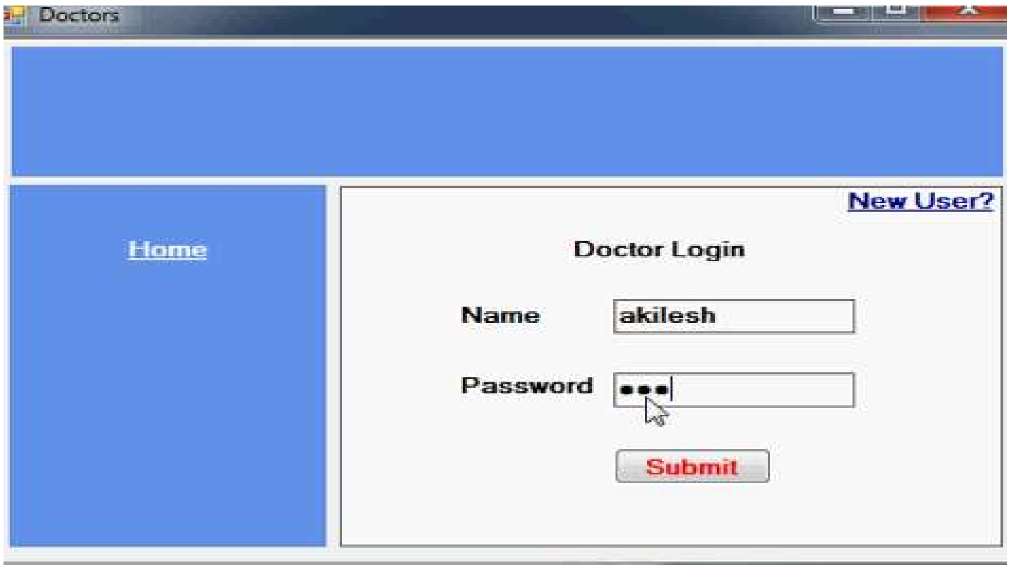


Doctor Name	akilesh
Password	123
Specialist	cardio
Gender	male
Email ID	akil@gmail.com
Contact No	9878987898
Country	
Appoinment Date	

Fig 6.4 Doctor Registration

Description

If an Doctor is new person, he has to enter all the necessary details and get registered. Once the registration is completed with all valid details the Doctor can then login



Name	akilesh
Password	●●●

Fig 6.5 Doctor Login

Description

If the registration is successful then the Doctor has access to login in and perform the task. Otherwise Doctor has to first register.

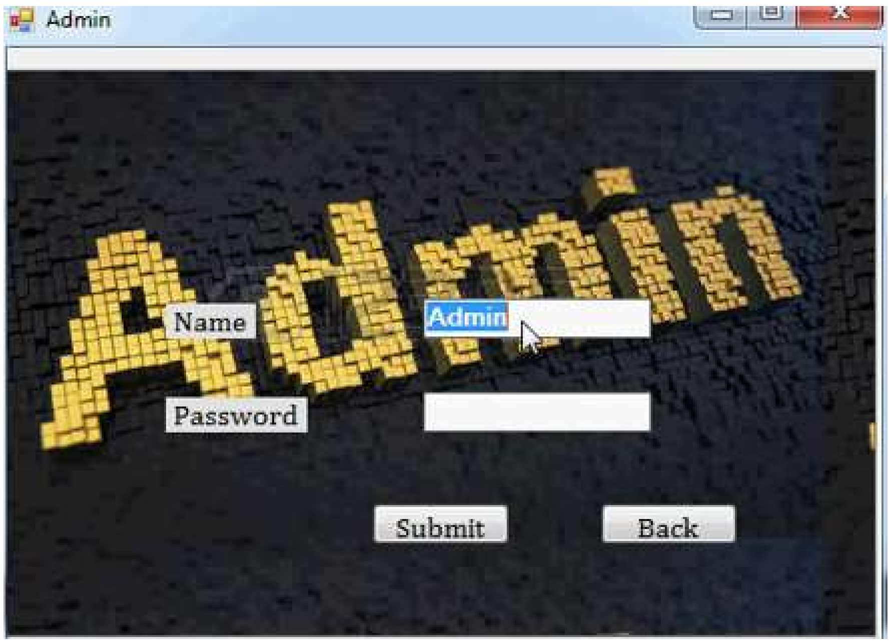


Fig 6.6 Admin Login

Description

Admin can login by entering valid inputs. There is no need of registration for admin.

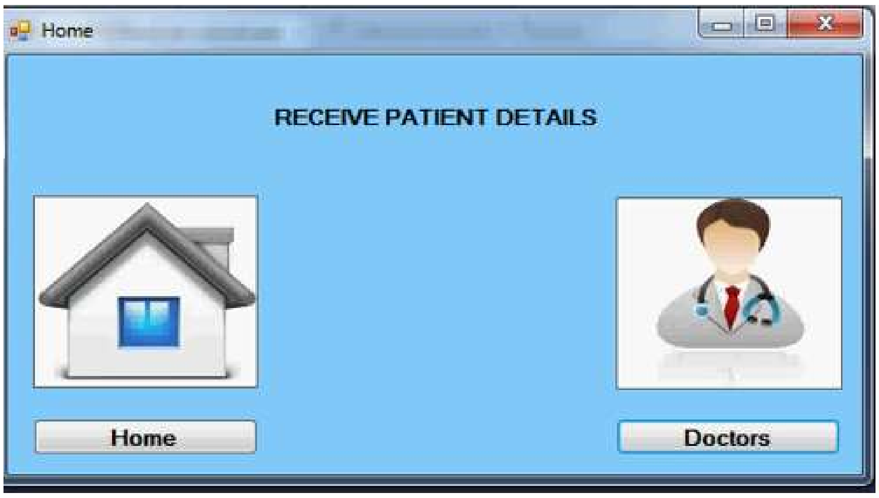
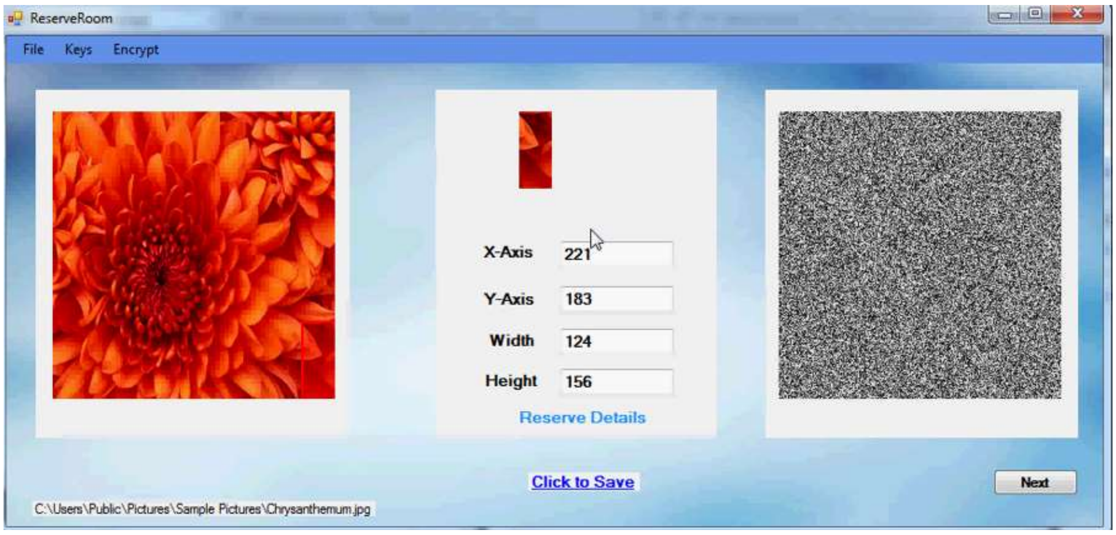


Fig 6.7 Home Page

Description

This is an home page of doctor where a person who is already registered can login directly. If he is an new user then the person has to register.



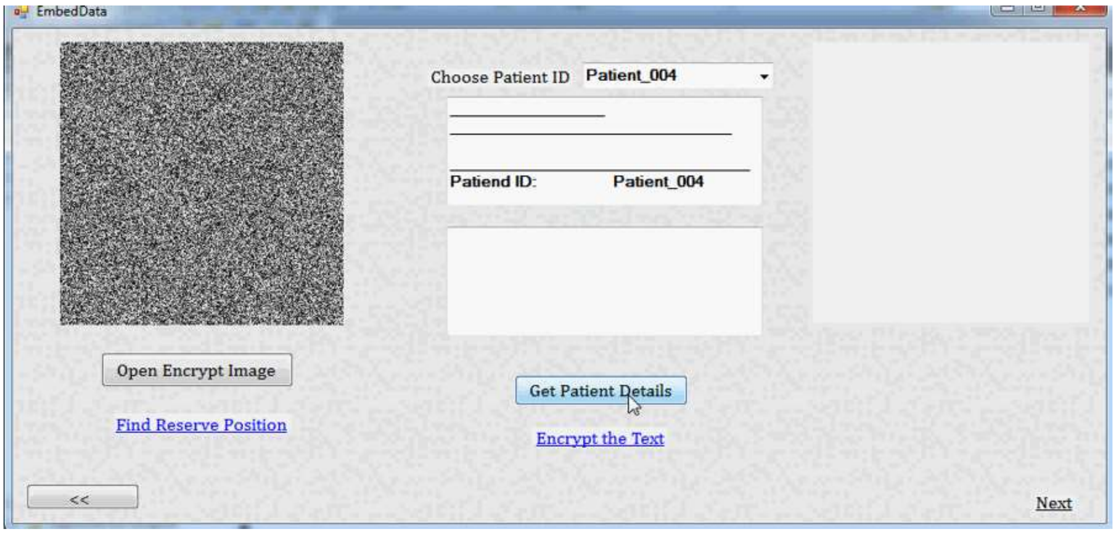
X-Axis 221
Y-Axis 183
Width 124
Height 156

C:\Users\Public\Pictures\Sample Pictures\Chrysanthemum.jpg

Fig 6.8 Encrypt data

Description

Once an image is choosen some part of image is selected for hiding the data . The reserve path is found and the image is encrypted by generating an key. The data is embedded to the encrypted image. The images are saved for futher decryption of data and image.



Choose Patient ID Patient_004

Patiend ID: Patient_004

Fig 6.9 Embeded data

Description

After the Reserve point is found and by using embedded image the patient details are decrypted.

7. CONCLUSION

7.1 Conclusion

The proposed framework reduces the design of secure steganography in empirical covers to the problem of finding local potentials for the distortion function that correlate with statistical delectability in practice. By working out the proposed methodology in detail for a specific choice of the distortion function, we experimentally validate the approach and discuss various options available to the steganographer in practice.

7.2. Application.

SkipCloud

Skip Cloud Multimedia content is complex to interact with for several reasons, in particular because media components such as images, audio and video offer distinct interaction affordances. When interacting with photos or music assets, for instance, users may have to deal with huge collections or with many separate collections.

7.3 Future Enhancement

The CPP rule considers a combination of several existing methods instead of remaining fixed on a single method. An essential principle in selecting candidate algorithms for the CPP rule is that basic methods have comparable security performances. In addition, the CPP rule provides a novel tool for designing steganographic schemes.The advantages of the CPP rule includes the Thermodynamic integration and the Rate-distortion bound.

In our future work, applying the CPP rule to other covers such as JPEG image, videos, and text, is an interesting direction

REFERENCES

[1] Yam bern Jina Chanu, ThemrichonTuithung, Kh. Manglem Singh "A Short Survey on Image Steganography and Steganalysis Techniques".

[2] Kejiang Chen, Weiming Zhang, Hang Zhou, Nenghai Yu "Defining Cost Functions for Adaptive Steganography at the Micro scale".

[3] Tomáˇs Filler, Jessica Fridrich "Gibbs Construction in Steganography".

[4] Jessica Fridrich, Jan Kodovský "Rich Models for Steganalysis of Digital Images".

[5] Vojtˇech Holub and Jessica Fridrich "Designing Steganographic distortion using directional filters"

[6] B. Li, J. He, J.w. Huang, and Y. Q. Shi , "A survey on image steganography and steganalysis," Journal of Information Hiding and Multimedia Signal Processing, vol. 2, no. 2, pp. 142-172, 2011.

[7] J. Fridrich, Steganography in Digital Media: Principles, Algorithms and Applications. Cambridge University Press, 2009.

[8] J. Fridrich and J. Kodovsk´y, "Rich models for steganalysis of digital images ," IEEE Trans. on Inf. Forensics Security, vol. 7, pp. 868-882, Jun. 2012.

[9] T. Pevn´y, T.Filler, and T. Bas, "Using high-dimensional image models to perform highly undetectable steganography," Proc. of International Workshop on Information Hiding, vol. LNCS 6387, pp. 161-177, Jun. 28-30, 2010.

[10] T. Pevn´y, P. Bas, and J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix," IEEE Trans. on Inf. Forensics and Security , vol. 5, no. 2, pp. 215-224, Jun. 2010.

[11] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich, “Selection-channel-aware rich model for steganalysis of digital images,” Proc. of 6th IEEE International Workshop on Information Forensics and Security, Atlanta, GA, USA, Dec. 3-5, 2014

[12] V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” Proc. of IEEE Workshop on Information Forensic and Security, pp. 234-239, Dec. 2-5, 2012.

[13] W. X. Tang, H. D. Li, W. Q. Lou, and J. W. Huang, “Adaptive Steganalysis Based on Embedding Probabilities of Pixels,” IEEE Trans. on Inf. Forensics Security, vol. 11(4), pp. 734-745, Apr. 2016.