

# **VISUALIZING AND PREDICTING HEART DISEASES WITH AN INTERACTIVE DASH BOARD**

*A Project report submitted to the Bharathidasan University, Tiruchirappalli in  
partial fulfillment of the requirements for the award of the degree of*

**M.Sc. COMPUTER SCIENCE**

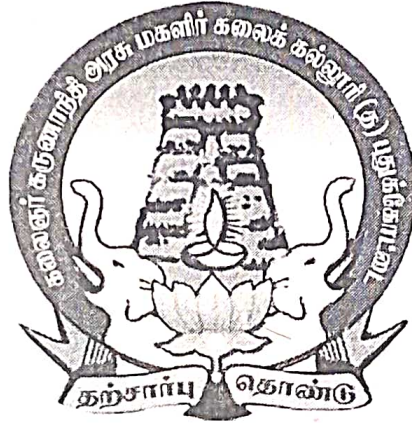
*Submitted by*

**S.KOWSALYA**

**Reg.No:P21S2627**

*Under the Guidance of*

**Dr.S.YASODHA M.C.A.,M.Phil.,Ph.D.**



**DEPARTMENT OF COMPUTER SCIENCE**

**KALAIIGNAR KARUNANIDHI GOVERNMENT ARTS**

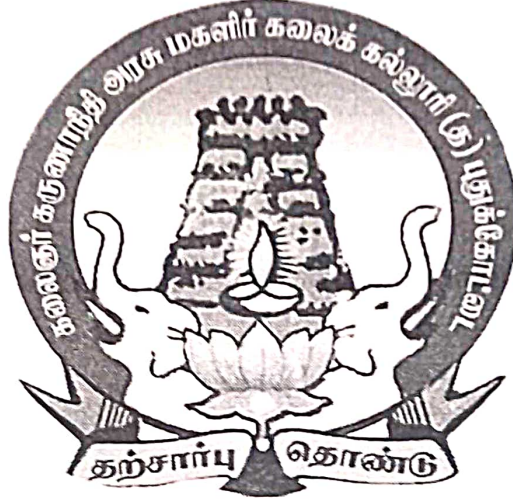
**COLLEGE FOR WOMEN (AUTONOMOUS)**

(Re-accredited with B++ Grade by NAAC)

**PUDUKKOTTAI – 622001**

**2022 – 2023**

**KALAINGAR KARUNANIDHI GOVERNMENT ARTS COLLEGE  
FOR WOMEN (AUTONOMOUS)**  
(Re-accredited with B++ Grade by NAAC)  
**PUDUKKOTTAI – 622001**



**CERTIFICATE**

This is to certify that the project work entitled “VISUALIZING AND PREDICTING HEART DISEASES WITH AN INTERACTIVE DASH BOARD” Submitted by S.KOWSALYA (Reg. No: P21S2627) in partial fulfillment of the requirements for the award of the degree of M.Sc., COMPUTER SCIENCE has been carried out under my guidance and supervision during the academic year 2022 – 2023.

S. Gnanajothi 27/3/23

HEAD OF THE DEPARTMENT

S. GNANAJOTHI, M.C.A., M.Phil.,

Associate Professor and Head,

Department of Computer Science,  
Kalaingar Karunanidhi Govt. Arts College  
for Women (Au), Pudukkottai - 622 001.

Wanodp 27/3/23  
INTERNAL GUIDE

EXTERNAL EXAMINER

1. 

2. Wanodp 5/4/23

## ACKNOWLEDGEMENT

At the outset, I pray “ **THE ALMIGHTY** ” for giving me the strength and energy to complete the course and this project work.

I would like to express my sincere thanks to our most honorable and beloved **Principal Dr. B. BHUVANESWARI, M.Com., M.Phil.,M.B.A., Ph.D.,** for her extensive leadership, support, valuable thoughts and advice rendered throughout my career.

I wish to express my deep sense of gratitude to respect **Head of the department, Mrs. S.GNANAJOTHI, M.C.A., M.Phil.,**Associate professor, Department of Computer Science , Kalaingar Karunanidhi Government Arts College for Women(Autonomous), Pudukkottai, for her concern as well as guidance during my period of study and completion of this project work.

My hearty and sincere thanks to my guide **Dr. S. YASOTHA.,M.C.A., M.Phil., Ph.D.,**Associate Professor, Department of Computer Science, Kalaingar Karunanidhi Government Arts College for Women(Autonomous), Pudukkottai, who guided and encouraged me in every endeavor by offered valuable suggestions and extended kind support throughout the progress of this work.

I record my sincere thanks to all the **Staff Members** and **Programmer** of the Department of Computer Science, Kalaingar Karunanidhi Government Arts College for Women (Autonomous), Pudukkottai, for their motivation and support.

I am very much thankful to my parents and friends for their valuable support and encouragement in completing the project successfully.

**S.KOWSALYA**

# CONTENT

S.NO	TITLE		PAGE NO.
	ABSTRACT		1
1	INTRODUCTION		2
2	SYSTEM SPECIFICATION		
	2.1	Hardware Requirements	4
	2.2	Software Specification	4
	2.3	Software Requirements	
		2.2.1 C#.NET	5
		2.2.2 SQL SERVER	10
3	SYSTEM ANALYSIS		
	3.1	Existing System	15
	3.2	Proposed System	16
4	SYSTEM DESIGN		
	4.1	Module Description	17
	4.2	Data Flow Diagram	21
	4.3	Table Design	27
5	SYSTEM TESTING		
6	SYSTEM IMPLEMENTATION		
	6.1	Source Code	32
	6.2	Screen Layout	56
7	FUTURE ENHANCEMENT		63
8	CONCLUSION		64
	BIBLIOGRAPHY		65

# ABSTRACT

## ABSTRACT

This project entitled “**Visualizing And Predicting Heart Diseases With An Interactive Dash Board**” has been developed under **C#.NET** as frontend and **SQL** backend.

Heart is one of the most vital structures in the human body. It is the center of the circulatory system. Heart disease is a main life intimidating disease that can cause either death or a severe long term disability. However, there is lack of effective tools to discover hidden relationships and trends in e-health data. Medical diagnosis is a complex task and plays a dynamic role in saving human lives so it needs to be executed accurately and efficiently. A suitable and precise computer based automated decision support system is required to reduce cost for achieving clinical tests. Health analytics have been proposed using ML machine learning to predict accurate patient data analysis. The data produced from health care industry is not mined. Data mining techniques can be used to build an intelligent model in medical field using data sets which involves risk factor of patients. The knowledge discovery in database (KDD) is started with development of approaches and techniques for making use of data.

This project provides an insight into deep learning and machine learning techniques used in diagnosing diseases. Numerous data mining classifiers have been conversed which has emerged in recent years for efficient and effective disease diagnosis. This project proposes a heart attack prediction system using Deep learning techniques, specifically Multi-Layer Perceptron (*MLP*) to predict the likely possibilities of heart related diseases of the patient. MLP is a very powerful classification algorithm that makes use of Deep Learning approach in Artificial Neural Network. The proposed model incorporates deep learning and data mining to provide the accurate results with minimum errors.

This project has five modules. They are:

- 1.Datasets Acquisition
- 2.Proprocessing
- 3.Features Selection
- 4.Classification
- 5.Diseases diagnosis

# INTRODUCTION

## 1. INTRODUCTION

This project entitled “**Visualizing And Prediction Heart Diseases With An Interactive Dash Board**” has been developed under **C#.NET** as frontend and **SQL** backend.

Deep learning which is a hot buzz nowadays and has firmly put down its roots in a vast multitude of industries that are investing in fields like Artificial Intelligence, Big Data and Analytics. For example, Google is using deep learning in its voice and image recognition algorithms whereas Netflix and Amazon are using it to understand the behavior of their customer. In fact, you won't believe it, but researchers at MIT are trying to predict future using deep learning. Deep learning can be considered as a subset of machine learning. It is a field that is based on learning and improving on its own by examining computer algorithms. While machine learning uses simpler concepts, deep learning works with artificial neural networks, which are designed to imitate how humans think and learn. Neural Network is the biological neurons, which is nothing but a brain cell. Until recently, neural networks were limited by computing power and thus were limited in complexity.

Heart disease, also known as cardiovascular disease, refers to a group of conditions that affect the heart and blood vessels. It is the leading cause of death worldwide, accounting for nearly one-third of all deaths globally. There are various types of heart diseases, including coronary artery disease, heart failure, arrhythmias, and heart disease, among others. These conditions can be caused by various factors, including high blood pressure, high cholesterol, smoking, diabetes, obesity, and a family history of heart disease. Early detection and diagnosis of heart disease can significantly improve the chances of successful treatment and management. Therefore, the development of predictive models to identify individuals at high risk of developing heart disease is of utmost importance. In recent years, machine learning algorithms have been used to predict the risk of heart disease by analyzing various risk factors and clinical variables. These models have shown promising results in predicting the likelihood of developing heart disease, allowing for early intervention and improved outcomes. The objective of using the MLP (Multilayer Perceptron) algorithm for heart disease prediction is to develop an accurate predictive model that can identify individuals at high risk of disease.

The MLP algorithm is a type of neural network that can learn and make predictions based on patterns in data. The MLP algorithm can be trained on a dataset containing various risk factors and clinical variables, such as age, gender, blood pressure, cholesterol levels, smoking status, diabetes, and family history of heart disease. The algorithm can then use this information to predict the likelihood of an individual developing heart disease. The ultimate goal of using the MLP algorithm for heart disease prediction is to improve early detection and diagnosis of heart disease, which can lead to earlier intervention and better outcomes for patients. By accurately identifying individuals at high risk of developing heart disease, healthcare providers can develop targeted interventions, such as lifestyle changes or medication, to reduce the risk of heart disease and improve overall health outcomes. Data mining tasks that systematically incorporate domain knowledge, particularly formal semantics, into the process are referred to as semantic data mining. Numerous studies conducted in the past have confirmed the advantages of including domain knowledge in data mining.

The family of domain knowledge has also been enriched by the growth of knowledge engineering, particularly formal semantics and Semantic Web ontologies. Ontology is a formal method of defining the semantics of knowledge and data. It is an explicit specification of conceptualization. Ontology is a natural way to encode domain knowledge for use in data mining due to its formal structure. Information in this data can improve business operations and help them target the right clients. However, data must be properly interpreted in order to be useful. Inaccurate data interpretation can produce flawed conclusions that may interfere with an organization's growth and stability plans. Businesses and other organisations are employing data scientists to assist with the collection, storage, and analysis of pertinent data in order to ensure that the vast amounts of information are utilised effectively. Despite the fact that these professionals have a variety of backgrounds, the developing field of data science offers several lucrative opportunities for engineers in particular. Because the fields overlap so significantly, people with engineering backgrounds are frequently a good fit for jobs involving data. Building predictive, prescriptive, and prescriptive analytical models using a vast amount of data is the focus of the field or domain known as data science. It involves collecting, analysing, building the model, validating, and using the data deploying the best mod

# SYSTEM SPECIFICATION

## **2. SYSTEM SPECIFICATION**

### **2.1 HARDWARE REQUIREMENTS**

- Processor : Intel core processor 2.6.0 GHZ
- RAM : 4 GB
- Hard disk : 320 GB
- Compact Disk : 650 Mb
- Keyboard : Standard keyboard
- Monitor : 15 inch color monitor

### **2.2 SOFTWARE REQUIREMENTS**

- Operating System : Windows OS
- Front End : .NET (C#)
- IDE : VISUAL STUDIO
- Back End : SQL SERVER
- Application : WINDOWS APPLICATION

## **2.3 SOFTWARE REQUIREMENTS**

### **ABOUT THE FRONT ENT**

#### **2.3.1 FEATURES OF C#.NET**

The .NET Framework pronounced dot net is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large library and provides language interoperability each language can use code written in other languages across several programming languages. Programs written for the .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

The .NET Framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with the .NET Framework and other libraries. The .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio

### **Design Features**

#### **Interoperability**

Because computer systems commonly require interaction between newer and older applications, the .NET Framework provides means to access functionality implemented in newer and older programs that execute outside the .NET environment. Access to COM components is provided in the System Runtime Interop Services and System. Enterprise Services namespaces of the framework; access to other functionality is achieved using the P/Invoke feature.

## **Common Language Runtime engine**

The Common Language Runtime CLR serves as the execution engine of the .NET Framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviors in the areas of memory management, security, and exception handling.

## **Language independence**

The .NET Framework introduces a Common Type System, or CTS. The CTS specification defines all possible data types and programming constructs supported by the CLR and how they may or may not interact with each other conforming to the Common Language Infrastructure (CLI) specification. Because of this feature, the .NET Framework supports the exchange of types and object instances between libraries and applications written using any conforming .NET language.

## **Base Class Library**

The Base Class Library (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides classes that encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation, and so on. It consists of classes, interfaces of reusable types that integrate with CLR (Common Language Runtime).

## **Simplified deployment**

The .NET Framework includes design features and tools which help manage the installation of computer software to ensure it does not interfere with previously installed software, and it conforms to security requirements.

## **Security**

The design addresses some of the vulnerabilities, such as buffer overflows, which have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

## Portability

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be platform-agnostic,<sup>[3]</sup> and cross-platform implementations are available for other operating systems (see Silverlight and the Alternative implementations section below). Microsoft submitted the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

## Common Language Infrastructure (CLI)

The purpose of the Common Language Infrastructure (CLI) is to provide a language-neutral platform for application development and execution, including functions for Exception handling, Garbage Collection, security, and interoperability. By implementing the core aspects of the .NET Framework within the scope of the CL, this functionality will not be tied to a single language but will be available across the many languages supported by the framework. Microsoft's implementation of the CLI is called the Common Language Runtime, or CLR.

The CIL code is housed in CLI assemblies. As mandated by the specification, assemblies are stored in the Portable Executable PE format, common on the Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly not to be confused with the filename on disk contains its simple text name, version number, culture, and public key token. Assemblies are considered equivalent if they share the same complete name, excluding the revision of the version number. A private key can also be used by the creator of the assembly for strong naming. The public key token identifies which public key an assembly is signed with. Only the creator of the keypair typically the .NET developer signing the assembly can sign assemblies that have the same strong name as a previous version assembly, since he is in possession of the private key. Strong naming is required to add assemblies to the Global Assembly Cache

## Portability

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be platform-agnostic,<sup>[3]</sup> and cross-platform implementations are available for other operating systems (see Silverlight and the Alternative implementations section below). Microsoft submitted the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

## Common Language Infrastructure (CLI)

The purpose of the Common Language Infrastructure (CLI) is to provide a language-neutral platform for application development and execution, including functions for Exception handling, Garbage Collection, security, and interoperability. By implementing the core aspects of the .NET Framework within the scope of the CL, this functionality will not be tied to a single language but will be available across the many languages supported by the framework. Microsoft's implementation of the CLI is called the Common Language Runtime, or CLR.

The CIL code is housed in CLI assemblies. As mandated by the specification, assemblies are stored in the Portable Executable PE format, common on the Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly not to be confused with the filename on disk contains its simple text name, version number, culture, and public key token. Assemblies are considered equivalent if they share the same complete name, excluding the revision of the version number. A private key can also be used by the creator of the assembly for strong naming. The public key token identifies which public key an assembly is signed with. Only the creator of the keypair typically the .NET developer signing the assembly can sign assemblies that have the same strong name as a previous version assembly, since he is in possession of the private key. Strong naming is required to add assemblies to the Global Assembly Cache

## Security

.NET has its own security mechanism with 2 general features: Code Access Security (CAS), and validation and verification. Code Access Security is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly whether it is installed on the local machine or has been downloaded from the intranet or Internet. Code Access Security uses evidence to determine the permissions granted to the code. Other code can demand that calling code is granted a specified permission. The demand causes the CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

## Class library

The .NET Framework includes a set of standard class libraries. The class library is organized in a hierarchy of namespaces. Most of the built-in APIs are part of either System or Microsoft.\* namespaces. These class libraries implement a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. The .NET class libraries are available to all CLI compliant languages. The .NET Framework class library is divided into two parts: the Base Class Library and the Framework Class Library

The Base Class Library (BCL) includes a small subset of the entire class library and is the core set of classes that serve as the basic API of the Common Language Runtime.<sup>[9]</sup> The classes in mscorlib.dll and some of the classes in System.dll and System.core.dll are considered to be a part of the BCL. The BCL classes are available in both .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silverlight and Mono.

The Framework Class Library (FCL) is a superset of the BCL classes and refers to the entire class library that ships with .NET Framework. It includes an expanded set of libraries, including Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation among others. The FCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to the standard libraries of Java.

## Memory management

The .NET Framework CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Memory is allocated instantiations of .NET types (objects) from the managed heap, a pool of memory managed by the CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection. .NET Framework includes a garbage collector which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them.

The .NET Garbage Collector (GC) is a non-deterministic, compacting, mark-and-sweep garbage collector. The GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, the GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (managed objects). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers.<sup>[10]</sup> When the GC runs, it pauses the application, and for each object referred to in the root, it recursively enumerates all the objects reachable from the root objects and marks them as reachable. It uses CLI metadata and reflection to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage. This is the mark phase. Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then compacted together to make used memory contiguous again. Any reference to an object invalidated by moving the object is updated by the GC to reflect the new location. The application is resumed after the garbage collection is over.

## ABOUT THE BACKEND

### 2.3.2 FEATURES OF MYSQL

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications which may run either on the same computer or on another computer across a network including the Internet. Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Data storage is a database, which is a collection of tables with typed columns. SQL Server supports different data types, including primary types such as Integer, Float, Decimal, Char (including character strings, Varchar (variable length character strings, binary (for unstructured blobs of data), Text (for textual data) among others. The rounding of floats to integers uses either Symmetric Arithmetic Rounding or Symmetric Round Down fix depending on arguments: `SELECT Round (2.5, 0)` gives 3.

Microsoft SQL Server also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log. A SQL Server database can contain a maximum of  $2^{31}$  objects, and can span multiple OS-level files with a maximum file size of  $2^{60}$  bytes (1 exabyte). The data in the database are stored in primary data files with an extension `mdf`. Secondary data files, identified with a `ndf` extension, are used to allow the data of a single database to be spread across more than one file, and optionally across more than one file system. Log files are identified with the `ldf` extension.

Storage space allocated to a database is divided into sequentially numbered pages, each 8 KB in size. A page is the basic unit of I/O for SQL Server operations. A page is marked with a 96-byte header which stores metadata about the page including the page number, page type, free space on the page and the ID of the object that owns it. Page type defines the data contained in the page: data stored in the database, index, allocation map which holds information about how pages are allocated to tables and indexes, change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While page is the basic unit of an I/O operation, space is actually managed in

which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While page is the basic unit of an I/O operation, space is actually managed in terms of an extent which consists of 8 pages. A database object can either span all 8 pages in an extent ("uniform extent") or share an extent with up to 7 more objects ("mixed extent"). A row in a database table cannot span more than one page, so is limited to 8 KB in size. However, if the data exceeds 8 KB and the row contains varchar or varbinary data, the data in those columns are moved to a new page or possibly a sequence of pages, called an allocation unit and replaced with a pointer to the data.

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

### **Concurrency and locking:**

SQL Server allows multiple clients to use the same database concurrently. As such, it needs to control concurrent access to shared data, to ensure data integrity—when multiple clients update the same data, or clients attempt to read data that is in the process of being changed by another client. SQL Server provides two modes of concurrency control: pessimistic concurrency and optimistic concurrency. When pessimistic concurrency control is being used, SQL Server controls concurrent access by using locks. Locks can be either shared or exclusive. Exclusive lock grants the user exclusive access to the data no other user can access the data as long as the lock is held. Shared locks are used when some

data is being read multiple users can read from data locked with a shared lock, but not acquire an exclusive lock. The latter would have to wait for all shared locks to be released. Locks can be applied on different levels of granularity on entire tables, pages, or even on a per-row basis on tables. For indexes, it can either be on the entire index or on index leaves. The level of granularity to be used is defined on a per-database basis by the database administrator. While a fine-grained locking system allows more users to use the table or index simultaneously, it requires more resources, so it does not automatically yield higher performance. SQL Server also includes two more lightweight mutual exclusion solutions latches and spinlocks which are less robust than locks but are less resource intensive. SQL Server uses them for DMVs and other resources that are usually not busy. SQL Server also monitors all worker threads that acquire locks to ensure that they do not end up in deadlocks in case they do, SQL Server takes remedial measures, which in many cases are to kill one of the threads entangled in a deadlock and roll back the transaction it started.<sup>[6]</sup> To implement locking, SQL Server contains the Lock Manager. The Lock Manager maintains an in-memory table that manages the database objects and locks, if any, on them along with other metadata about the lock. Access to any shared object is mediated by the lock manager, which either grants access to the resource or blocks it. SQL Server also provides the optimistic concurrency control mechanism, which is similar to the multiversion concurrency control used in other databases. Social database frameworks are the most critical database frameworks utilized as a part of the product business today. A standout amongst the most remarkable frameworks is Microsoft SQL Server. SQL Server is a database administration framework created and showcased by Microsoft. It runs solely under Windows NT and Windows 95/98.

**The most critical parts of SQL Server 8 are:**

- SQL Server is anything but difficult to utilize.
- SQL Server scales from a portable tablet to symmetric multiprocessor frameworks.
- SQL Server gives information warehousing elements that as of recently have just been accessible in Oracle and other more costly DBMSs.

A database framework is a general gathering of distinctive database programming segments and databases containing the parts viz. Database application projects, Front-End segments, Database administration frameworks, and Databases.

➤ **A database framework must give the accompanying elements:**

- A mixture of client interfaces
- Physical information autonomy
- Logical information autonomy
- Query advancement
- Data honesty
- Concurrency control
- Backup and recuperation
- Security and approval

SQL Server is a Relational Database Management System. The SQL Server social dialect is called Transact-SQL. SQL is resource arranged dialect. This implies that SQL can inquiry numerous lines from one or more tables utilizing only one announcement. This component permits the utilization of this dialect at a coherently larger amount than procedural dialects. Another vital property of SQL is its non-procedurally. SQL contains two sub dialects DDL and DML.

SQL Server functions as a characteristic augmentation of Windows NT and windows 95/98. SQL Server is generally simple to oversee through the utilization of a graphical registering environment for each undertaking of framework and database organization. SQL Server uses administrations of Windows NT to offer new or expanded database capacities, for example, sending and accepting messages and overseeing login security.

The SQL Server chairman's essential device for connecting with the framework is Enterprise Manager. The Enterprise Manager has two primary purposes: Administration of the database server and Management of database items.

- SQL Server Query Analyzer gives a graphical presentation of the execution arrangement of a question and a programmed segment that recommends which list

ought to be utilized for a chose inquiry. This intelligent segment of SQL Server performs the assignments like:

- Generating and executing Transact-SQL explanations
- Putting away the produced Transact-SQL explanations in a document
- Analyzing execution gets ready for produced inquiries
- Graphically representing the execution arrangement for a chose question.

A put away method is an exceptional sort of clump written in Transact-SQL utilizing the SQL dialect and SQL augmentations. It is saved money on the database server to enhance the execution and consistency of monotonous undertakings. SQL Server backings put away methods and framework techniques. Put away techniques can be utilized for the accompanying purposes: to control access approval, to make a review trial of exercises in database tables, to discrete information definition & information control articulations concerning a database & every single comparing application.

The database article perspective can be utilized for:

- Restricting the utilization of specific sections and lines of tables - that is to control access to a specific piece of one or more tables,
- To shroud the points of interest of confounded inquiries, to limit embedded & redesigned qualities to certain extents.

The Query Optimizer is the piece of SQL Server that chooses how to best perform a question. It creates a few inquiry execution gets ready for the given question & chooses the arrangement with the most minimal expense.

# SYSTEM ANALYSIS

### 3. SYSTEM ANALYSIS

#### 3.1 EXISTING SYSTEM

Present days one of the major application areas of machine learning algorithms is medical diagnosis of diseases and treatment. Machine learning algorithms also used to find correlations and associations between different diseases. Nowadays many people are dying because of sudden heart attack. Prediction and diagnosing of heart disease becomes a challenging factor faced by doctors and hospitals both in India and abroad. In order to reduce number of deaths because of heart diseases, we have to predict whether person is at the risk of heart disease or not in advance. Data mining techniques and machine learning algorithms play a very important role in this area. Many researchers are carrying out their research in this area to develop software that can help doctors to take decision regarding both prediction and diagnosing of heart disease. In this paper we focused on how data mining techniques can be used to predict heart disease in advance such that patient is well treated. An important task of any diagnostic system is the process of attempting to determine and/or identify a possible disease or disorder and the decision reached by this process. For this purpose, machine learning algorithms are widely employed. For these machine learning techniques to be useful in medical diagnostic problems, they must be characterized by high performance, the ability to deal with missing data and with noisy data, the transparency of diagnostic knowledge, and the ability to explain decisions. As people are generating more data everyday so there is a need for such a classifier which can classify those newly generated data accurately and efficiently. This System mainly focuses on the supervised learning technique called the Random forests for classification of data by changing the values of different hyper parameters in Random Forests Classifier

#### DISADVANTAGE

- ⌘ Labeled data based disease classification
- ⌘ Provide high number of false positive
- ⌘ Binary classification can be occurred
- ⌘ Computational complexity

### 3.2 PROPOSED SYSTEM

Cardiovascular disease continues to claim an alarming number of lives across the globe. CVD disease is the greatest scourge affecting the industrialized nations. CVD not only strikes down a significant fraction of the population without warning but also causes prolonged suffering and disability in an even larger number. Although large proportion of CVDs is preventable they continue to rise mainly because preventive measures are inadequate. Heart disease diagnosis has become a difficult task in the field of medicine. This diagnosis depends on a thorough and accurate study of the patient's clinical tests data on the health history of an individual. The tremendous improvement in the field of machine learning aim at developing intelligent automated systems which helps the medical practitioners in predicting as well as making decisions about the disease. Such an automated system for medical diagnosis would enhance timely medical care followed by proper subsequent treatment thereby resulting in significant lifesaving. Incorporating the techniques of classification in these intelligent systems achieve at accurate diagnosis. Neural Networks has emerged as an important method of classification. Multi-layer Perceptron Neural Network with Back-propagation has been employed as the training algorithm in this work. This project proposes a diagnostic system for predicting heart disease with improved accuracy. The propagation algorithm has been repeated until minimum error rate was observed. And it is quite evident from the results presented in the previous section that the accuracy rate is maximized.

### ADVANTAGES

- ∞ Accuracy is high
- ∞ Parallel processing
- ∞ Multiple heart diseases are predicted
- ∞ Reduce number of false positive rate

# SYSTEM DESIGN

## **4.SYSTEM DESIGN**

### **4.1MODULE DESCRIPTION**

#### **MODULE**

- ∞ Datasets Acquisition
- ∞ Preprocessing
- ∞ Features Selection
- ∞ Classification
- ∞ Disease diagnosis

#### **DATASETS ACQUISITION**

A data set is a collection of data. Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. The term data set may also the data in a collection of closely related tables, corresponding to a particular experiment or event. This module, is used to upload the cardiovascular datasets related to heart diseases which includes the attributes such as age, gender, height, weight, systolic blood pressure, diastolic blood pressure, cholesterol, glucose, smoke, alcohol, active status, cardio labels.

#### **PREPROCESSING**

Data preprocessing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis. If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. It eliminate the irrelevant values and also estimate the missing values of data. In order to provide structured datasets.

## FEATURES SELECTION

Feature selection refers to the process of reducing the inputs for processing and analysis, or of finding the most meaningful inputs. Feature engineering (or feature extraction), refers to the process of extracting useful information or features from existing data. Feature selection methods apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often uni-variate and consider the feature independently, or with regard to the dependent variable. It can be used to construct the multiple heart diseases. In this module, multiple features are selected from uploaded datasets. The datasets are trained with various disease labels such as Coronary heart diseases, Cardiac arrest, High blood pressure, Arrhythmia and normal.

## CLASSIFICATION

This module implements classification algorithm to predict the heart diseases. and deep learning algorithm such as Multi-layer perceptron algorithm to predict the diseases. A multilayer perceptron MLP is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. It MLP consists of multiple layers of nodes in a directed graph, and each layer is fully connected to the next one. Each node is a neuron with a nonlinear activation function except for the input nodes. MLP utilizes a supervised learning technique called back propagation for training the network. MLP is a modified form of the standard linear perceptron and can distinguish data that are not linearly separable.

If a multilayer perceptron MLP has a simple on-off mechanism i.e. linear activation function in all neurons ,to determine whether or not a neuron fires, then it is easily proved with linear algebra that any number of layers can be reduced to the standard two-layer input-output model. The gradient techniques are then applied to the optimization methods to adjust the weights to minimize the loss function in the network. Hence, the algorithm requires a known and a desired output for all inputs in order to compute the gradient of loss function. Usually, the generalization of MultiLayerd Feed Forward Networks is done using delta rule which possibly makes a chain of iterative rules to compute gradients for each layer. Back Propagation Algorithm necessitates the activation function to be different between the neurons. The ongoing researches on parallel, distributed computing and computational

neuroscience are currently implemented with the concepts of MultiLayerd Perceptron using a Back Propagation Algorithm. MLP Back Propagation Algorithm has also gained focus in pattern recognition domain. They are so convenient in research, because of their ability in solving complex problems, and also for their fitness approximation results even with critical predictions. MLP is one of the Neural Network models, has the same architecture of Feed-Forward back Propagation for Supervised training. The multilayer perceptron is the most known and most frequently used type of neural network. User can provide the features and automatically predict the diseases.

## **BPNN ALGORITHM**

Back Propagation Neural Network (BPNN) is a sort of artificial neural network that is frequently used in pattern recognition, data categorization, and prediction applications. The approach is built on the supervised learning principle, which implies that the network is trained using labelled data, with each input connected with a known output. The BPNN algorithm is made up of many layers: an input layer, one or more hidden layers, and an output layer. Each layer is made up of a collection of nodes, or neurons, that accept input from the previous layer and generate output that is passed on to the next layer. The BPNN method changes the weights between the neurons in the network during training to reduce error. The BPNN method changes the weights between the neurons in the network during training to reduce the error between the actual and predicted output. This is accomplished by employing a mathematical process known as the back propagation algorithm to propagate the mistake back through the network, from the output layer to the input layer, and modifying the weights accordingly. The error gradient for each weight in the network is calculated by the back propagation method and used to update the weights in the opposite direction as the gradient. This procedure is performed for each input in the training data until the network achieves an acceptable degree of accuracy. After training, the BPNN technique may be used to generate predictions on fresh, previously unseen data by passing the inputs through the network and creating an output that reflects the network's forecast. The BPNN method offers various advantages, including the capacity to learn complicated patterns in data, high prediction accuracy, and the ability to deal with noisy or missing data. It does, however, have certain drawbacks, such as the risk of overfitting if the network is too complicated if the training data is not representative of the genuine population

## **MULTI LAYER PERCEPTRON**

A Multi-Layer Perceptron (MLP) is a type of artificial neural network that is commonly used in machine learning. It is composed of multiple layers of interconnected nodes or neurons, where each neuron is a mathematical function that receives input signals, performs a computation on them, and produces an output. The basic architecture of an MLP includes an input layer, one or more hidden layers, and an output layer. The input layer receives the input data, and each subsequent hidden layer applies a non-linear transformation to the output of the previous layer. The output layer produces the final output of the network. During training, an MLP adjusts the weights and biases of the neurons in the network to minimize a cost or loss function, which measures the difference between the predicted output and the actual output. This process, called backpropagation, uses the chain rule of calculus to compute the gradient of the loss function with respect to the weights and biases. Once the MLP is trained, it can be used to make predictions on new data by feeding the data through the input layer and propagating it through the network to produce an output. MLPs are widely used in applications such as image and speech recognition, natural language processing, and predictive modeling.


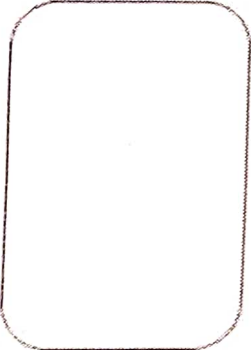


## **DIAGNOSIS**

Medical decision support system is a decision-support program which is designed to assist physicians and other health professionals with decision making tasks, such as determining diagnosis of patients' data. In this module, provide the diagnosis information based on predicted heart diseases. Proposed system provides improved accuracy in heart disease prediction. Risk factors are conditions or habits that make a person more likely to develop a disease.

## 4.2 DATA FLOW DIAGRAM

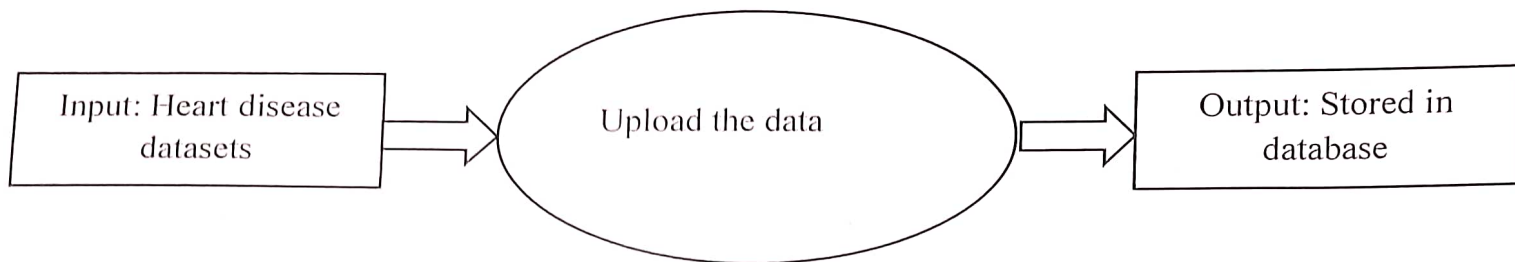
A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

### Data flow Symbols:

Symbol	Description
	An <b>entity</b> . A source of data or a destination for data.
	A <b>process</b> or task that is performed by the system.
	A <b>data store</b> , a place where data is held between processes.
	A <b>data flow</b> .

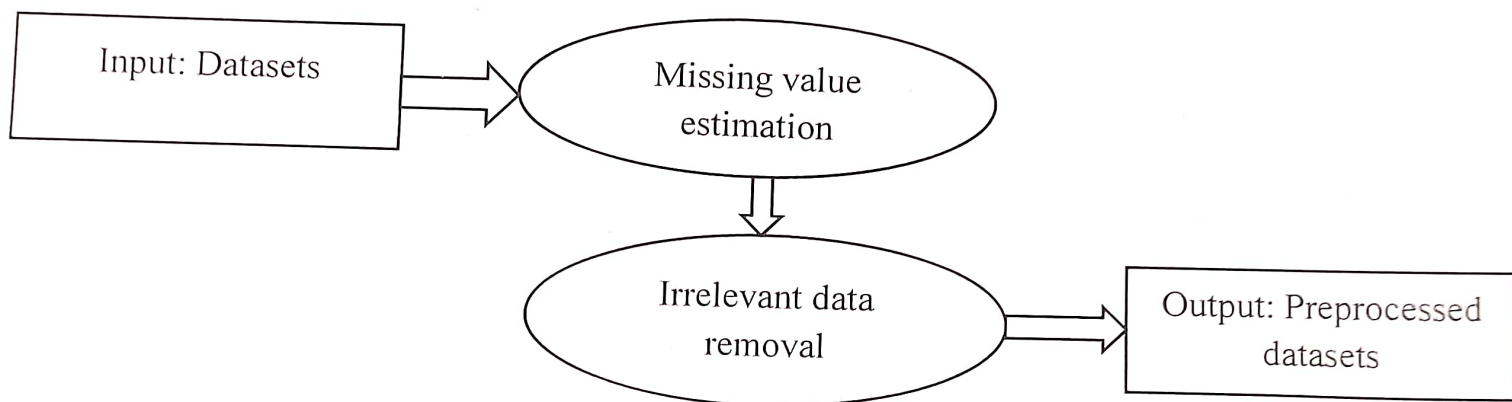
## LEVEL 0

The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



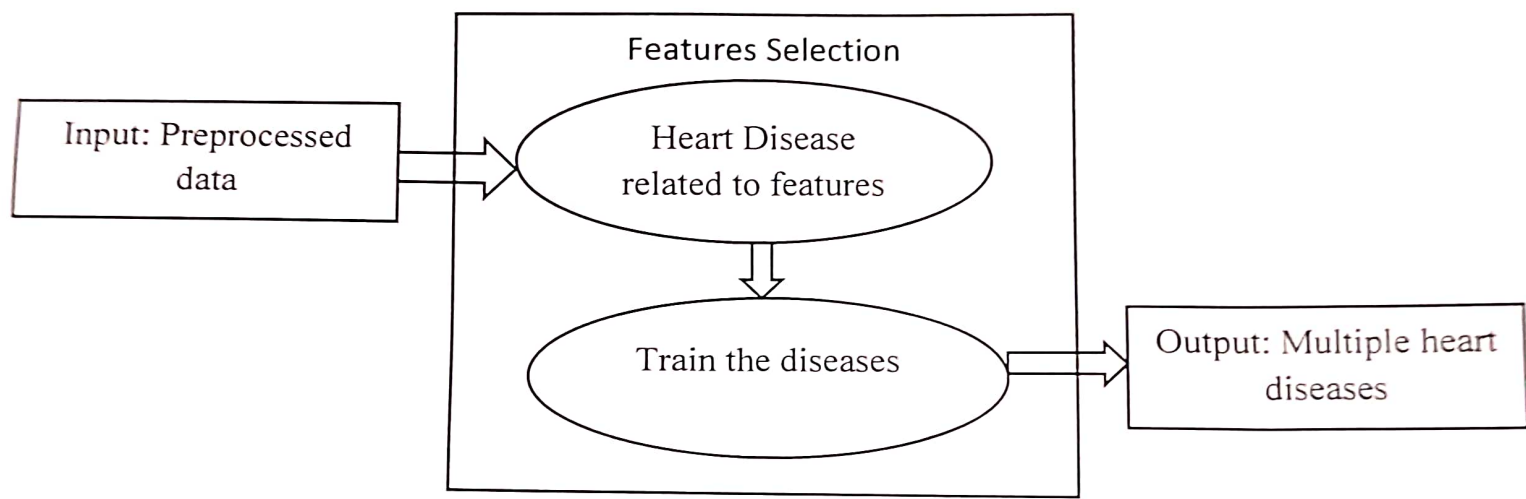
## LEVEL-1

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.



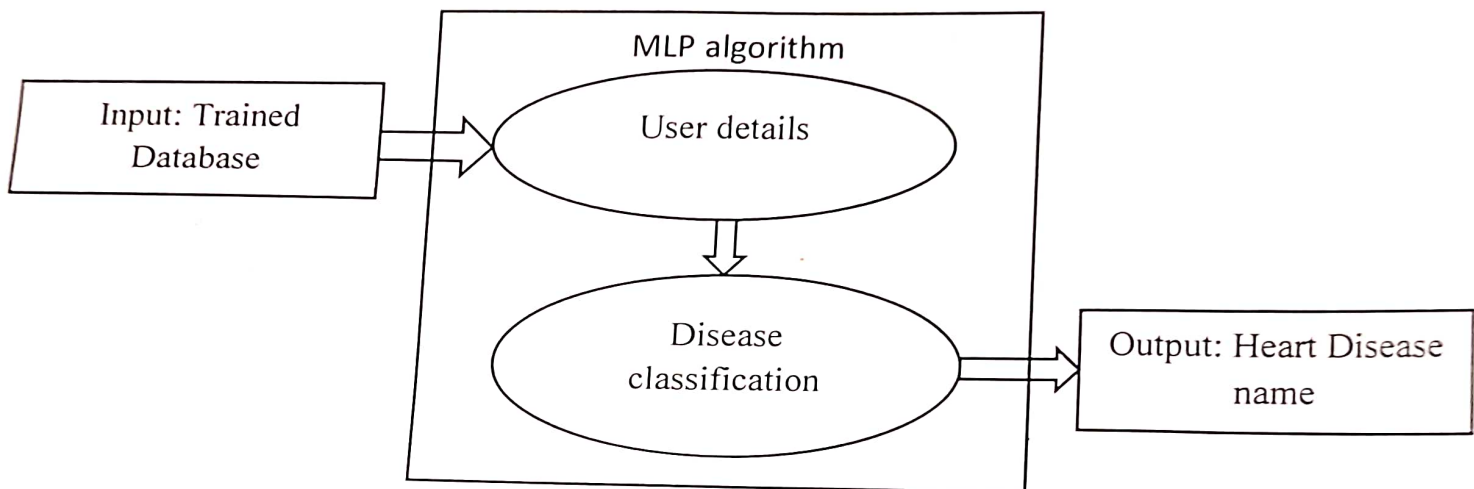
**LEVEL-2**

A Data Flow Diagram (DFD) tracks processes and their data paths within the business or system boundary under investigation. A DFD defines each domain boundary and illustrates the logical movement and transformation of data within the defined boundary. The diagram shows 'what' input data enters the domain, 'what' logical processes the domain applies to that data, and 'what' output data leaves the domain. Essentially, a DFD is a tool for process modeling and one of the oldest.



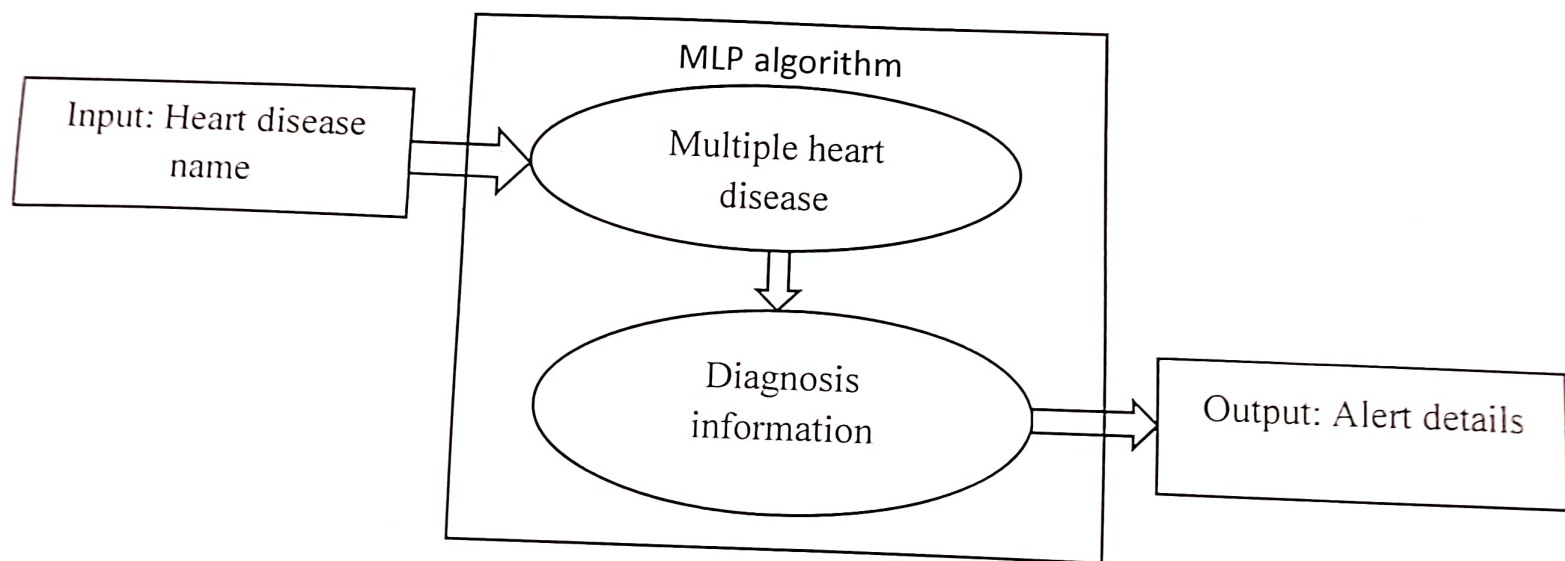
### LEVEL-3

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system. A DFD shows the flow of data from data sources and data stores to processes, and from processes to data stores and data sinks. DFDs are used for modelling and analyzing the flow of data in data processing systems, and are usually accompanied by a data dictionary, an entity-relationship model, and a number of process descriptions.



#### LEVEL 4

A DFD may look similar to a flow chart. However, there is a significant difference with the data flow diagram. The arrows in DFDs show that there is a flow of data between the two components and not that the component is sending the data that must be executed in the following component. A component in DFD may not continue execution when sending data and during execution of the component receiving the data. The component sending data can send multiple sets of data along several connections. In fact, a DFD node can be a component that never ends.



#### 4.3TABLE DESIGN

A table is a data structure that organizes information into rows and columns. It can be used to both store and display data in a structured format. For example, databases store data in tables so that information can be quickly accessed from specific rows. Websites often use tables to display multiple rows of data on page. Spreadsheets combine both purposes of a table by storing and displaying data in a structured format.

Databases often contain multiple tables, with each one designed for a specific purpose. For example, a company database may contain separate tables for employees, clients, and suppliers. Each table may include its own set of fields, based on what data the table needs to store. In database tables, each field is considered a column, while each entry (or record), is considered a row. A specific value can be accessed from the table by requesting data from an individual column and row.

**TABLE NAME:** regtb

Name	nvarchar(50)	Checked
Gender	nvarchar(50)	Checked
Mobile	nvarchar(50)	Checked
Email	nvarchar(50)	Checked
UserName	nvarchar(50)	Checked
Password	nvarchar(50)	Checked

**TABLE NAME** resulttb

Id	bigint	Unchecked
Pid	nvarchar(50)	Checked
Age	nvarchar(50)	Checked
Gender	nvarchar(50)	Checked
Height	nvarchar(50)	Checked
Weight	nvarchar(50)	Checked
Ap_hi	nvarchar(50)	Checked
Ap_lo	nvarchar(50)	Checked
Cholesterol	nvarchar(50)	Checked
Gluc	nvarchar(50)	Checked
smoke	nvarchar(50)	Checked
Alco	nvarchar(50)	Checked

Active	nvarchar(50)	Checked
Result	nvarchar(50)	Checked

**TABLE NAME** rulestb

Id	bigint	Unchecked
StartRange	int	Checked
EndRange	int	Checked
Attribute	nvarchar(50)	Checked
HeartDisease	nvarchar(50)	Checked

**TABLE NAME** temptb

Id	bigint	Unchecked
Pid	nvarchar(50)	Checked
Age	nvarchar(50)	Checked
Gender	nvarchar(50)	Checked
Height	nvarchar(50)	Checked
Weight	nvarchar(50)	Checked
Ap_hi	nvarchar(50)	Checked
Ap_lo	nvarchar(50)	Checked
Cholesterol	nvarchar(50)	Checked
Gluc	nvarchar(50)	Checked
smoke	nvarchar(50)	Checked
Alco	nvarchar(50)	Checked
Active	nvarchar(50)	Checked

# SYSTEM TESTING

## **5.SYSTEM TESTING**

Testing is a series of different tests that whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all work should verify that all system element have been properly integrated and performed allocated function. Testing is the process of checking whether the developed system works according to the actual requirement and objectives of the system. The philosophy behind testing is to find the errors. A good test is one that has a high probability of finding an undiscovered error. A successful test is one that uncovers the undiscovered error. Test cases are devised with this purpose in mind. A test case is a set of data that the system will process as an input.

### **Types of Testing**

#### **System testing**

After a system has been verified, it needs to be thoroughly tested to ensure that every component of the system is performing in accordance with the specific requirements and that it is operating as it should including when the wrong functions are requested or the wrong data is introduced.

Testing measures consist of developing a set of test criteria either for the entire system or for specific hardware, software and communications components. For an important and sensitive system such as an electronic voting system, a structured system testing program may be established to ensure that all aspects of the system are thoroughly tested.

Testing measures that could be followed include:

- Applying functional tests to determine whether the test criteria have been met
- Applying qualitative assessments to determine whether the test criteria have been met.
- Conducting tests in “laboratory” conditions and conducting tests in a variety of “real life” conditions.
- Conducting tests over an extended period of time to ensure systems can perform consistently.

- Conducting “load tests”, simulating as close as possible likely conditions while using or exceeding the amounts of data that can be expected to be handled in an actual situation.

Test measures for hardware may include:

- Applying “non-operating” tests to ensure that equipment can stand up to expected levels of physical handling.
- Testing “hard wired” code in hardware (firmware) to ensure its logical correctness and that appropriate standards are followed.

Tests for software components also include:

- Testing all programs to ensure its logical correctness and that appropriate design, development and implementation standards have been followed.
- Conducting “load tests”, simulating as close as possible a variety of “real life” conditions using or exceeding the amounts of data that could be expected in an actual situation.
- Verifying that integrity of data is maintained throughout its required manipulation.

## **Unit Testing**

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results. Functional and reliability testing in an Engineering environment. Producing tests for the behavior of components (nodes and vertices) of a product to ensure their correct behavior prior to system integration

## **Integration Testing**

Testing in which modules are combined and tested as a group. Modules are typically code modules, individual applications, source and destination applications on a network, etc. Integration Testing follows unit testing and precedes system testing. Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is release.

# SYSTEM IMPLEMENTATION

## 6.SYSTEM IMPLEMENTATION

### 6.1SOURCE CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace Heart_Disease
```

```
{
```

```
public partial class Form1 : Form
```

```
{
```

```
public Form1()
```

```
{
```

```
InitializeComponent();
```

```
}
```

```
private void button1_Click(object sender, EventArgs e)
```

```
{
```

```
}
```

```
private void button1_Click_1(object sender, EventArgs e)
```

```
{
```

```
UploadDataset up = new UploadDataset();
```

```
up.Show();
```

```
}
```

```
private void button2_Click(object sender, EventArgs e)
```

```

    {
        UserLoginuu = new UserLogin();
        uu.Show();

    }
}

```

## UPLOAD

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Data.SqlClient;
namespace Heart_Disease
{
    public partial class UploadDataset : Form
    {
        SqlConnection con1 = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\NewProject2019\PGPROJECT\JJHeart_
Disease\Heart_Disease\Heartdb.mdf;Integrated Security=True;User Instance=True");
        SqlCommand cmd1;
        string filepath;
        public UploadDataset()
        {
            InitializeComponent();
        }
    }
}

```

```

private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog op = new OpenFileDialog();
    op.ShowDialog();

    textBox1.Text = op.FileName;

    filepath = op.FileName;

    button2.Enabled = true;
}

private void button2_Click(object sender, EventArgs e)
{
    StreamReader sr = new StreamReader(filepath);
    string line = sr.ReadLine();
    string[] value = line.Split(',');
    DataTable dt = new DataTable();
    DataRow row = null;
    foreach (string dc in value)
    {
        dt.Columns.Add(new DataColumn(dc));
    }

    while (!sr.EndOfStream)
    {
        value = sr.ReadLine().Split(',');
        if (value.Length == dt.Columns.Count)
        {
            row = dt.NewRow();
            row.ItemArray = value;
            dt.Rows.Add(row);
        }
    }
}

```

```
    }  
    }  
    dataGridView1.DataSource = dt;
```

```
        button3.Enabled = true;  
    }
```

```
private void button3_Click(object sender, EventArgs e)
```

```
{  
    con1.Open();  
        cmd1 = new SqlCommand("truncate table temptb", con1);  
    cmd1.ExecuteNonQuery();  
    con1.Close();
```

```
    for (int rows = 0; rows < dataGridView1.Rows.Count - 1; rows++)
```

```
    {  
        stringconstring = @"Data  
Source=.\SQLEXPRESS;AttachDbFilename=D:\NewProject2019\PGPROJECT\JJ\Heart_  
Disease\Heart_Disease\Heartdb.mdf;Integrated Security=True;User Instance=True";  
        using (SqlConnection con = new SqlConnection(constring))
```

```
        {  
            using (SqlCommandcmd = new SqlCommand("INSERT INTO temptb  
VALUES(@Pid,@Age,@Gender,@Height,@Weight,@Ap_hi,@Ap_lo,@Cholesterol,@G  
luc,@smoke,@Alco,@Active)", con))
```

```
            {  
                cmd.Parameters.AddWithValue("@Pid", dataGridView1.Rows[rows].Cells[0].Value);  
                cmd.Parameters.AddWithValue("@Age", dataGridView1.Rows[rows].Cells[1].Value);  
                cmd.Parameters.AddWithValue("@Gender", dataGridView1.Rows[rows].Cells[2].Value);  
                cmd.Parameters.AddWithValue("@Height", dataGridView1.Rows[rows].Cells[3].Value);  
                cmd.Parameters.AddWithValue("@Weight", dataGridView1.Rows[rows].Cells[4].Value);  
                cmd.Parameters.AddWithValue("@Ap_hi", dataGridView1.Rows[rows].Cells[5].Value);
```

```

cmd.Parameters.AddWithValue("@Ap_lo", dataGridView1.Rows[rows].Cells[6].Value);

cmd.Parameters.AddWithValue("@Cholesterol",
dataGridView1.Rows[rows].Cells[7].Value);
cmd.Parameters.AddWithValue("@Gluc", dataGridView1.Rows[rows].Cells[8].Value);
cmd.Parameters.AddWithValue("@smoke", dataGridView1.Rows[rows].Cells[9].Value);
cmd.Parameters.AddWithValue("@Alco", dataGridView1.Rows[rows].Cells[10].Value);
cmd.Parameters.AddWithValue("@Active", dataGridView1.Rows[rows].Cells[11].Value);

```

```

con.Open();
cmd.ExecuteNonQuery();
con.Close();
    }
}
}

```

```

DataTable newTable = (DataTable) dataGridView1.DataSource;

```

```

Preprocessing p = new Preprocessing();
p.dt = newTable;
p.Show();
}

```

```

private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs
e)
{

}
}
}
}

```

## PREPROCESSING

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Heart_Disease
{
    public partial class Preprocessing : Form
    {
        SqlConnection con = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\NewProject2019\PGPROJECT\JJHeart_
Disease\Heart_Disease\Heartdb.mdf;Integrated Security=True;User Instance=True");
        SqlCommand cmd;

        public DataSet ds;
        public DataTable dt;

        public Preprocessing()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            cmd = new SqlCommand("delete from temptb where Age=" or Gender=" or Height=" or
Weight="or Ap_hi=" or Ap_lo=" or Cholesterol=" or Gluc=" or smoke=" or Alco=" or
Active=" ", con);
            con.Open();

```

```

int numberOfRecords = cmd.ExecuteNonQuery();

con.Close();

    label2.Text = numberOfRecords + "Remove(s)";
cmd = new SqlCommand("select * from temptb", con);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);

    dataGridView2.DataSource = dt;
dataGridView2.Refresh();
    button2.Enabled = true;
}

private void Preprocessing_Load(object sender, EventArgs e)
{
    dataGridView1.DataSource = dt;
dataGridView1.Refresh();
}

private void button2_Click(object sender, EventArgs e)
{
    MLP f2 = new MLP();
f2.Show();

}
}
}

```

## MLP

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;

```

```

using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

```

```

namespace Heart_Disease

```

```

{
    public partial class MLP : Form
    {

```

```

        SqlConnection con = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\NewProject2019\PGPROJECT\JJHeart_
Disease\Heart_Disease\Heartdb.mdf;Integrated Security=True;User Instance=True");
        SqlCommand cmd;

```

```

        public MLP()
        {
            InitializeComponent();
        }

```

```

        int aa;

```

```

        private void button1_Click(object sender, EventArgs e)
        {

```

```

            if (textBox2.Text == "" || textBox1.Text == "")
            {

```

```

            else

```

```

            {

```

```

                aa = Convert.ToInt32(textBox2.Text);

```

```

cmd = new SqlCommand("insert into rulestb values('" + textBox1.Text + "','" +
textBox2.Text + "','" + comboBox1.Text + "','" + comboBox2.Text + "')", con);
con.Open();
cmd.ExecuteNonQuery();
con.Close();
MessageBox.Show("Rules Construction");

```

```

cmd = new SqlCommand("select * from rulestb", con);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);

dataGridView1.DataSource = dt;
dataGridView1.Refresh();

```

```

}

```

```

}

```

```

private void button2_Click(object sender, EventArgs e)

```

```

{

```

```

con.Open();

```

```

cmd = new SqlCommand("Delete from rulestb where id='" + dataGridView1[2,
dataGridView1.CurrentRow.Index].Value.ToString() + "' ", con);

```

```

cmd.ExecuteNonQuery();

```

```

con.Close();

```

```

MessageBox.Show("Record Deleted!");

```

```

cmd = new SqlCommand("select * from rulestb", con);

```

```

SqlDataAdapter da = new SqlDataAdapter(cmd);

```

```
DataTable dt = new DataTable();  
da.Fill(dt);  
    dataGridView1.DataSource = dt;  
dataGridView1.Refresh();  
  
}
```

```
private void MLP_Load(object sender, EventArgs e)  
{  
    cmd = new SqlCommand("select * from rulestb", con);  
    SqlDataAdapter da = new SqlDataAdapter(cmd);  
    DataTable dt = new DataTable();  
    da.Fill(dt);  
    dataGridView1.DataSource = dt;  
    dataGridView1.Refresh();  
}
```

```
private void button3_Click(object sender, EventArgs e)  
{  
    Result rr = new Result();  
    rr.Show();  
  
}
```

```
private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs  
e)  
{  
  
}  
}  
}
```

## RESULT

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient ;
```

```
namespace Heart_Disease
```

```
{
```

```
public partial class Result : Form
```

```
{
```

```
SqlConnection con = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\NewProject2019\PGPROJECT\JJ\Heart_
Disease\Heart_Disease\Heartdb.mdf;Integrated Security=True;User Instance=True");
SqlCommand cmd;
```

```
SqlConnection con1 = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\NewProject2019\PGPROJECT\JJ\Heart_
Disease\Heart_Disease\Heartdb.mdf;Integrated Security=True;User Instance=True");
SqlCommand cmd1;
```

```
public Result()
```

```
{
```

```
InitializeComponent();
```

```
}
```

decimal sAp\_hi1, tAp\_hi1, sAp\_Lo1, tAp\_Lo1, sCholesterol1, tCholesterol1, sGluc1, tGluc1;

decimal sAp\_hi2, tAp\_hi2, sAp\_Lo2, tAp\_Lo2, sCholesterol2, tCholesterol2, sGluc2, tGluc2;

decimal sAp\_hi3, tAp\_hi3, sAp\_Lo3, tAp\_Lo3, sCholesterol3, tCholesterol3, sGluc3, tGluc3;

decimal sAp\_hi4, tAp\_hi4, sAp\_Lo4, tAp\_Lo4, sCholesterol4, tCholesterol4, sGluc4, tGluc4;

decimalAp\_hi,Ap\_Lo,Cholesterol,Gluc;

SqlDataReaderdr;

string result;

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

{

cmd = new SqlCommand("select  
Pid, Age, Gender, Height, Weight, Ap\_hi, Ap\_lo, Cholesterol, Gluc, smoke, Alco, Active from  
temptb ", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

dataGridView1.DataSource = dt;

dataGridView1.Refresh();

con.Open();

cmd = new SqlCommand("Select \* from rulestb where HeartDisease='Coronary Heart  
Disease' and Attribute='Ap\_hi' ", con);

```
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sAp_hil = Convert.ToDecimal(dr["StartRange"].ToString());
    tAp_hil = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();
```

```
con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Coronary Heart Disease' and Attribute='Ap_Lo' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sAp_Lo1 = Convert.ToDecimal(dr["StartRange"].ToString());
    tAp_Lo1 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();
```

```
con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Coronary Heart Disease' and Attribute='Cholesterol' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sCholesterol1 = Convert.ToDecimal(dr["StartRange"].ToString());
    tCholesterol1 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();
```

```
con.Open();
```

```
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Coronary Heart  
Disease' and Attribute='Gluc' ", con);  
dr = cmd.ExecuteReader();  
if (dr.Read())  
{  
    sGluc1 = Convert.ToDecimal(dr["StartRange"].ToString());  
    sGluc1 = Convert.ToDecimal(dr["EndRange"].ToString());  
}  
con.Close();  
//2
```

```
con.Open();  
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Cardiac Arrest' and  
Attribute='Ap_hi' ", con);  
dr = cmd.ExecuteReader();  
if (dr.Read())  
{  
    sAp_hi2 = Convert.ToDecimal(dr["StartRange"].ToString());  
    tAp_hi2 = Convert.ToDecimal(dr["EndRange"].ToString());  
}  
con.Close();
```

```
con.Open();  
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Cardiac Arrest' and  
Attribute='Ap_Lo' ", con);  
dr = cmd.ExecuteReader();  
if (dr.Read())  
{  
    sAp_Lo2 = Convert.ToDecimal(dr["StartRange"].ToString());  
    tAp_Lo2 = Convert.ToDecimal(dr["EndRange"].ToString());  
}  
con.Close();
```

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Cardiac Arrest' and
Attribute='Cholesterol' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sCholesterol2 = Convert.ToDecimal(dr["StartRange"].ToString());
    tCholesterol2 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();

```

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Cardiac Arrest' and
Attribute='Gluc' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sGluc2 = Convert.ToDecimal(dr["StartRange"].ToString());
    sGluc2 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();

```

//3

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='High Blood Pressure'
and Attribute='Ap_hi' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sAp_hi3 = Convert.ToDecimal(dr["StartRange"].ToString());

```

```

        tAp_hi3 = Convert.ToDecimal(dr["EndRange"].ToString());
    }
    con.Close();

    con.Open();
    cmd = new SqlCommand("Select * from rulestb where HeartDisease='High Blood Pressure'
and Attribute='Ap_Lo' ", con);
    dr = cmd.ExecuteReader();
    if (dr.Read())
    {
        sAp_Lo3 = Convert.ToDecimal(dr["StartRange"].ToString());
        tAp_Lo3 = Convert.ToDecimal(dr["EndRange"].ToString());
    }
    con.Close();

```

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='High Blood Pressure'
and Attribute='Cholesterol' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sCholesterol3 = Convert.ToDecimal(dr["StartRange"].ToString());
    tCholesterol3 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();

```

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='High Blood Pressure'
and Attribute='Gluc' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())

```

```

        sGluc3 = Convert.ToDecimal(dr["StartRange"].ToString());
        sGluc3 = Convert.ToDecimal(dr["EndRange"].ToString());
    }
con.Close();
//4

```

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Arrhythmia' and
Attribute='Ap_hi' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sAp_hi4 = Convert.ToDecimal(dr["StartRange"].ToString());
    tAp_hi4 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();

```

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Arrhythmia' and
Attribute='Ap_Lo' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sAp_Lo4 = Convert.ToDecimal(dr["StartRange"].ToString());
    tAp_Lo4 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();

```

```

con.Open();

```

```

cmd = new SqlCommand("Select * from rulestb where HeartDisease='Arrhythmia' and
Attribute='Cholesterol' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sCholesterol4 = Convert.ToDecimal(dr["StartRange"].ToString());
    tCholesterol4 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();

```

```

con.Open();
cmd = new SqlCommand("Select * from rulestb where HeartDisease='Arrhythmia' and
Attribute='Gluc' ", con);
dr = cmd.ExecuteReader();
if (dr.Read())
{
    sGluc4 = Convert.ToDecimal(dr["StartRange"].ToString());
    sGluc4 = Convert.ToDecimal(dr["EndRange"].ToString());
}
con.Close();
con1.Open();
cmd1 = new SqlCommand("truncate table resulttb", con1);
cmd1.ExecuteNonQuery();
con1.Close();

```

```

for (int rows = 0; rows < dataGridView1.Rows.Count - 1; rows++)
{
    stringconstring = @"Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\NewProject2019\PGPROJECT\JJ\Heart_
Disease\Heart_Disease\Heartdb.mdf;Integrated Security=True;User Instance=True";
using (SqlConnection con11 = new SqlConnection(constring))
{

```

```
        using (SqlCommand cmd2 = new SqlCommand("INSERT INTO resulttb  
VALUES(@Pid,@Age,@Gender,@Height,@Weight,@Ap_hi,@Ap_lo,@Cholesterol,@G  
luc,@smoke,@Alco,@Active,@Result)", con11))  
        {
```

```
            cmd2.Parameters.AddWithValue("@Pid", dataGridView1.Rows[rows].Cells[0].Value);
```

```
            cmd2.Parameters.AddWithValue("@Age", dataGridView1.Rows[rows].Cells[1].Value);
```

```
            cmd2.Parameters.AddWithValue("@Gender",  
dataGridView1.Rows[rows].Cells[2].Value);
```

```
            cmd2.Parameters.AddWithValue("@Height", dataGridView1.Rows[rows].Cells[3].Value);
```

```
            cmd2.Parameters.AddWithValue("@Weight",  
dataGridView1.Rows[rows].Cells[4].Value);
```

```
            cmd2.Parameters.AddWithValue("@Ap_hi", dataGridView1.Rows[rows].Cells[5].Value);
```

```
            cmd2.Parameters.AddWithValue("@Ap_lo", dataGridView1.Rows[rows].Cells[6].Value);
```

```
            cmd2.Parameters.AddWithValue("@Cholesterol",  
dataGridView1.Rows[rows].Cells[7].Value);
```

```
            cmd2.Parameters.AddWithValue("@Gluc", dataGridView1.Rows[rows].Cells[8].Value);
```

```
            cmd2.Parameters.AddWithValue("@smoke", dataGridView1.Rows[rows].Cells[9].Value);
```

```
            cmd2.Parameters.AddWithValue("@Alco", dataGridView1.Rows[rows].Cells[10].Value);
```

```
cmd2.Parameters.AddWithValue("@Active",  
dataGridView1.Rows[rows].Cells[11].Value);
```

```
Ap_hi = Convert.ToDecimal(dataGridView1.Rows[rows].Cells[5].Value);  
Ap_Lo = Convert.ToDecimal(dataGridView1.Rows[rows].Cells[6].Value);
```

Cholesterol

=

```
Convert.ToDecimal(dataGridView1.Rows[rows].Cells[7].Value);
```

```
Gluc = Convert.ToDecimal(dataGridView1.Rows[rows].Cells[8].Value.ToString());
```

```
if ((Ap_hi >= sAp_hi1 && Ap_hi <= tAp_hi1) || (Ap_Lo >= sAp_Lo1 && Ap_Lo <= tAp_Lo1) || (Cholesterol >= sCholesterol1 && Cholesterol <= tCholesterol1) && (Gluc >= sGluc1 && Gluc <= tGluc1))
```

```
{
```

```
result = "Coronary Heart Disease";
```

```
}
```

```
else if ((Ap_hi >= sAp_hi2 && Ap_hi <= tAp_hi2) || (Ap_Lo >= sAp_Lo2 && Ap_Lo <= tAp_Lo2) || (Cholesterol >= sCholesterol2 && Cholesterol <= tCholesterol2) && (Gluc >= sGluc2 && Gluc <= tGluc2))
```

```
{
```

```
result = "Cardiac Arrest";
```

```
}
```

```
else if ((Ap_hi >= sAp_hi3 && Ap_hi <= tAp_hi3) || (Ap_Lo >= sAp_Lo3 && Ap_Lo <= tAp_Lo3) || (Cholesterol >= sCholesterol3 && Cholesterol <= tCholesterol3) && (Gluc >= sGluc3 && Gluc <= tGluc3))
```

```
{
```

```
result = "High Blood Pressure";
```

```
}
```

```
else if ((Ap_hi>= sAp_hi4 &&Ap_hi<= tAp_hi4) || (Ap_Lo>= sAp_Lo4 &&Ap_Lo<=
tAp_Lo4) || (Cholesterol >= sCholesterol4 && Cholesterol <= tCholesterol4) && (Gluc>=
sGluc4 &&Gluc<= tGluc4))
```

```
{
```

```
result = "Arrhythmia";
```

```
}
```

```
else
```

```
{
```

```
result = "Normal";
```

```
}
```

```
cmd2.Parameters.AddWithValue("@Result", result);
```

```
con11.Open();
```

```
cmd2.ExecuteNonQuery();
```

```
con11.Close();
```

```
}
```

```
}
```

```
}//dataset
```

```
cmd1 = new SqlCommand("select * from resulttb ", con1);
```

```
SqlDataAdapter da1 = new SqlDataAdapter(cmd1);
```

```
DataTable dt1 = new DataTable();
```

```
da1.Fill(dt1);
```

```
dataGridView2.DataSource = dt1;
```

```
dataGridView2.Refresh();
```

```
}
```

```

private void button2_Click(object sender, EventArgs e)
{
    cmd1 = new SqlCommand("select * from resulttb where
Result='"+comboBox2.Text +"' ", con1);
    SqlDataAdapter da1 = new SqlDataAdapter(cmd1);
    DataTable dt1 = new DataTable();
    da1.Fill(dt1);
    dataGridView2.DataSource = dt1;
    dataGridView2.Refresh();
}

```

```

int a, b,c,d,f;

```

```

private void button1_Click(object sender, EventArgs e)
{
    con.Open();
    cmd = new SqlCommand("Select count(*) as count from resulttb where Result='Coronary
Heart Disease'", con);
    dr = cmd.ExecuteReader();
    if (dr.Read())
    {
        a = Convert.ToInt32(dr["count"].ToString());
    }
    con.Close();

```

```

con.Open();
cmd = new SqlCommand("Select count(*) as count from resulttb where Result='Cardiac
Arrest'", con);
dr = cmd.ExecuteReader();
if (dr.Read())

```

```
    {  
        b = Convert.ToInt32(dr["count"].ToString());  
    }
```

```
con.Close();
```

```
con.Open();
```

```
cmd = new SqlCommand("Select count(*) as count from resulttb where Result='High Blood  
Pressure'", con);
```

```
dr = cmd.ExecuteReader();
```

```
if (dr.Read())
```

```
{
```

```
    c = Convert.ToInt32(dr["count"].ToString());
```

```
}
```

```
con.Close();
```

```
con.Open();
```

```
cmd = new SqlCommand("Select count(*) as count from resulttb where  
Result='Arrhythmia'", con);
```

```
dr = cmd.ExecuteReader();
```

```
if (dr.Read())
```

```
{
```

```
    d = Convert.ToInt32(dr["count"].ToString());
```

```
}
```

```
con.Close();
```

```
con.Open();
```

```
cmd = new SqlCommand("Select count(*) as count from resulttb where Result='Normal'",  
con);
```

```
dr = cmd.ExecuteReader();
```

```
if (dr.Read())
```

```
{
```

```
    f = Convert.ToInt32(dr["count"].ToString());
```

```
}
```

```

    }
    b = Convert.ToInt32(dr["count"].ToString());
}

con.Close();

con.Open();

cmd = new SqlCommand("Select count(*) as count from resulttb where Result='High Blood Pressure'", con);

dr = cmd.ExecuteReader();

if (dr.Read())
{
    c = Convert.ToInt32(dr["count"].ToString());

}

con.Close();

con.Open();

cmd = new SqlCommand("Select count(*) as count from resulttb where Result='Arrhythmia'", con);

dr = cmd.ExecuteReader();

if (dr.Read())
{
    d = Convert.ToInt32(dr["count"].ToString());
}

con.Close();

con.Open();

cmd = new SqlCommand("Select count(*) as count from resulttb where Result='Normal'", con);

dr = cmd.ExecuteReader();

if (dr.Read())
{
    f = Convert.ToInt32(dr["count"].ToString());

}

```

```
con.Close();  
    Form2 ff = new Form2();  
    ff.a = a;  
    ff.b = b;  
    ff.c = c;  
    ff.d = d;  
    ff.f = f;  
    ff.Show();  
}
```

```
private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs  
e)  
{  
}  
}  
}
```

## 6.2 SCREEN LAYOUT

Form1

# Heart Disease Prediction Using Deep Learning

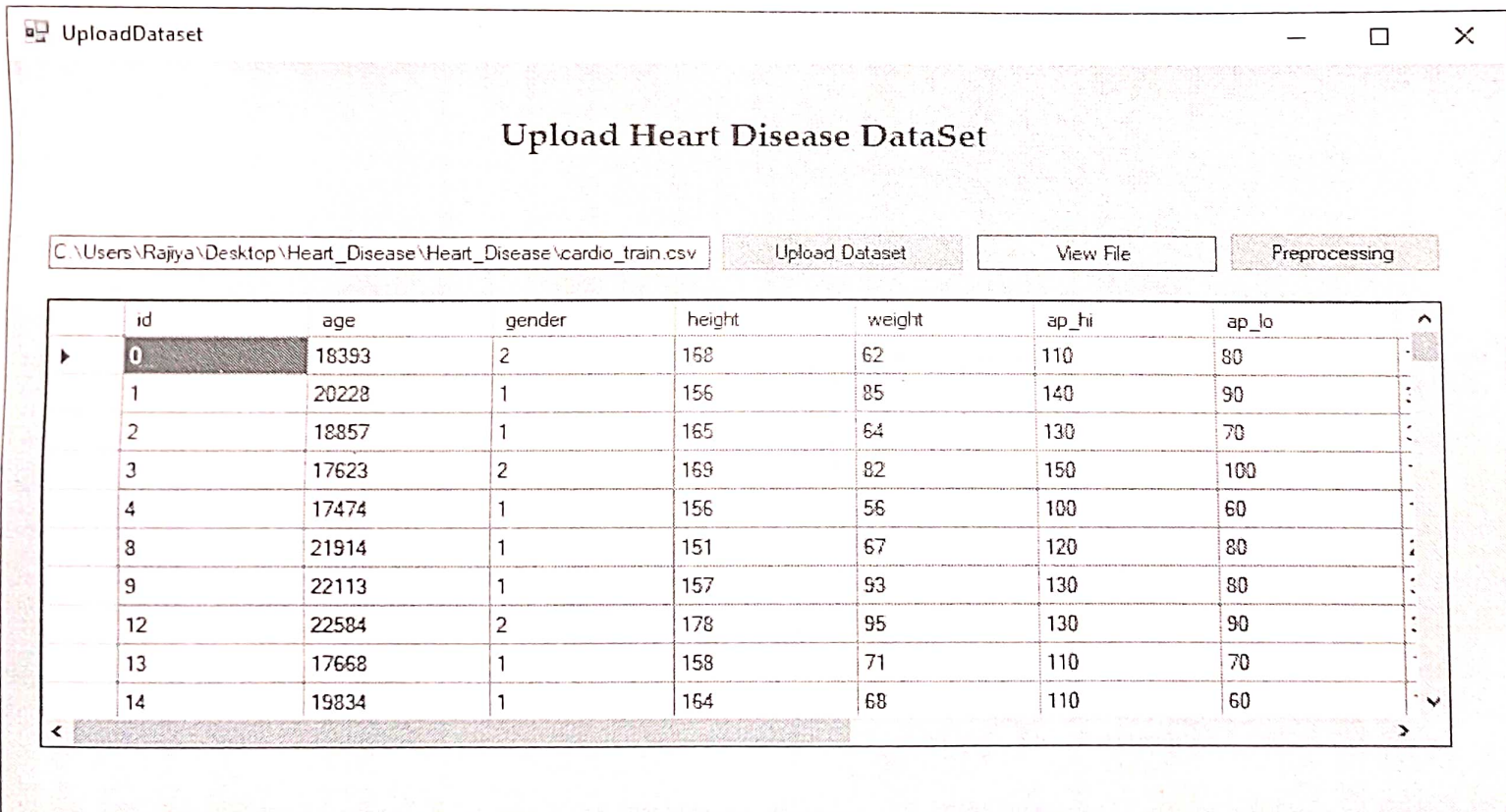
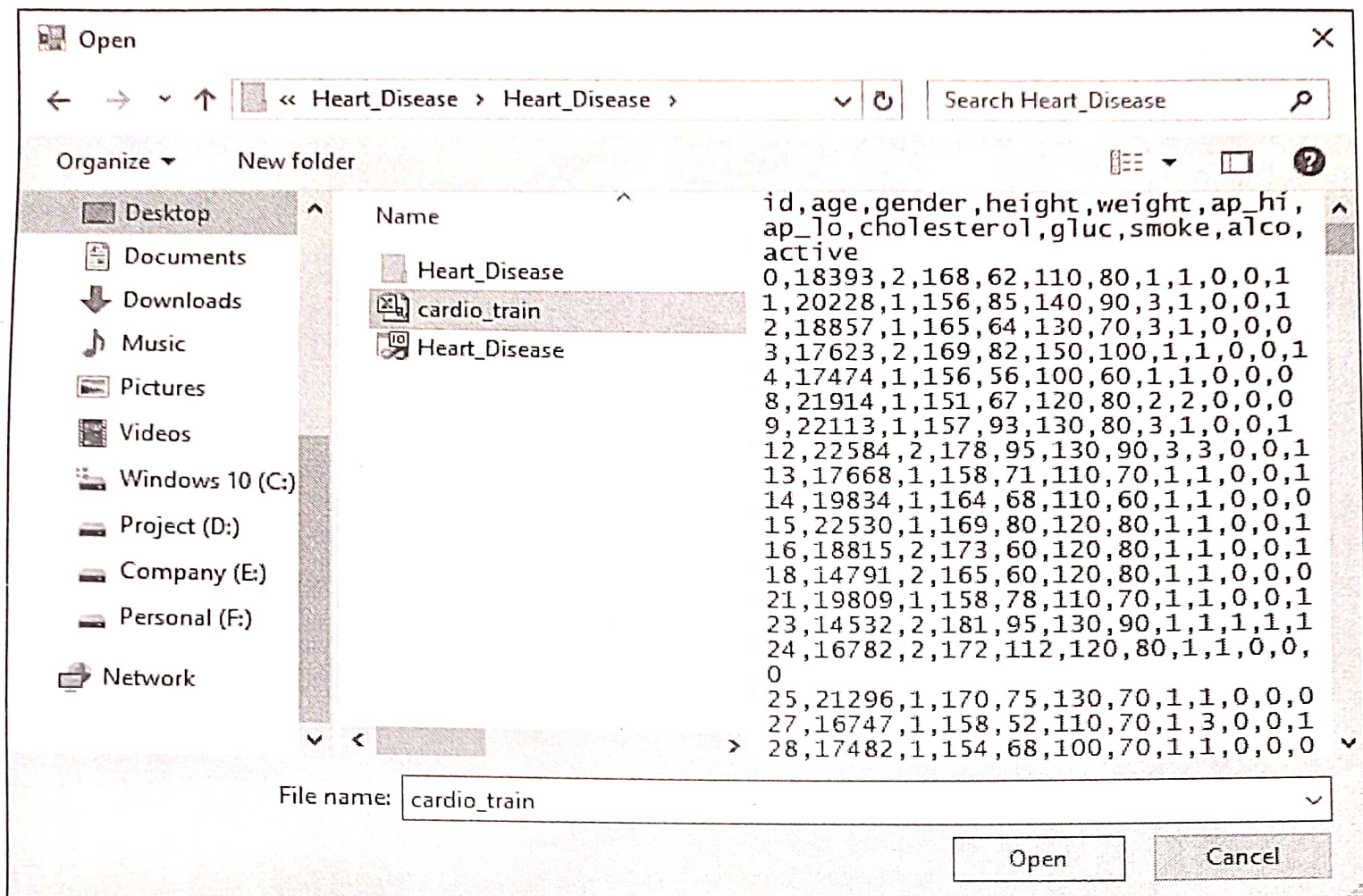
Training

Testing

UploadDataset

## Upload Heart Disease DataSet

Upload Dataset



## Preprocessing

id	age	gender	height	weight	ap_lo	ap_hi	cholesterol
0	18393	2	168	62	110	80	1
1	20228	1	156	85	140	90	3
2	18857	1	165	64	130	70	3
3	17623	2	169	82	150	100	1
4	17474	1	156	56	130	40	1
5	21914	1	151	67	120	80	2
6	22113	1	157	93	130	80	3
12	22584	2	170	95	130	90	3
13	17668	1	158	71	110	70	1
14	19034	1	164	68	110	40	1

Remove Empty Rows

Remove(s)

Rules Construction

## Preprocessing

id	age	gender	height	weight	ap_lo	ap_hi	cholesterol
0	18393	2	168	62	110	80	1
1	20228	1	156	85	140	90	3
2	18857	1	165	64	130	70	3
3	17623	2	169	82	150	100	1
4	17474	1	156	56	130	40	1
5	21914	1	151	67	120	80	2
6	22113	1	157	93	130	80	3
12	22584	2	170	95	130	90	3
13	17668	1	158	71	110	70	1
14	19034	1	164	68	110	40	1

Remove Empty Rows

4Remove(s)

Rules Construction

id	age	gender	height	weight	ap_lo	ap_hi	cholesterol
0	18393	2	168	62	110	80	1
1	20228	1	156	85	140	90	3
2	18857	1	165	64	130	70	3
3	17623	2	169	82	150	100	1
4	17474	1	156	56	130	40	1
5	21914	1	151	67	120	80	2
6	22113	1	157	93	130	80	3
12	22584	2	170	95	130	90	3
13	17668	1	158	71	110	70	1

## Layer Construction

id	StartRange	EndRange	Attribute	HeartDisease
17	170	190	Ap_hi	Coronary Heart D...
18	50	60	Ap_Lo	Coronary Heart D...
19	1	3	Cholesterol	Coronary Heart D...
20	1	3	Gluc	Coronary Heart D...
21	160	170	Ap_hi	Cardiac Arrest
22	45	50	Ap_Lo	Cardiac Arrest
23	1	3	Cholesterol	Cardiac Arrest
24	1	3	Gluc	Cardiac Arrest
26	150	160	Ap_hi	High Blood Press ...
27	40	45	Ap_Lo	High Blood Press ...
31	140	150	Ap_hi	Arrhythmia
29	1	3	Cholesterol	High Blood Press ...
30	1	3	Gluc	High Blood Press ...
32	35	40	Ap_Lo	Arrhythmia
34	1	3	Cholesterol	Arrhythmia
35	1	3	Gluc	Arrhythmia

Next

## New User Registration

Name

RAJIYA

Gender

☐

Male

☒

Female

Mobile

9874563210

Email

rajiya@gmail.com

UserName

rajiya

Password

••••••

Submit

Clear

×

Record Saved!

OK

UserLogin

User Login Here..!

User Name

rajiya

Password

\*\*\*\*\*

NewUser

Login

Clear

UserHome

Heart Disease Prediction Using Deep Learning

Ap\_hi

(90 to 190)

Ap\_lo

(60 to 100)

Cholesterol

1

xxx

Glucose

1

Smoke

▼

Alcohol

▼

Find Result

Clear

## Heart Disease Prediction Using Deep Learning

Ap\_hi  (90 to 190)

Ap\_lo  (60 to 100)

Find Result

Cholesterol

High Blood Pressure

Glucose

Smoke

Alcohol

Find Result

Clear



High Blood Pressure

OK

# FUTURE ENHANCEMENT

## **7. FUTURE ENHANCEMENT**

In future the of disease prediction can be improved by integrating data mining techniques with deep learning algorithms.

This project can also be extended to predict other critical diseases like stroke tumour etc.

CONCLUSION

## **8. CONCLUSION**

This project proposal and implemented a heart disease predication algorithm in the field of deep learning. The focus is on using different algorithms and combinations of several target attributes for intelligent and effective heart disease prediction by for extracting valuable medical rules hidden in medical data and acts as an important role in disease prediction and clinical diagnosis.

There is an increasing interest in using classification to identify diseases. Classification algorithm is very sensitive to noisy data. If any noisy data is present then it causes very serious problems It not only slows down the task of classification algorithm but also degrades its performance. Hence, before applying classification algorithm it must be necessary to remove all those attributes This project implemented classification rule algorithm for classifying datasets which are uploaded by user. By analyzing the experimental results it is observed that the Multi-layer perceptron technique has yields better result than other techniques.

# BIBLIOGRAPHY

## **BIBLIOGRAPHY**

### **BOOK REFERENCE:**

- N.J. Belkin and W.B. Croft, "Information Filtering and Information Retrieval: Two Sides of the Same Coin?"Comm.
- Advanced .NET Remoting in VB.NET (Ingo Rammer, Apress, July 2002)
- ASP to ASP.NET Migration Handbook (Christian Nagel et al, Wrox, January 2003)
- Beginning Visual C# (Christian Nagel et al, Wrox, September 2001)
- Data-Centric .NET Programming (Christian Nagel et al, Wrox, December 2001)
- Professional .NET Network Programming 2<sup>nd</sup> Edition (Christian Nagel et al, Wrox, September 2004)
- P.W. Foltz and S.T. Dumais, "Personalized Information Delivery: An Analysis of Information Filtering Methods," Comm.
- S. Pollock, "A Rule-Based Message Filtering System," ACM Trans. Office Information Systems.

### **WEBSITE REFERENCE**

- [www.csharpcornar.com](http://www.csharpcornar.com)
- [www. Microsoft.com/sql](http://www.Microsoft.com/sql)
- [www.databasejournal.com](http://www.databasejournal.com)
- [www.microsoft.com/vcsharp](http://www.microsoft.com/vcsharp)