GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Disease Symptoms Analysis

A Report for the Evaluation 3 of Project 2

Submitted by

PRIYANK BHARDWAJ

(1613101522/16SCSE101348)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

MR. JAYDEEP KISHORE

Assistant Professor

APRIL / MAY- 2020

GALGOTIAS
UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report " **DISEASE SYMPTOMS ANALYSIS** " is the bonafide work of " **PRIYANK BHARDWAJ** " who carried out the project work under my supervision.

SIGNATURE OF HEAD	**SIGNATURE OF SUPERVISOR**
DR MUNISH SHABARWAL ,	**MR JAYDEEP KISHORE**
PhD (Management), PhD (CS)	**Assistant Professor,**
Professor & Dean,	**School of Computer Science &**
School of Computing Science & Engineering	**Engineering**

1. ABSTRACT AND SCOPE OF PROJECT 4
1.1 Purpose 5
1.2 Objective 9
1.3 Project Scope 11
1.3.1 Member Module 11
1.3.2 Administration Module 12
1.4 Technologies 13
1.4.1 Operating Environment 13
1.4.2 Deployment Environment 14
1.4.3 Development Tools and Technologies 15
1.4.4 Development Environment 18
2. OVERALL DESCRIPTION 19
2.1 User Characteristics 19
2.1.1 Member 19
2.1.2 System User 19
2.1.3 Administrator 19
2. 2.Assumptions 19
3. SYSTEM FEATURES 20
3. 1.Systern features 20
3.1.1 .Description: 20
3.1 .2.Constraints 20
3.1.4 Action/result 20
4. REQUIREMENT ANALYSIS 21
4.1 Business goals 21
4.2 Design goals 22
4.3 Business Requirements 23
4.4 User Requirements 25
4.4.1 Member 25
4.4.2 System Users 25
4.4.3 Administration 25
4.5 Operational requirements 26
4.6 System Requirements 26
4.7 System Constraints 27
4.8 Behavioral Description 27
5. NONFUNCTIONAL REQUIREMENTS 28
5.1 Performance Requirements 28
5.2 Availability Requirements 29
5.3 Reliability Requirement 31
5.4 Scalability Requirement 32
5.5 Security Requirements 36
Security Features of .NET Technologies 38
5.6 Interoperability 41

6. PROJECT MANAGEMENT **42**

6.1 Development Methodologies 42
6.1.1 Waterfall model. 42
6.1.2 Spiral model. 42
6.1.3 MSF Process Model: 43
6.2 Project Development Life Cycle (PDLC) 43
6.2.1 Envisioning 43
6.2.2 Planning 44
6.2.3 Developing and Designing the System 44
6.2.4 Stabilizing 44

7. ESTIMATION AND PLANNING **45**

7.1 Planning 45
7.1 Application of the MSF model in our project 45
6.2 Estimation 49
6.2. 1 Basic COCOMO 49
6.2.2 Function Point Estimation 51

7. PRELIMINARY DESIGN **53**

7.1 Use Case 53
7.2 Specification of actors 54
7.2.1 Member 54
7.2.2 System User 54
7.2.3 Administrator 54
7.3 Specification of Use Cases 55
7.3.1 Use Case Member Registration 55
7.3.2 Use Case. Registration 59
ER DIAGRAM 66

8 CODING AND WEB PAGES **56**

8.1 Disease Symptoms Analysis Member Home Page 59
8.2 Disease Symptoms Analysis – Member Login Page 68
8.3 Disease Symptoms Analysis – My Account 75
8.4 Disease Symptoms Analysis – Member Registration and Edit Profile 80
8.5 Disease Symptoms Analysis – Change Password 85
8.6 Disease Symptoms Analysis – Forget Password 95

9. TEST PLAN **96**

Introduction 97
9.1 Test Scope 97

9.2 Test Strategy 98
9.3 Preconditions 98
9.4 Test Priorities 99
9.5 Test Techniques 99

9.6 Test Organization 100
9.7 Deliverables 100
9.8 Test Environment 101
9.9 Testing Automation Software 102

10. FUTURE ENHANCEMENT 103

11. BIBLIOGRAPHY 103

11.1 Websites 104

1.Problem Definition and scope of project

1.1 Purpose

The online disease symptoms analysis will permit it’s user to register on it’s platform and allow them to update all the info and all it’s disease data. And after the data is updated they can also edit that data and they can also insert the dataset of their medical history and after using so many existing algorithm they will check from it’s data that the user is having which disease.

After the user has known the disease he\she is having the site also suggest the various main causes which are mainly responsible for that disease he\she is having. And after knowing the causes and examining all the factors the user can lead to the fact which he\she can change in the lifestyle.

For Example if in an dataset the main factor that leads to the heart disease in very young may
depends on it’s smoking and alcohol habits and the user can easily know that the causes which having the same food habits and same symptoms. and after all the suggestions he\she will avoid drinking and for a while.

So, like this project not only deals with the fact that which heart disease the user may have but it also deals to the facts that may be leading to that heart disease as we discussed above.

1.2 Objective

Online Disease Symptoms Analysis is complete end to end solution to cover all aspects of disease that the patient may had.

The basic objective of developing this project is:

- Provides complete solution, search disease symptoms, , info about diseases, including methods, Disease management, Member management, Logical access management.
- All disease are categorized by categories, and provide their disease search facilities; search can be done by using disease name, disease description.
- Allow user to search its continent and also it's country because different countries have different norms.
- Add multiple disease to medical history, can also edit the disease symptoms .
- Can able to store diseases image with all patients.
- Member can list their problems in the main menu.
- Member can see the solutions given to others in my account page.
- Administrator can also add diseases or edit disease listing
- All administrator option must perform from web interface only includes management .
- Disease Descriptions can include HTML formatting: We know that having complete control over each description is important, and no two diseases are alike. System administration can enter disease descriptions.
- Member can view Search History.

1.3.1 Member Module

An enhanced interface for Member to registration, edit profile, login, search disease, add symptoms to profile , check the comments. Following modules pages are available for Member.

CM-1: Home – It's the default page for the location. Member can view Disease category wise or can search disease from. All links are available in this page.

CM-2: Login – Member need to login to view his account information and search the disease. If Member forgets his password he can revisit old password from Forget password link.

CM-3: Register – New Member need to register to make it's profile. Type all the small prints of the Member like email id, name, address, contact details and submit. Member got to type valid email id because if he forget password old password are going to be sent to the present email address.

CM-4: My Account – It shows the small prints of currently logged Member details and other links like Edit Profile, Logout, and alter Password.

CM-5: Change Password – Member must type his previous password to change the password with new password.

CM-6: Disease – Member can view the Disease by disease category or search disease from the database. It display disease image, disease name, disease description. Member can view disease by clicking view It link.

1.1 Technologies

1.4.1 Operating Environment

OE-1: The Disease Symptoms Analysis web application will operate with the following Web Browsers: Microsoft Internet Explorer version 5.0, 6.0. 7.0

OE-2: The Disease Symptoms Analysis web application shall operate on a server running the latest versions of IIS (Internet Information Server).

OE-3: The Disease Symptoms Analysis web application shall permit user access from Internet connection.

OE-4: Operating System:

OE-5: Software requirements: SQL Server , .net framework .

OE-6: Languages used are asp.net using c# and scripting is completed using JavaScript. OE-

7: Hardware Requirements: 256(minimum)/512(recommended) MB RAM

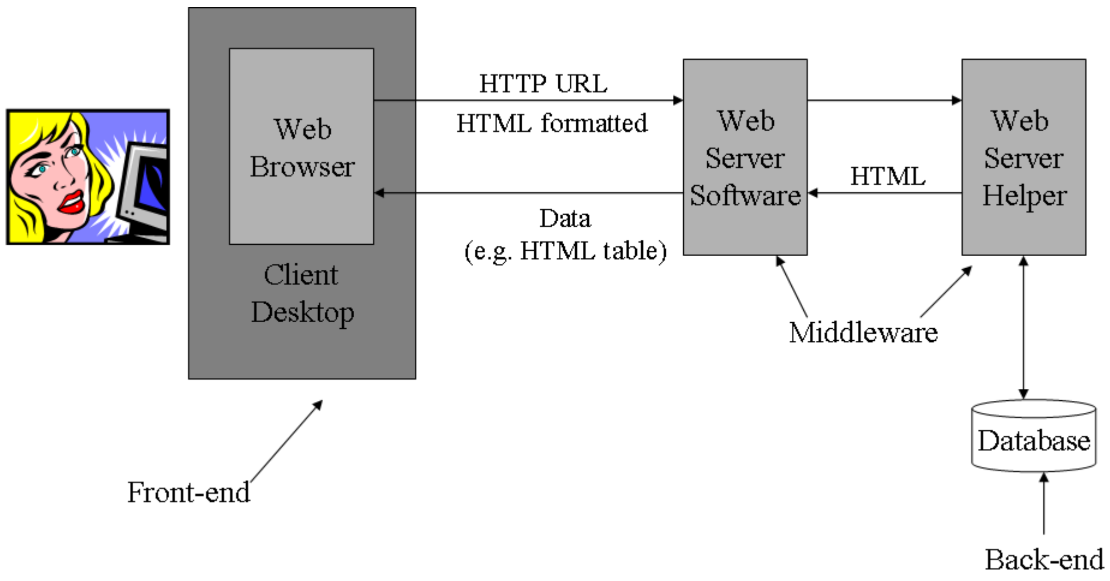
OE-8: Hard disc- n GB depending upon the need to store data minimum of 25GB.

1.4.2 Deployment Environment

DE-1: Database Server
OS – Win 2003 Enterprise Server
SQL Server 2005
HDD – Min 10 GB, Recommended 25 GB RAM
– Min 2 GB, Recommended 4 GB Processor -
Pentium Dual Xenon Processor

DE-2: Application Server
OS – Win 2003 Enterprise Server
IIS – Internet Information Server
HDD – Min 5 GB, Recommended 10 GB RAM –
Min 2 GB, Recommended 4 GB Processor -
Pentium Dual Xenon Processor

DE-3: The Disease Symptoms Analysis web application will operate with the following Web Browsers: Microsoft Internet Explorer version 5.0, 6.0. 7.0.



[Disease Symptoms Analysis System Architecture]

1.4.3 Development Tools and Technologies

1.1.1.1 DT-1: ASP.Net

As we need to develop Web Application for Disease Symptoms Analysis Application. We will used ASP.Net as it is the new Microsoft technology to develop any application which support and integrate other server disease for the internet. By ASP.Net we can develop in web application by .NET framework and manage environment with any .NET support language like VB.Net and C#.

1.1.1.2 DT-2: C#

.NET is built on the Windows Server System to take major advantage of the OS and which comes with a host of different servers which allows for building, deploying, managing and maintaining Web-based solutions. The Windows Server System is designed with performance as priority and it provides scalability, reliability, and manageability for the global, Web- enabled enterprise. The Windows Server System integrated software diseases are built for interoperability using open Web standards such as XML and SOAP.
.NET is a "Software Platform". It is a language-neutral environment for developing rich .NET experiences and building applications that can easily and securely operate within it. When developed applications are deployed, those applications will target .NET and will execute wherever .NET is implemented instead of targeting a particular Hardware/OS combination. The components that make up the .NET platform are collectively called the
.NET Framework.

The .NET Framework is a managed, type-safe environment for developing and executing applications. The .NET Framework manages all aspects of program execution, like, allocation of memory for the storage of data and

instructions, granting and denying permissions to the application, managing execution of the application and reallocation of memory for resources that are not needed.
The .NET Framework is designed for cross-language compatibility. Cross- language compatibility means, an application written in Visual Basic .NET may reference a DLL file written in C# (C-Sharp). A Visual Basic .NET class might be derived from a C# class or vice versa.
The .NET Framework consists of two main components:
Common Language Runtime (CLR)
Class Libraries

The advantage of C# includes Powerful Windows-based Applications Building Web-based Applications Simplified Deployment

- Powerful, Flexible, Simplified Data Access
- Improved Coding
- Direct Access to the Platform
- Full Object-Oriented Constructs
- XML Web Services
- COM Interoperability

1.1.1.3 DT-3: SQL Server 2005

Disease Symptoms Analysis Application uses SQL Server 2005 as storing the data. Microsoft SQL Server 2005 as our database and it has so many features which is ideal for our dot net based application. Features Includes

- Support for Multiple Platforms
- Integration with Windows Back office servers
- Integration with Microsoft .NET Enterprise Servers
- Scalability
- Replication
- Centralized Management
- Reliability
- Automating Task

1.4.4 Development Environment

1.1.1.4 DE-1: 1. Visual Studio 2005 IDE

Most advanced integrated development environment for developing .NET application. Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.
Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It allows plug-ins to be added that enhance the functionality at almost every level - including adding support for source control systems (like Subversion and Visual SourceSafe) to adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports languages by means of language services, which allow any programming language to be supported (to varying degrees) by the code editor and debugger, provided a language-specific service has been authored. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), and C# (via Visual C#). Support for other languages such as Chrome, F#, Python, and Ruby among others has been made available via language services which are to be installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Language-specific versions of Visual Studio also exist which provide more limited language services to the user. These individual packages are called Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

2. **Overall Description**

2.1 User Characteristics

2.1.1 Member

Able to register, login, edit personal information, disease details, find the disease information
with updated diseases symptoms.

2.1.2 System User

View disease and symptoms details, Update personal details, Generate and view rep.

2.1.3 Administrator

Administrator will have all the access rights. Administrator can insert new disease, users and view
the reports respective access rights to them.

2. 2.Assumptions

1) System User and Administrator can communicate with each other via emails.

3. **System Features**

3. 1.Systern features

1.1.23.1.1 . Description:

A web based Disease Symptoms application which will use to search diseases, over the internet. Proposed Disease Symptoms Analysis application system is end to end solution and generation of reports for the management.

3.1 .2.Constraints

Linking and integration with legacy system for accounting. Integration and database through Web Services. Connecting to third-party OLAP applications for generating reports. Develop sophisticated system to automate the add multiple payment gateway.

1.1.33.1.4 Action/result

Action	**Result**
Member registers himself	After validation, naive member becomes registered member.
User/Admin Logins via Login page	According to the access rights redirected to the accordance page.
Register new User (Admin)	Redirected to User Details page and User Access page.
Searching Diseases	Redirected to Disease Details page
Add New Disease (Admin)	Redirected to Disease Info page

4. Requirement Analysis

The requirement analysis outlines the approach the development team will take to meet the goals of the project and provides the basis for proceeding to the planning phase. After identifying the business problem and defining the vision and scope, the team creates the solution concept that explains in general terms how the team intends to meet the requirements of the project.

[Solution concept of Disease Symptoms Analysis]

For a project to be successful, it is essential that we correctly identify the goals of the project. Project goals can be categorized as follows:

- Business goals
- Design goals

4.1 Business goals

Business goals represent what the organization wants to achieve with the solution. Business goals form the basis for determining the success criteria of the solution. The purpose of defining business goals is to clearly articulate the objectives for the project and to ensure that your solution supports those business requirements. The team needs to determine the best method for identifying the goals and agreeing on them.

Throughout the life of the project, the team makes tradeoffs among resources, schedule, and features. It is important that business goals are prioritized in a way that will allow the team to have a clear understanding about which ones the Member believes are most important, in case some of the goals cannot be achieved.

4.2 **Design goals**

Design goals are similar to business goals in many ways. The difference is that design goals focus more on the attributes of the solution and less on what the solution will accomplish for the business. Design goals address not only what the team wants to accomplish but also what the team is not trying to accomplish with the solution. As with business goals, you need to prioritize design goals so that the team knows which goals must be accomplished, in case the project cannot achieve all of them.

Consider the case of this Disease Symptoms Analysis Web site. Some of the design goals for the online shopping cart might include:

- Improve the user experience by reducing page-download wait times to 5 seconds or less.
- Limit dependency on connectivity with the server.
- Reduce the time and level of effort required for a user to complete the online registration.

- The service and all supporting applications must be localized for users worldwide.
- The service must have an availability of 99.99 percent.
- The service cannot lose data.
- The service must permit access only by authorized users.

4.3 Business Requirements

The following preliminary lists are based on initial interviews and study of existing manual system,

The business goal for the application is to support an increase the diseaseivity and complete automation of existing manual or semi automatic Disease Symptoms Analysis Application. Business requirements are discussed in the Scope section, with the following additional detail:

- Improve the search facility and Member should get all the information in a second.
- Admin user must able to upload diseases image.
- The Administrator should be able to enter or update master information details from web interface only.
- System Administrator must able to control the access rights by each user as per requirement.
- The application should support the capability to use multi user environment.
- The Admin user should able to generate all type of reports as and when required by the management.
- All Diseases must be categorized before listing.
- Every page on the Web site must display a search option with appropriate search controls and a navigation bar on the left side.

- The Member must be able to search for disease by using a part of a disease description.

- The home page must display diseases that are currently in the region, and it must include a picture and description of each disease.

- A Disease Details page must display information about a single disease model. It must contain the disease name and description, a picture.

4.4 **User Requirements**

User requirements are categorized by user type.

4.4.1 Member

- Able to edit personal information, disease details.
- Able to find the disease information with updated symptoms status.
- Able to apply change symptoms for selected disease.
- Able to modify the disease.
- Able to view previous search and symptoms details in My Account page.

4.4.2 System Users

- View disease and symptoms details.
- Update personal details
- Generate and view reports.

4.4.3 Administration

- Administrator must able to add new system users, modify details.
- Able to add and update disease details.
- Able to add and update symptoms details.
- Generate Reports as per requirement.

4.5 **Operational requirements**

The following requirements provide a high-level view of how the system will run:

- Processor usage should not exceed 80 percent during concurrent uses.
- Backups will occur incrementally throughout the day.
- A full weekly backup is required to WORM drives.
- Ensure that information is easy to access either, and meaningful for the system users and the company.
- Minimize the technical knowledge that system users and Member need to access the data, generate ad hoc queries, search and view disease information.
- Any change to information must be reflected immediately, and the changes must be propagated to the search engine so that system users that perform searches see this new information.
- The application should work with the existing communications and networking infrastructure.
- The application should deploy with a minimum of additional operational processes, manual or otherwise.

4.6 System Requirements

These are additional constraints from a system perspective:

- The administrator must be able to monitor everything from the IT department.
- The information must be accessible by everyone in the company in intranet and in internet for the members as per the rights specify.
- All data especially disease symptoms must be up to date.

4.7 System Constraints

Constraints indicate the parameters to which the final business solution must adhere. They are aspects of the business environment that cannot or will not be changed. Often, these constraints become design goals for the application. If constraints are not identified properly, the project team might design a disease that cannot be deployed within the business.

Examples of possible constraints that you should document include:

- Budget limitations
- Characteristics of earlier supporting systems
- Network system architecture
- Security requirements
- Operating systems
- Planned upgrades to technologies
- Network bandwidth limitations
- Maintenance and support agreements and structures
- Knowledge level of development or support staff
- Learning limitations of users

4.8 Behavioural Description

The Member of Disease Symptoms Analysis system registers himself by simply accessing the system's home page and fills the simple form. When the Member register, he can able to , search the disease and able to add symptoms to profile. Member can view the *my account* page to last search details.

A System User can view and generate various reports update the profile.

Administrator is the user of the system with administrative rights. Administrator manage disease to search and defines master information like symptoms. It creates user and specifies appropriate access rights to them. Monitor the system to view and generate various reports.

5. Nonfunctional Requirements

5.1 Performance Requirements

An application's performance is defined by metrics such as transaction throughput and resource utilization. A user might define an application's performance in terms of its response time. No more than 5-percent degradation in average query response is allowed while all concurrent users are using the system. Processor utilization should not exceed 80 percent during all concurrent users are using the system.

We must define performance requirements before the team proceeds to the developing phase. To define a good performance requirement, we must identify project constraints, determine services that the application will perform, and specify the load on the application.

PR-1: Identifying constraints - Constraints in the project include budget, schedule, infrastructure, and the choice of development tools or technologies. For example, we might need to deploy this Disease Symptoms Analysis application by a specific date. We might also need to use a specific development tool because the team has expertise in that tool only. We might not be able to design and develop applications that are processor intensive because the client computers do not have adequate hardware. We need to design an application so that it meets its performance goals within the limitations of the constraints. Instead of changing some aspects of a project to improve performance, you can modify aspects of the project that are not constrained to determine how we can improve performance. For example, can the team be trained so that they can create components by using a different tool? Can data access be improved by changing the data access technology?

PR-2: Determining features - The features of this application correspond to use cases and usage scenarios. We can identify the usage scenarios that affect the performance of the application and, for each such scenario, specify what the user does and what the application does in response, including how databases and other system services are accessed. In addition, you need to determine how often each feature will be used. This information can help you create tests for measuring performance that resemble actual usage of the application as closely as possible.

PR-3: Specifying the load - We can specify the load of this Disease Symptoms Analysis application as the number of clients that will use the application. In addition, we can examine how the load might vary over time. For example, the number of requests for this symptoms site will be higher during certain times of year. We can use the load to define the performance metrics of this application.

5.2 Availability Requirements

Availability is a measure of how often the application is available to handle service requests as compared to the planned run time. Availability also takes into account repair time because an application that is being repaired is not available for use.

Designing for availability includes anticipating, detecting, and resolving hardware or software failures before they result in service errors, faults, or data corruption, thereby minimizing downtime. To ensure availability, provide multiple routes to application services and data. Use only tested, proven processes (both automated and people-based) that support the application throughout its life cycle.

In addition to unplanned downtime, planned downtime must be reduced. Planned downtime can include maintenance changes, operating system upgrades, backups, or any other activity that temporarily removes the application from service.

Availability of an application also depends on its reliability. For a highly available and reliable application, you need a reliable foundation: good application design, rigorous testing, and certification. Some of the techniques used for designing for availability include:

AR-1: Reduce planned downtime - To avoid planned downtime, use rolling upgrades. For example, to update a component on a server, move the server's resource groups to another server, take the server offline, update the component, and then bring the server online. Meanwhile, the other servers handle the workload, and this application experiences no downtime. You can use this strategy in an application that scales out.

AR-2: Reduce unplanned downtime with clustering - Clustering is a technology for creating high-availability applications. A cluster consists of multiple computers that are physically networked and logically connected using cluster software. By using clustering, a multiple server Web site can withstand failures with no interruption in service. When the active server fails, the workload is automatically moved to a passive server, current client processes are switched over, and the failed application service is restarted automatically. If a resource fails, Members connected to that server cluster might experience a slight delay, but the service will be completed. Cluster software can provide failover support for applications, file and print services, databases, and messaging systems that have been designed as cluster-aware and assigned to a cluster.

AR-3: Use network load balancing - Network load balancing (NLB) is used to distribute traffic evenly across available servers. NLB also helps increase the availability of an application: if a server fails, you can use NLB to redefine the cluster and direct traffic to the other servers. NLB is especially beneficial .

applications that link external clients with transactions to data servers. As client traffic increases, you can scale out the Web server farm by adding up to 32 servers in a single cluster. NLB automatically detects server failures and redirects client traffic to the remaining servers, all the time maintaining continuous, unbroken client service.

AR-4: Use redundant array of independent disks (RAID) for data stores. - RAID uses multiple hard disks to store data in multiple places. If a disk fails, the application is transferred to a mirrored data image and the application continues running. The failed disk can be replaced without stopping the application.

AR-5: Isolate mission-critical applications - An application is constantly performing tasks and requesting resources such as network communications, data access, or process threads. Each of these resource requests can affect the performance and availability of applications sharing the same resources. If an application shares these services on the same servers, the workload and throughput characteristics for these servers might change unfavorably. It is recommended that mission-critical applications use dedicated infrastructures and private networks.

AR-6: Use queuing - Queuing enables your application to communicate with other applications by sending and receiving asynchronous messages. Queuing guarantees message delivery; it does not matter whether the necessary connectivity currently exists (with mobile applications, for example). Queuing removes a failure point from the application. Queuing is also a solution for managing peak workloads that can require a lot of hardware. In addition, by increasing the number of routes for successful message delivery, an application can increase the chances for successful and immediate message completion.

Calculation of availability

Measurement Types for Calculating Availability		
Name	Calculation	Definition
Mean Time Between Failure (MTBF)	Hours / Failure Count	Average length of time the application runs before failing
Mean Time To Recovery (MTTR)	Repair Hours / Failure Count	Average length of time needed to repair and restore service after a failure

The formula for calculating availability is:

Availability = (MTBF / (MTBF + MTTR)) × 100

For example, the typical availability requirement for this Disease Symptoms Analysis application is that the site is available 24 hours a day, 7 days a week. If you assume 1000 continuous hours as a checkpoint, two 1-hour failures during this time period results in availability of:

((1000 / 2) / ((1000 / 2) + 1)) × 100 = (500 / 501) × 100 = .998 × 100 = 99.8%.

A popular way to describe availability is by the nines, for example, three nines for 99.9 percent availability. However, the implication of measuring by nines is often misunderstood. We need to do the arithmetic to discover that three nines (99.9 percent availability) represent about 8.5 hours of service outage in a single year. The next level, four nines (99.99 percent), represents about 1 hour of service outage in a year. Five nines (99.999 percent) represent about 5 minutes of outage per year.

5.3 **Reliability Requirement**

The reliability of an application refers to the ability of the application to provide accurate results. Reliability and availability are closely related. While availability measures the capacity to handle all requests and to recover from a failure with the least loss of access to the application. Users bypass unreliable Web sites, resulting in lost revenue and reduced future sales. In addition, the expense of repairing corrupted data increases the cost of application failure. Unreliable systems are also difficult to maintain or improve because the failure points are typically hidden throughout the system. In addition, existing disaster recovery and backup plans and procedures must be revised to incorporate the Disease Symptoms Analysis Application.

To design for reliability, you need to examine the application's expected usage pattern, create a reliability profile, and create a solution that meets the profile. You must examine how a particular service is provided, evaluate failure scenarios, and design preferred alternatives. In addition, you need to consider the application's interactions with other applications.

It is difficult to identify reliability problems and solutions for a system that has not been developed. However, we can begin by analyzing the currently running applications in the organization. We can use this information to design a reliable solution.

A reliable solution ensures error-free data input, data transformations, state management, and non-corrupting recovery from any failure conditions. Creating a high-reliability application depends on the entire software development lifecycle, from the planning phase, through development and testing, to deployment and stabilizing.:

- Using a good architectural infrastructure
- Including management information in the application
- Using redundancy
- Using quality development tools
- Using reliability checks that are provided by the application
- Implementing error handling

5.4 Scalability Requirement

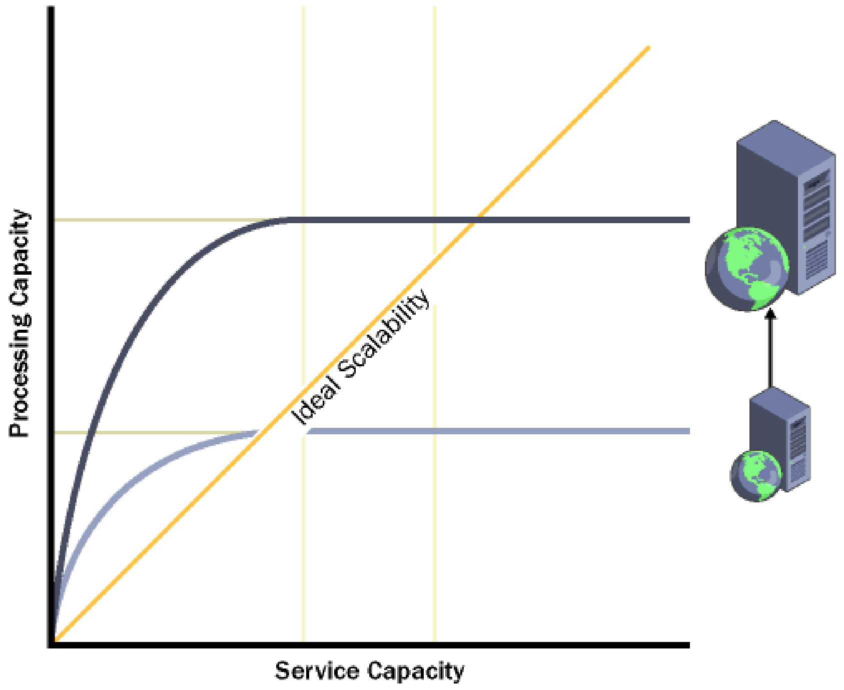
Scalability is defined as the capability to increase resources to produce an increase in the service capacity. In a scalable application, you can add resources to manage additional demands without modifying the application itself.

A scalable application requires a balance between the software and hardware used to implement the application. You might add resources to either software or hardware to increase the scalability of the application. Adding these resources might produce a benefit; however, it could also have a negative or null effect, with the application showing no significant increase in service capacity. For example, you might implement load balancing in an application. This will help only minimally if the application has been written to make synchronous method calls or to retrieve lengthy datasets in response to a user's request.

Disease Symptoms Analysis Application an average load of 1500 concurrent users after the system is fully operational, and expects that to grow by 25 percent each year for the next five years.

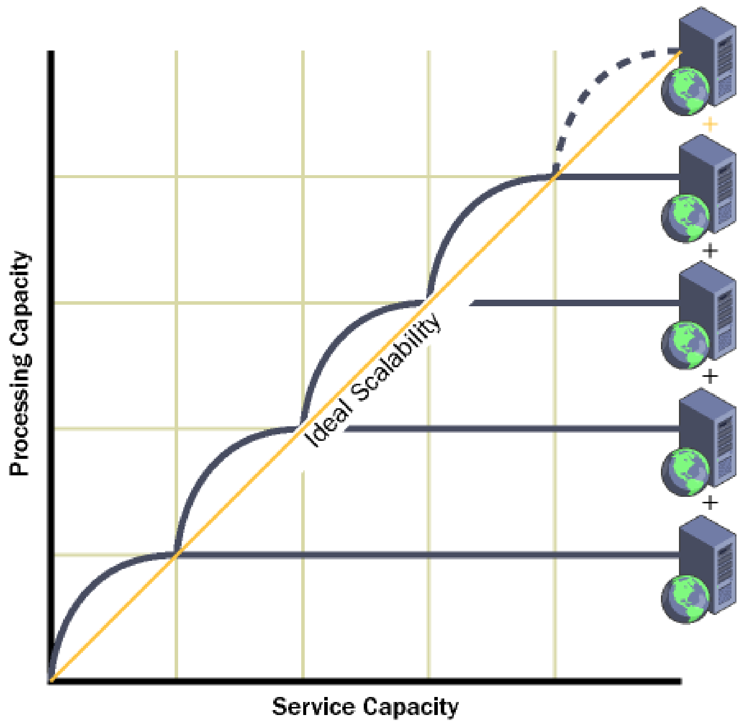
The two most common approaches to scalability are:

> **SR-1: Scaling up** - Refers to achieving scalability by improving the existing server's processing hardware. Scaling up includes adding more memory, more or faster processors, or migrating the application to a more powerful computer. The primary goal of scaling up an application is to increase the hardware resources available to the application. Typically, you can scale up an application without changing the source code. In addition, the administrative effort does not change drastically. However, the benefit of scaling up tapers off eventually until the actual maximum processing capability of the machine is reached.



[Scaling Up]

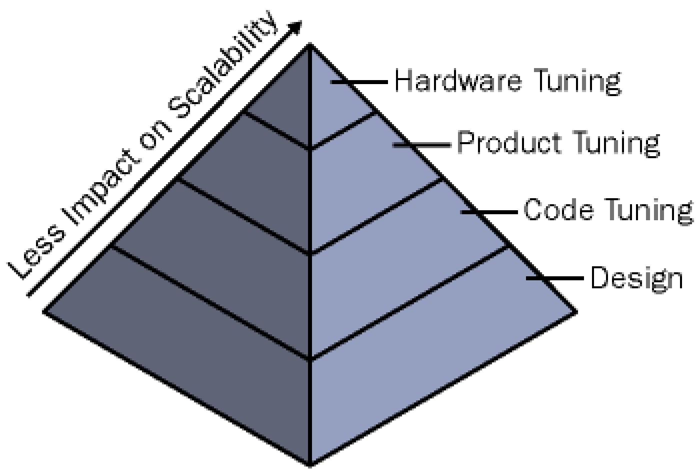
SR-2: Scaling out - Refers to distributing the processing load across more than one server. Although scaling out is achieved by using multiple computers, the collection of computers continues to act as the original device configuration from the end-user perspective. Again, the balance between software and hardware is important. The application should be able to execute without needing information about the server on which it is executing. This concept is called location transparency. Scaling out also increases the fault tolerance of the application.



[Scaling Out]

Good design is the foundation of a highly scalable application. The planning phase has the greatest impact on the scalability of an application.

Bellow figure illustrates the role of design, code tuning, disease tuning, and hardware tuning in the scalability of an application. Design has more impact on the scalability of an application than the other three factors. As you move up the pyramid, the impact of various factors decreases. The pyramid illustrates that effective design adds more scalability to an application than increased hardware resources.



[Scalability pyramid]

To design for scalability, we need to following guidelines:

SR-3: Design processes such that they do not wait - A process should never wait longer than necessary. A process can be categorized as synchronous or asynchronous. A synchronous process waits for another process to complete before it continues. Such processes must wait for another process to succeed or fail completely before performing another operation. Applications that implement synchronous processes encounter bottlenecks for resources. These bottlenecks affect both the performance and the scalability of the application. One way to achieve scalability is to implement asynchronous processes. In applications that have asynchronous processes, long-running operations can be queued for completion later by a separate process.

SR-4: Design processes so that processes do not compete for resources - One of the biggest causes of scalability problems is competition for resources such as memory, processor cycles, bandwidth, or database connections. Plan your resource usage to minimize these problems:

- Sequence resource usage to use the most plentiful resources first and the least plentiful resources last.
- Acquire resources as late as possible. The shorter the amount of time a process uses a resource, the sooner the resource becomes available to another process.

SR-5: Design processes for commutability - Two or more operations are called commutative if they can execute in any order and still obtain the same result.

Typically, operations that do not involve transactions are commutative. For example, a busy e-commerce site that continuously updates the inventory of its diseases might experience contention for record locks. To prevent this, each inventory increment and decrement could become a record in a separate inventory transaction table. Periodically, the database could add the rows of this table for each disease and then update the disease records with the net change in inventory.

SR-6: Design components for interchangeability - An interchangeable component is designed to release its resources, move into a pool managed by a resource manager, and be re-initialized for use by a new client. Design the component so that no client-specific state persists from client to client. In addition, the component should support aggregation and not be bound to a specific thread. Resource pooling schemes such as COM+ component pooling and Open Database Connectivity (ODBC) connection pooling use interchangeable resources. For example, you can use the Component Services Administration tool to enable object pooling, set minimum and maximum pool size, and create timeout settings. For more information.

SR-7: Partition resources and activities - Minimize relationships between resources and activities by partitioning them. This helps you avoid the risk of bottlenecks. Partitioning activities can also ease the load on critical resources such as the processor and bandwidth. For example, using Secure Sockets Layer (SSL) to provide a secure connection results in significant overhead. Therefore, you might decide to use SSL only for pages that require a high level of security and use dedicated Web servers to handle SSL sessions. You can also partition resources and activities by creating many small components rather than a few large components, and by limiting cross-device communication. However, partitioning can make a system more complex. Dividing resources that have dependencies can add significant overheads to an operation.

5.5 Security Requirements

Malicious attackers use various methods to exploit system vulnerabilities to achieve their goals. Vulnerabilities are weak points or loopholes in security that an attacker exploits to gain access to an organization's network or to resources on the network. Some vulnerabilities, such as weak passwords, are not the result of application or software development design decisions. However, it is important for an organization to be aware of such security weaknesses to better protect its systems. Common vulnerabilities of applications include:

SR-1: Weak passwords - A weak password might give an attacker access not only to a computer, but to the entire network to which the computer is connected.

SR-2: Misconfigured software - Often the manner in which software is configured makes the system vulnerable. If services are configured to use the local system account or are given more permissions than required, attackers can exploit the services to gain access to the system and perform malicious actions on the system.

SR-3: Social engineering - A common form of discovering passwords that generally occurs when users are not aware of security issues and can be deceived into revealing their passwords. For example, an attacker posing as a help desk administrator might persuade a user to reveal his or her password under the pretext of performing an administrative task.

SR-4: Internet connections - The default installation of Internet Information Services (IIS) version 5.0 often enables more services and ports than are necessary for the operation of a specific application. These additional services and ports provide more opportunities for potential attacks. For example, modem connections bypass firewalls that protect networks from outside intruders. If an intruder can identify the modem telephone number and password, the intruder can connect to any computer on the network.

SR-5:Unencrypted data transfer - If the data sent between a server and the users is in clear text, there is a possibility that the data can be intercepted, read, and altered during transmission by an attacker.

SR-6: Buffer overrun - Malicious users probe applications looking for ways to trigger a buffer overrun because they can use a buffer overrun to cause an application or an operating system to crash. They can then find more security weaknesses by reading error messages.

SR-7: SQL injection - SQL injection occurs when developers dynamically build SQL statements by using user input. The attacker can modify the SQL statement and make it perform operations that were not intended.

SR-8: Secrets in code - Many security problems are created when a malicious user is able to find secrets that are embedded in code, such as passwords and encryption keys.

To design a secure Disease Symptoms Analysis application, we should be familiar with the following principles of security and employ them when creating security strategies:

SR-9: Rely on tested and proven security systems - Whenever possible, we should rely on tested and proven security systems rather than creating your own custom solution. Use industry-proven algorithms, techniques, platform-supplied infrastructure, and medically-tested and supported technologies. If we decide to develop a custom security infrastructure, validate our approach and techniques with expert auditing and security review organizations before and after implementing them.

SR-10: Never trust external input - We should validate all data that is entered by users or submitted by other services.

SR-11: Assume that external systems are not secure - If our application receives unencrypted sensitive data from an external system, assume that the information is compromised.

SR-12: Apply the principle of least privilege - Do not enable more attributes on service accounts than those minimally needed by the application. Access resources with accounts that have the minimal permissions required.

SR-13: Reduce available components and data - Risk will increase with the number of components and amount of data you have made available through the application, so you should make available only the functionality that you expect others to use.

SR-14: Default to a secure mode - Do not enable services, account rights, and technologies that you do not explicitly need. When we deploy the application on client or server computers, its default configuration should be secure.

SR-15: Do not rely on security by obscurity - Encrypting data implies having keys and a proven encryption algorithm. Secure data storage will prevent access under all circumstances. Mixing up strings, storing information in unexpected file paths, and so on, is not security.

SR-9: Follow STRIDE principles - Each letter in the STRIDE acronym specifies a different category of security threat: spoofing identity, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. These are classes of security vulnerabilities a system needs to protect itself against.

1.1.4 Security Features of .NET Technologies

.NET Web applications implement one or more of the logical services by using technologies such as Microsoft ASP.NET, Enterprise Services, XML Web services, remoting, Microsoft ADO.NET, and Microsoft SQL Server. To create effective security strategies, we need to understand how to fine-tune the various security features within each disease and technology area, and how to make them work together.

1.1.4.1 Authentication

Authentication is the process of discovering and verifying the identity of a user by examining the user's credentials and then validating those credentials against some authority. A variety of authentication mechanisms are used, some of which can be used with .NET Framework role-based security.

Examples of commonly used authentication mechanisms include the operating system, Passport, and application-defined mechanisms, such as NTLM and Kerberos version 5 authentication.

1.1.4.2 Authorization

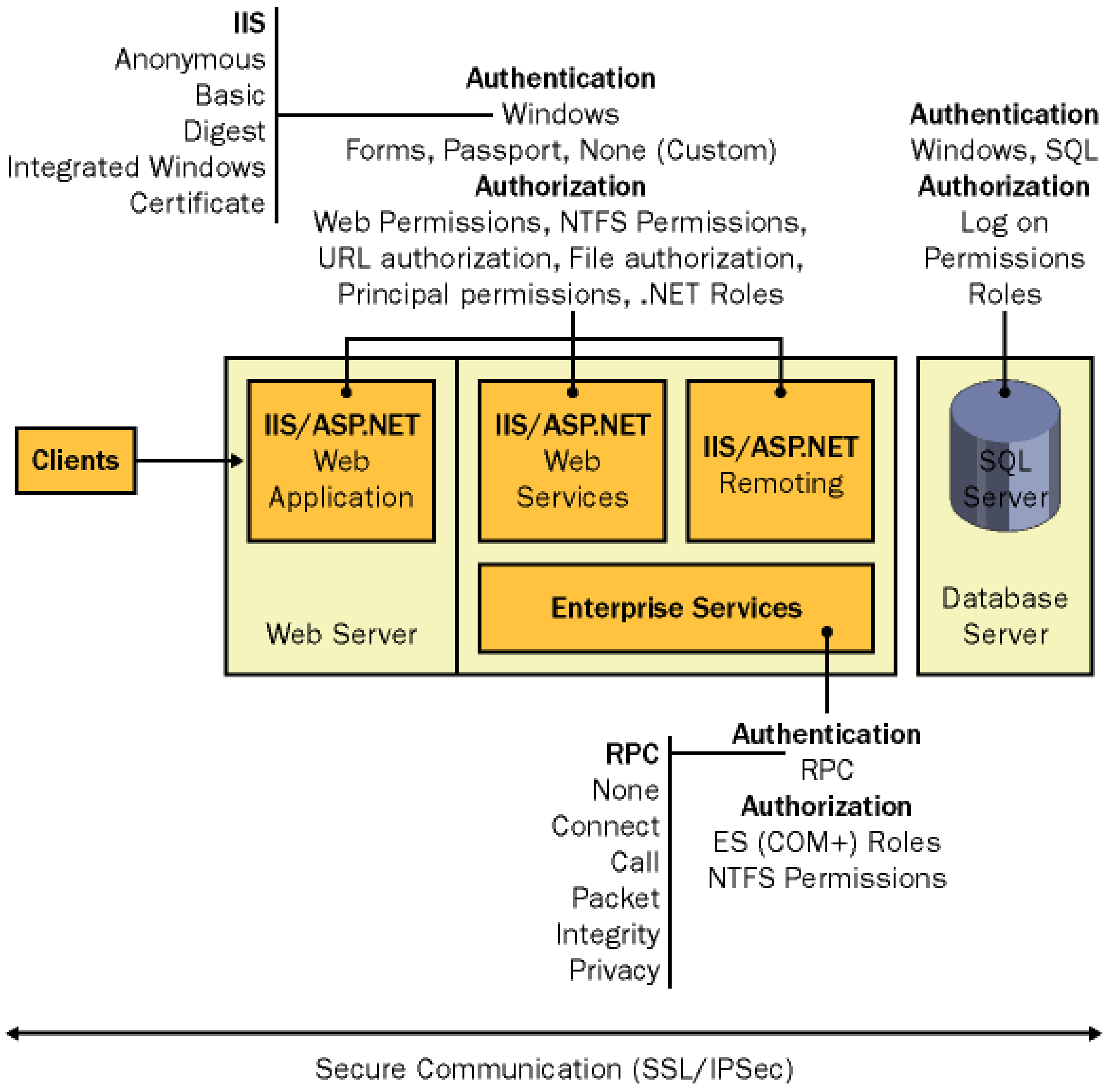
Authorization is the process of determining whether a user is allowed to perform a requested action. Authorization occurs after authentication and uses information about a user's identity and roles to determine what resources that user can access. You can use
.NET Framework role-based security to implement authorization.

1.1.4.3 ASP.NET security

ASP.NET provides a useful tool for application developers to use to create Web pages. When a Web site records a user's credit card information, the file or database that stores such information must be secured from public access. ASP.NET, in conjunction with IIS, can authenticate user credentials such as names and passwords

1.1.4.4 ADO.NET and SQL Server

ADO.NET provides data access services. It is designed for distributed Web applications, and supports disconnected scenarios. When we build Web-based applications, it is essential that we must use a secure approach to accessing and storing data. ADO.NET and SQL Server provide several security features that can be used to ensure secure data access.



[Security architecture]

5.6 Interoperability

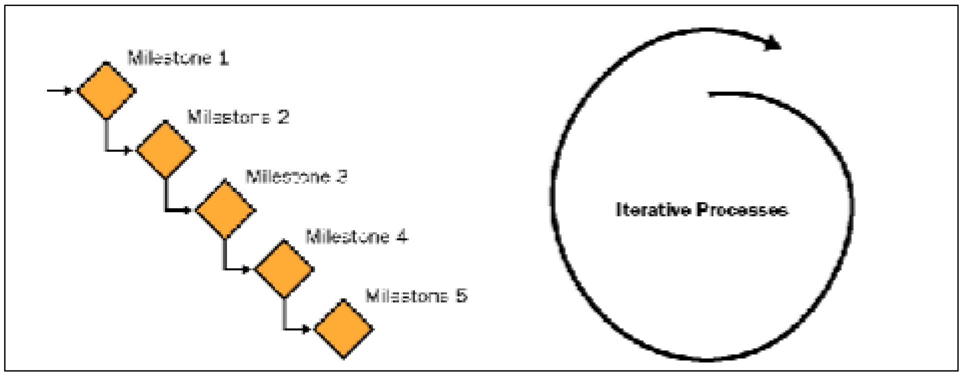
Typically, medium and large organizations have heterogeneous computing environments. For example, many organizations deploy distributed n-tier client/server applications that require access to data or transactions on existing systems. In addition, your application might need to interact with applications that have been developed using proprietary or third-party software. In Version 1.0 of the Disease Symptoms Analysis Application, there are no requirements for interoperability with other systems.

6. Project Management

6.1 Development Methodologies

What Are Process Models and development methodology?

A process model guides the order of project activities and represents the life cycle of a project. Historically, some process models are static and others do not allow checkpoints. Two such process models are the waterfall model and the spiral model.



[The waterfall model and the spiral model]

These models provide two different approaches to the project life cycle. The preceding illustration shows the waterfall model's cascading checkpoints and the spiral model's circular approach to process.

6.1.1 Waterfall model.

This model uses milestones as transition and assessment points. When using the waterfall model, you need to complete each set of tasks in one phase before moving on to the next phase. The waterfall model works best for projects in which the project requirements can be clearly defined and are not liable to modifications in the future. Because this model has fixed transition points between phases, you can easily monitor schedules and assign clear responsibilities and accountability.

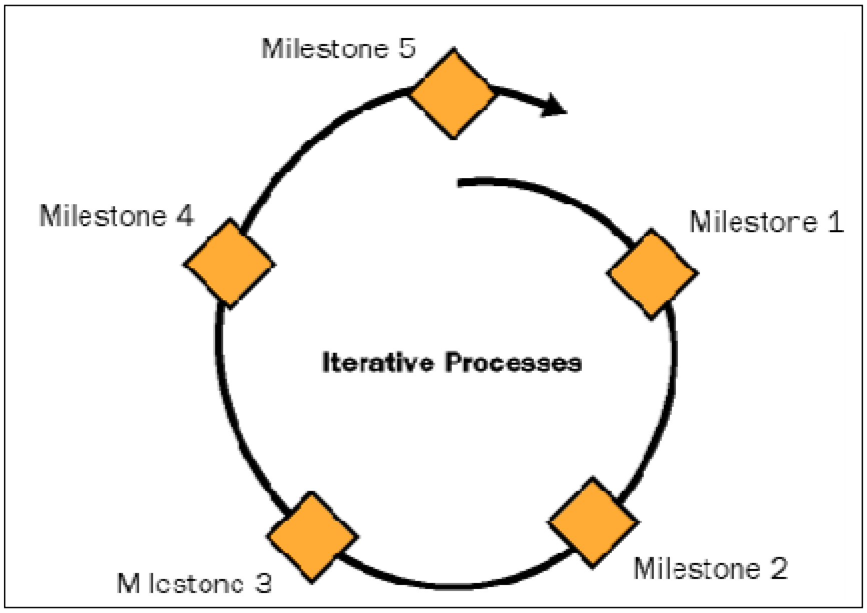
6.1.2 Spiral model.

This model is based on the continual need to refine the requirements and estimates for a project. The spiral model is effective when used for rapid application development of very small projects. This approach can generate great synergy between the development team and the Member because the Member is involved in all stages by providing feedback and approval. However, the spiral model does not incorporate clear checkpoints. Consequently, the development process might become chaotic.

6.1.3 MSF Process Model:

The MSF – Microsoft Solution Framework Process Model combines the best principles of the waterfall and spiral models. It combines the waterfall model's milestone-based planning and resulting predictability with the spiral model's benefits of feedback and creativity.

[MSF Process Model]



In Disease Symptoms Analysis devolvement process we have used MSF Process Model. This is a combination of spiral and waterfall process.

6.2 **Project Development Life Cycle (PDLC)**

The project will be completed by this four distinct phases as per MSF Process Model.

6.2.1 Envisioning

Envisioning is gathering the requirement of the project from different sources. Some techniques for gathering information are interviewing, shadowing, user instructions, and prototyping. Creating and identify the project scope. The scope of the project specifies what will and will not be included in the project. In this phase we have created the scope document. It includes information about the team and project structure, the problem statement, the vision statement, the scope of the project, the solution concept, user profiles, and project goals.

6.2.2 Planning

The planning phase results in the architecture and design of the solution, the plans to accomplish the development and deployment of the solution, and the schedules associated with tasks and resources. There are three design processes in the planning phase: conceptual, logical, and physical design.

6.2.3 Developing and Designing the System

Design of any application is not complete without a way for users to interact with the system. User interaction takes place through the application's presentation layer. The presentation layer is the part of the application that provides a communication mechanism between the user and the business service layer of the system. The most simple presentation layers contain user interface components, such as Windows Forms or ASP.NET Web Forms. For more complex user interactions, you can design user process components to orchestrate the user interface elements and control the user interaction.
User interface components display data to users, acquire and validate data from user input, and interpret user gestures that indicate the user wants to perform an operation on the data. Additionally, the user interface should filter the available actions to let users perform only the operations that are appropriate at a certain point in time.

6.2.4 Stabilizing

The purpose of the stabilizing phase is to reduce the risks of releasing the solution to diseaseion. A successful stabilizing phase requires that the team make the transition from a mindset focused on building features to one focused on getting the solution to a known state of quality. Deliverables of the deploying phase are operations and support information systems, repository of all versions of documentation and code, and project closeout reports.

7. Estimation and Planning

7.1 Planning

The MSF – Microsoft Solution Framework Process Model describes a generalized sequence of activities for building and deploying enterprise solutions. This process is flexible and can accommodate the design and development of a broad range of enterprise projects. The MSF Process Model is a phase-based, milestone-driven, and iterative model that can be applied to developing and deploying traditional applications, enterprise solutions for Disease Symptoms Analysis , and Web-distributed applications.

MSF guidance includes disciplines for managing the people, processes, and technology elements that most projects encounter. The three key MSF disciplines are risk management, readiness management, and project management.

7.1 Application of the MSF model in our project:

7.1.1 Envisioning process:

Each phase in the MSF Process Model has interim milestones and a major milestone. Interim milestones are associated with the various activities that are performed in a phase, such as creating a team and creating a vision/scope document. The major milestone indicates that the team can progress to the next phase in the MSF Process Model. For example, the major milestone of the envisioning phase is the vision/scope approved milestone. When the team reaches this milestone, the team can progress to the planning phase of the MSF Process Model. The team creates deliverables for each task in the envisioning phase. Together, these deliverables provide context and direction for the team for the remainder of the project, and communicate the project vision and scope to the Member. The deliverables that the team creates during the envisioning phase include:

- Vision/scope - Problem statements and business objectives, A review of the existing processes, A broad definition of user requirements.
- Project structure - A project structure and process standards for the team to follow
- Risk assessment - A preliminary risk assessment, Plans for mitigating or eliminating the identified risks

7.1.2 **Planning process:**

During the planning phase, the team determines what to develop and plans how to create the solution. The team prepares the functional specification, creates a design of the solution, and prepares work plans, cost estimates, and schedules for the various deliverables.

The planning phase involves the analysis of requirements. These requirements can be categorized as business requirements, user requirements, operational requirements, and system requirements. These requirements are used to design the solution and its features and to validate the correctness of the design.

After gathering and analyzing the requirements, the team creates the design of the solution. The team creates user profiles that specify the various users of the solution and their roles and responsibilities. The team then creates a series of usage scenarios. A usage scenario specifies the activity performed by a particular type of user. Therefore, the team needs to create usage scenarios for all user profiles. After creating usage scenarios, the team creates use cases for the usage scenarios. A use case specifies the sequence of steps that a user will perform in a usage scenario.

The planning phase deliverables provide the basis for making future tradeoff decisions. The following deliverables are produced during the planning phase:

- Functional specification
- Risk management plan
- Master project plan and master project schedule

7.1.3 Design Process:

The three design stages are:

- Conceptual design, in which you view the problem from the perspective of the users and business requirements and define the problem and solution in terms of usage scenarios.
- Logical design, in which you view the solution from the perspective of the project team and define the solution as a set of services.
- Physical design, in which you view the solution from the perspective of the developers and define the technologies, component interfaces, and services of the solution.

You document the solution design in the functional specification. The functional specification describes the behavior and appearance of each feature of the solution. It also describes the architecture and the design for all features.

7. 1.4 Development process:

During the developing phase, the project team creates the solution. This process includes creating the code that implements the solution and documenting the code. In addition to developing code, the team also develops the infrastructure for the solution.

The deliverables of the developing phase include:

- Source code and executable files
- Installation scripts and configuration settings for deployment
- Finalized functional specification
- Performance support elements
- Test specifications and test cases

7. 1.5 Stabilization process:

During the stabilizing phase, the team performs integration, load, and beta testing on the solution. In addition, the team tests the deployment scenarios for the solution. The team focuses on identifying, prioritizing, and resolving issues so that the solution can be prepared for release. During this phase, the solution progresses from the state of all features being complete as defined in the functional specification for this version to the state of meeting the defined quality levels. In addition, the solution is ready for deployment to the business

The deliverables of the stabilizing phase are as follows:

- Final release
- Release notes
- Performance support elements
- Test results and testing tools
- Source code and executable files
- Project documents
- Milestone review

7. 1.6 Deployment process:

During this phase, the team deploys the solution technology and site components, stabilizes the deployment, transfers the project to operations and support, and obtains final Member approval of the project. After deployment, the team conducts a project review and a Member satisfaction survey. The deploying phase culminates in the deployment complete milestone.

The deliverables of the deploying phase are as follows:

- Operation and support information systems

    - Procedures and processes
    - Knowledge base, reports, and logbooks
- Documentation repository for all versions of documents and code developed during the project
- A training plan
- Project completion report
    - Final versions of all project documents
    - Member satisfaction data
    - Definition of next steps

1.2 6.2 Estimation

1.2.1 6.2. 1 Basic COCOMO

The Constructive Cost Model (COCOMO) is an algorithmic Software Cost Estimation Model developed by Barry Boehm. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics. Constructive Cost Model: It is a hierarchy of estimation models that address: Application composition model: Used during the early stage of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount..

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes

The basic COCOMO equations take the form

Effort Applied = $a_b(\text{KLOC})^{b_b}$ [man-months]

Development Time = $c_b(\text{Effort Applied})^{d_b}$ [months]

People required = Effort Applied / Development Time [count] The

coefficients a_b, b_b, c_b and d_b are given in the following table.

Software Project	*a*	*b*	*c*	*d*
Organic	*2.4*	*1.05*	*2.5*	*0.38*
Semi – Detached	*3.0*	*1.12*	*2.5*	*0.35*
Embedded	*3.6*	*1.2*	*2.5*	*0.32*

E =3.0(KLOC)1.12
= 3.0(5000)^1.12
= 18 person months

D = 2.5E0.35
= 2.5(22)0. 35
= 6.8 months

Therefore No. of persons required =18/6.8=3persons.

6.2.2 Function Point Estimation

A function point is a unit of measurement to express the amount of business functionality an information system provides to a user. Function points are the units of measure used by the IFPUG Functional Size Measurement Method. The IFPUG FSM Method is an ISO recognised software metric to size an information system based on the functionality that is perceived by the user of the information system, independent of the technology used to implement the information system.

The method of measuring the size of an information system and expressing it in a number of function points is called function point analysis (FPA). The method is kept up to date by worldwide cooperating FPA user groups like NESMA and IFPUG. A function point analysis expresses the functional size of an information system in a number of function points (for example: the size of a system is 314 FPs). There are many uses and benefits of function points and the functional size may be used as input into many types of project and organization decisions including determining the:

- Budget for application development or enhancement costs.
- Budget for the annual maintenance costs of the application portfolio.
- Project diseaseivity after completion of the project.
- Software Size for cost estimating.

Function-Oriented Metrics

1.2.1.1 FP = count total * [0.65 + 0.01 * sum of F_i]

1. Does the system require reliable backup and recovery=5

2. Are data communications required=4

3. Are there distributed processing functions=2

4. Is performance critical=5

5. Will the system run an existing, heavily utilized operational environment=5

6. Does system requires online data entry=5

7. Does online data entry req. input transaction to be build on multiple screens=3

8. Are master files updated online =4

9. Are I/ps , 0/ps, files and inquires complex=3 10.Is

essential processing complex=5

11. Is code reusable=4

12.Are conversion and installation included in design=2 13.Is

system supports multiple installations =2

14. Is application designed to facilitate change and ease of use by user=5 sum

of **F_i = 54**

Measurement Parameter	Count		Simple	Average	Complex		
Number of User Inputs	12	*	3	4	6	=	36
Number of User outputs	4	*	4	5	7	=	16
Number of User inquires	1	*	3	4	6	=	6
Number of files	5	*	7	10	15	=	35
Number of External interface	3	*	5	7	10	=	15
Count Total							108

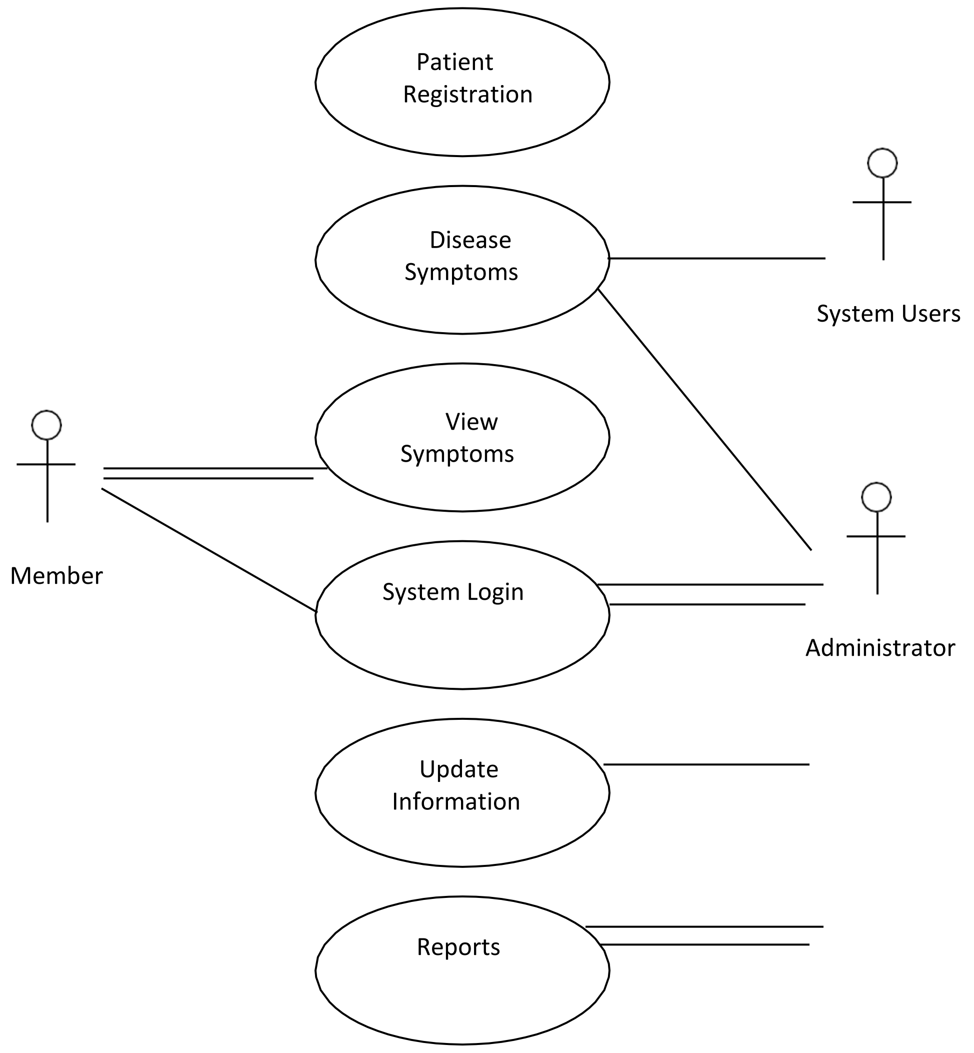
FP = Count total * [0.65 + 0.01 * sum of F*i*]

FP=108*[0.65 + 0.01 *
54] FP=129

7. Preliminary Design

7.1 **Use Case**

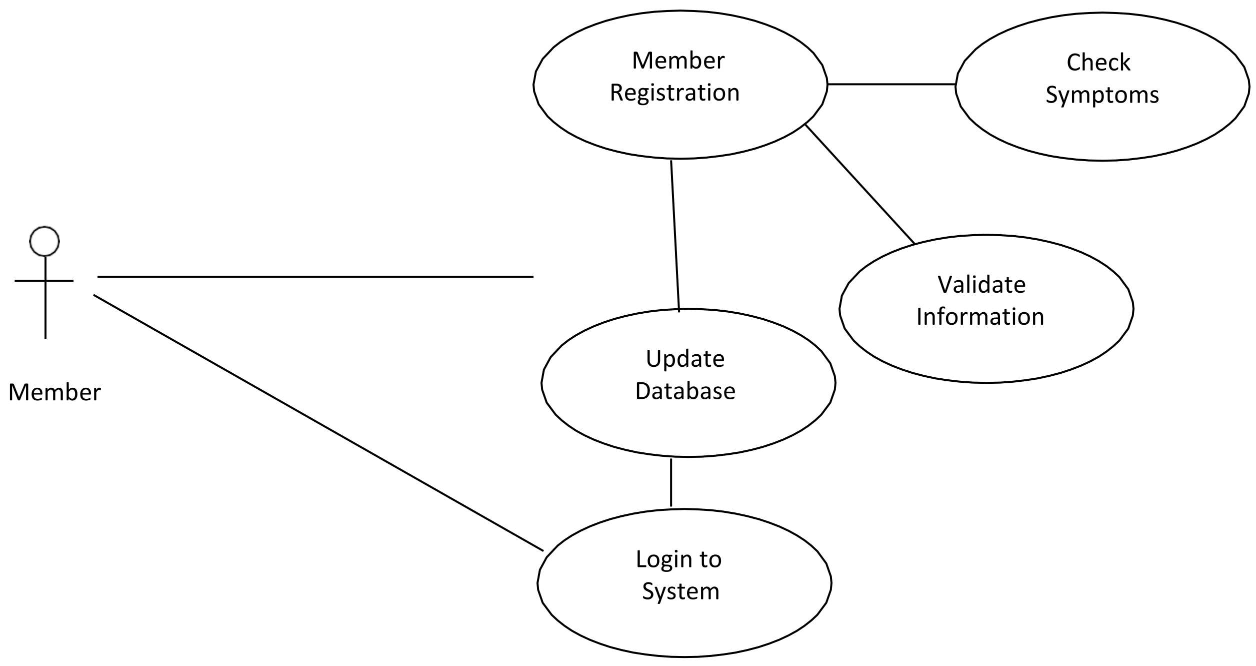
Disease Symptoms Analysis Application Version 1.0 will address the following use cases. The complete usage scenarios will be completed during the information-gathering process. Use cases will be created and prioritized. Selected use cases will be expanded into usage scenarios and features that are derived from both use cases and the usage scenarios, as represented in the following diagram:



[Disease Symptoms Analysis] Usage Scenario – This usage scenario, or scenario for short, describes a real- world example of how one or more people or organizations interact with Disease Symptoms Analysis

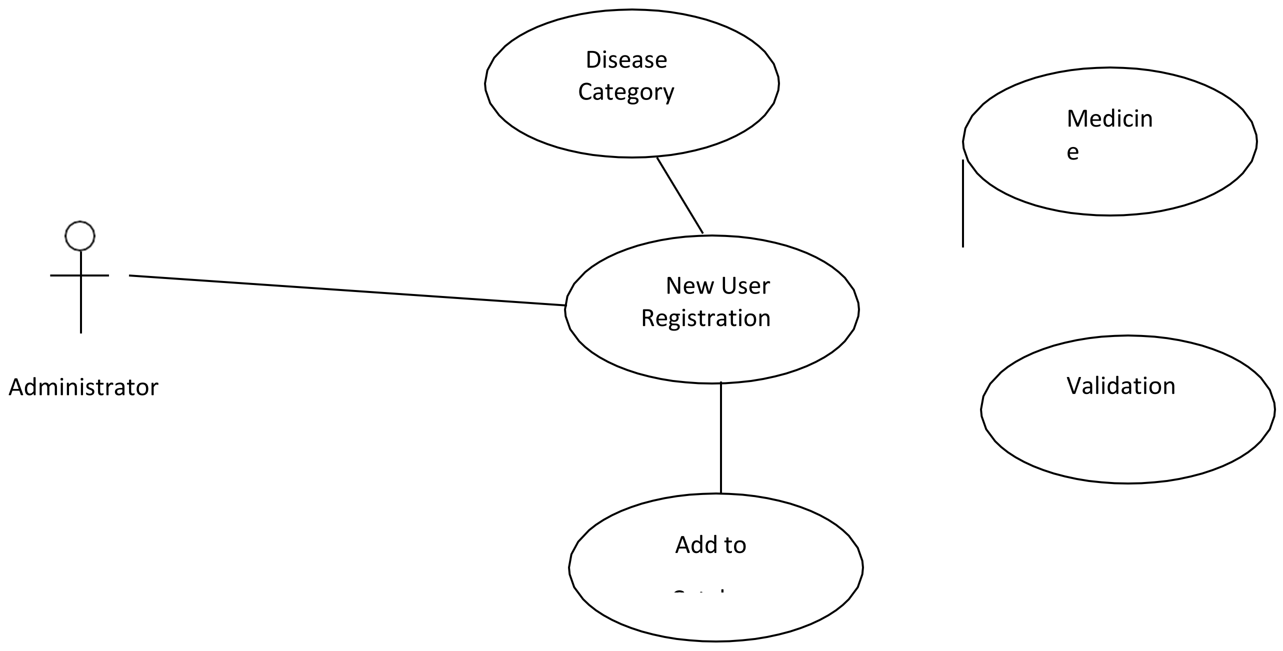
system. It describes the steps, events, and/or actions which occur during the interaction. This Usage scenarios indicating exactly how someone works with the user interface, or reasonably high level describing the critical business actions but not the indicating how they're performed.
]

1.2.1.2 Use Case Diagram : Member Registration



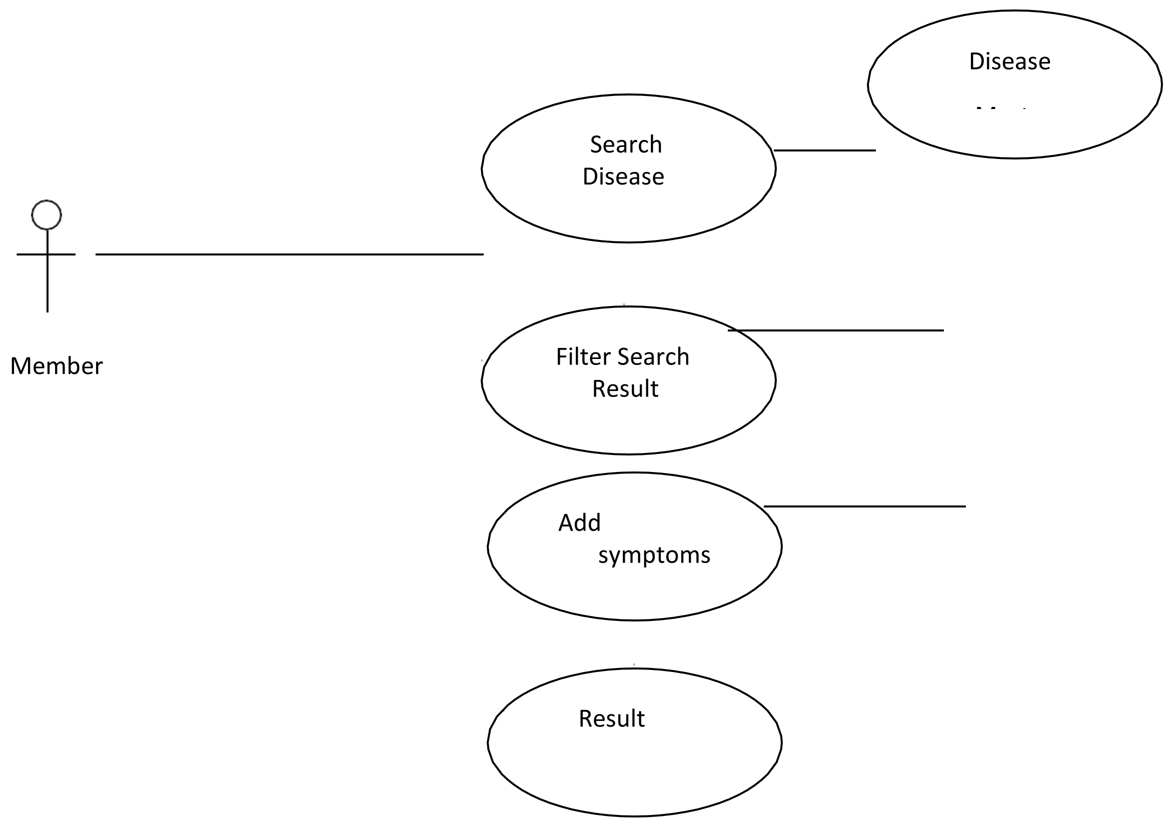
[Use Case: Member Registration – This use case scenario, or scenario for short, describes how Member will registered into Disease Symptoms Analysis System. It describes the steps, events, and/or actions which occur during the interaction. This Usage scenarios indicating exactly how someone works with the Member registration interface.]

1.2.1.3 Use Case Diagram: Disease Registration



[Use Case: Disease Registration – This use case scenario, or scenario for short, describes how administrator will add new disease details into Disease Symptoms Analysis System. It describes the steps, events, and/or actions which occur during the interaction.

1.2.1.4 Use Case Diagram: Sales Registration

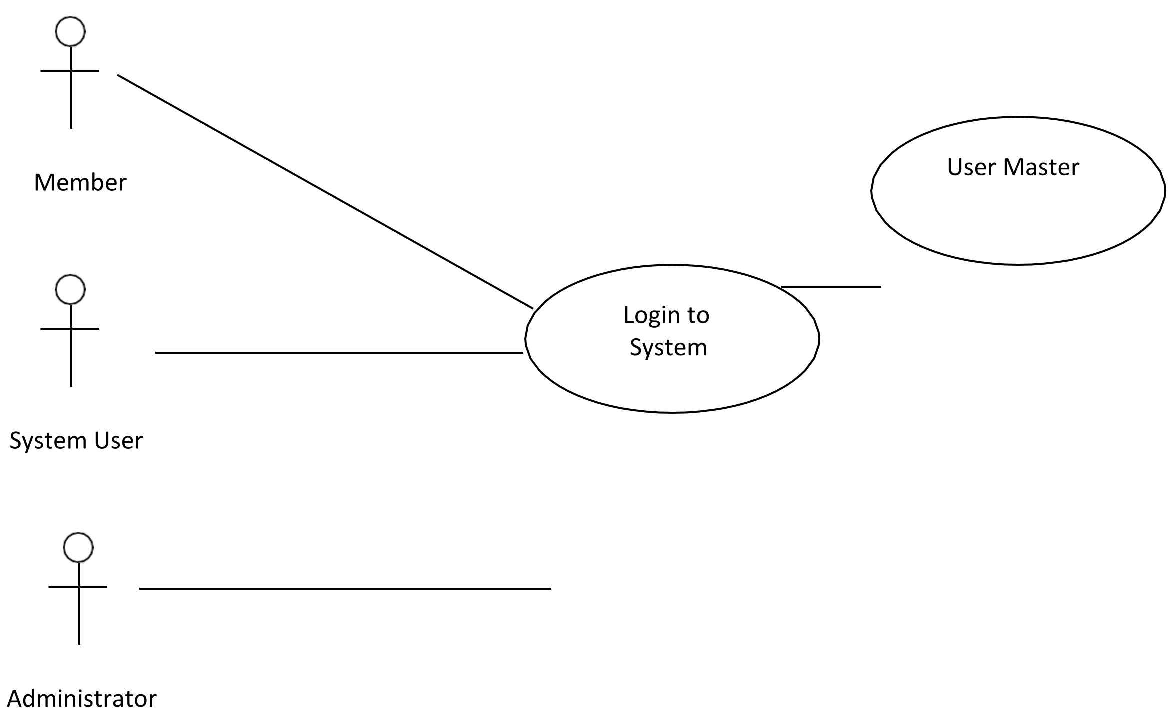


[Use Case: User Registration – This use case scenario, or scenario for short, describes how Member will search disease, add disease to shopping cart and make checkout from Disease Symptoms Analysis System. It describes the steps, events, and/or actions which occur during the interaction.]

7.1.1 Use Case. System Login

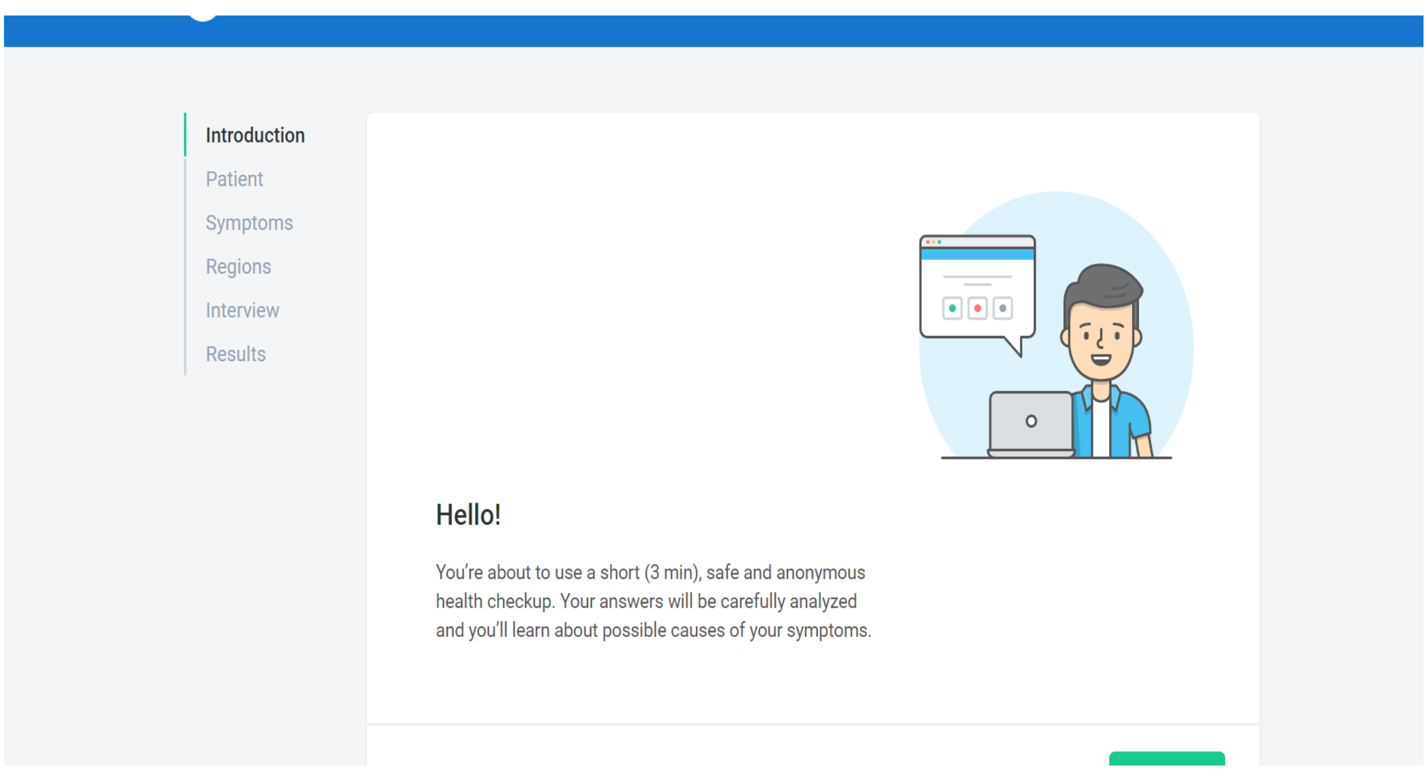
System Login	
Element	**Details**
Actor	Member, System User, Administrator
Trigger	The user wish to start using the system.
Pre Conditions	The user is not logged into the system.
Post Conditions	The user is logged into the system, and the system menu is displayed.
Normal course	1. The user click the link for the Disease Symptoms Analysis application and a login page appear on the screen. 2. The user types his username and password into the form and press the login button. 3. The system confirms that the user is logged on.
Alternative courses	2a. The user is not a valid user or the user name or the password is mistyped. 2a1. The system reject login with an error message that express wrong login name or password.

1.2.1.5 Use Case Diagram: System Login



[Use Case: System Login – This use case scenario, or scenario for short, describes how actors will perform login Disease Symptoms Analysis System. It describes the steps, events, and/or actions which occur during the interaction.

8 CODING AND WEB PAGES



Hello!

You're about to use a short (3 min), safe and anonymous health checkup. Your answers will be carefully analyzed and you'll learn about possible causes of your symptoms.

```
<!DOCTYPE html>

<html lang="en"

><head><title>Disease Symptoms Analysis – Check your symptoms online</title>

<meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="author" content="Infermedica">

<meta name="description" content="">

<meta name="keywords" content="">

<meta name="viewport" content="width=device-width, initial-scale=1,

user-scalable=no, minimum-scale=1, maximum-scale=1">

<link rel="apple-touch-icon" href="/static/healthform/img/apple-touch-icon.png">

<base href="/diagnosis/en/"><style type="text/css">

.loader{box-sizing:border-box;width:12px;height:12px;border-radius:50%;border:2px

   solid rgba(255,255,255,0.35);border-top-color:#fff;-

webkit-animation:rotate 0.8s cubic-bezier(0.8,0.4,0.4,0.8)

infinite;animation:rotate

0.8s

 cubic-bezier(0.8,0.4,0.4,0.8) infinite}.loader.medium{width:32px;height:32px}.

loader.big{width:64px;height:64px}.

loader.gray{border-color:rgba(148,164,179,0.3);

border-top-color:#94A4B3}@-webkit-keyframes

rotate{100%{-webkit-transform:rotate(360deg);

transform:rotate(360deg)}}@keyframes

rotate{100%{-webkit-transform:rotate(360deg)

;transform:rotate(360deg)}}

#initialization-loader{display:block;position:fixed;

top:0;right:0;bottom:0;left:0;z-index:10000;

background-color:#1576D1;font-family:'Roboto',

sans-serif;font-size:15px;line-height:1.4675;
-webkit-font-smoothing:antialiased;

-moz-osx-font-smoothing:grayscale;text-rendering:optimizeLegibility}

#initialization-loader .wrapper{margin:0

auto;

min-width:200px;position:absolute;top:50%;

left:50%;-webkit-transform:translate(-50%,-50%);

-ms-transform:translate(-50%,-50%);

-o-transform:translate(-50%,-50%);

transform:translate(-50%,-50%)}

#initialization-

loader .loader{margin:0 auto}

#initialization-loader

.font-prefetch{position:absolute;top:-999999px;

visibility:hidden;height:0;

overflow:hidden}

#initialization-

loader .font-prefetch

.span-normal

{font-weight:400}

#initialization-loader

.font-prefetch .span-bold

{font-weight:500}#initialization-loader

.font-prefetch .span-bolder

{font-weight:700

}#initialization-loader
```

```
p{margin:14px;color:

#fff;text-align:center;

font-family:sans-serif}

#initialization-loader.ready{display:none}

#initialization-loader.ready-add

{display:block;opacity:1;

-webkit-transition:opacity

0.3s ease-out;

-o-transition:opacity 0.3s

 ease-out;

transition:opacity 0.3s ease-out}

#initialization-loader.

ready-add-active{opacity:0}</style><script

 async

src="https://www.googletagmanager.com/gtag/js">

</script><script type="text/javascript">window.dataLayer=window.dataLayer||[];

function gtag(){dataLayer.push(arguments);}

gtag('js',new Date());gtag('config','UA-38774656-1',

{'send_page_view':false,'anonymize_ip':true});gtag('config','AW-

1001570007',{'anonymize_ip':true});(function(f,b){if(!b.__SV){var

a,e,i,g;window.mixpanel=b;b._i=[];b.init=function(a,e,d){function

 f(b,h){var a=h.split(".");2==a.length&&(b=b[a[0]],h=a[1]);

b[h]=function(){b.push([h].

concat(Array.prototype.slice.call(arguments,0)))}}var

c=b;"undefined"!==typeof d?c=b[d]=[]:
```

```
d="mixpanel";c.people=c.people||[];

c.toString=function(b){var a="mixpanel";"mixpanel"!==d&&(a+="."+d);b|

|(a+=" (stub)");return a};c.people.toString=function()

{return c.toString(1)+".people (stub)"};

i="disable track track_pageview track_links track_forms register

register_once alias unregister

identify name_tag set_config

people.set people.set_once people.increment

people.append people.union

people.track_charge people.clear_charges

people.delete_user".split(" ");

for(g=0;g<i.length;g++)f(c,i[g]);b._i.push([a,e,d])};b.__SV=1.2;

a=f.createElement("script");a.type="text/javascript";

a.async=!0;a.src="undefined"!==typeof MIXPANEL_CUSTOM_LIB_URL?

MIXPANEL_CUSTOM_LIB_URL:"//cdn.mxpnl.com/libs/mixpanel-2-

latest.min.js";e=f.getElementsByTagName("script")[0];

e.parentNode.insertBefore(a,e)}})

(document,window.mixpanel|

|[]);mixpanel.init("5222bd0e135718d557f7191b588d3167",

{secure_cookie:true,cross_subdomain_cookie:

false,track_pageview:false});(function(e,t){var n=e.amplitude||{_q:[],_iq:{}};var

r=t.createElement("script");r.type="text/javascript";r.integrity="sha384-

a+mq7tiLwde/00Oc7avFHLn/ttGfdAq1rtZc7u97SEzIiyYoT2IsOKWCkAThwdEu";

r.crossOrigin="anonymous";

r.async=true;r.src=https://cdn.amplitude.com/libs/amplitude-5.3.0-min.gz.js
```

```
;r.onload=function(){if

(!e.amplitude.runQueuedFunctions)

{console.log("[Amplitude]

 Error: could not load SDK")}};

var

i=t.getElementsByTagName("script")[0];

i.parentNode.insertBefore(r,i);

function s(e,t){e.prototype[t]=function(){this._q.push([t].

concat(Array.prototype.slice.call(arguments,0)))

;return this}}var o=function(){this._q=[];

return this};

var

a=["add","append","clearAll","prepend","set","setOnce","unset"];

for(var u=0;u<a.length;u+

+){s(o,a[u])}n.Identify=o;var c=function(){this._q=[];return

this};

varl=["setDiseaseId","setQuantity","setPrice","setRevenueType",

"setEventProperties"];for(var

p=0;p<l.length;p++){s(c,l[p])}n.Revenue=c;var

 d=["init","logEvent","logRevenue","setUserId","

setUserProperties",

"setOptOut","setVersionName","setDomain",

"setDeviceId","setGlobalUserProperties",

"identify","clearUserProperties",

"setGroup","logRevenueV2","
regenerateDeviceId","groupIdentify","

onInit","logEventWithTimestamp",

"logEventWithGroups","setSessionId","resetSessionId"];function

v(e){function t(t){e[t]=function()

{e._q.push([t].concat(Array.prototype.slice.call(arguments,0)))}}

for(var n=0;n<d.length;n++){t(d[n])}}v(n);

n.getInstance=function(e){e=(!e||e.length===0?

"$default_instance":e)

.toLowerCase();if(!n._iq.hasOwnProperty(e)){n._iq[e]={_q:[]};v(n._iq[e])}return

n._iq[e]};e.amplitude=n})(window,document);amplitude.getInstance().

init("d697e5b84bf262ccc1814c7eb689bc4b",

null,{forceHttps:true,domain:'symptomate.com',

trackingOptions:

{ip_address:false}});!function(f,b,e,v,n,t,s)

{if(f.fbq)return;n=f.fbq=function()

{n.callMethod?n.callMethod.apply(n,arguments):

n.queue.push(arguments)};

if(!f._fbq)f._fbq=n;n.push=n;n.loaded=!0;

n.version='2.0';n.queue=[];t=b.createElement(e);t.async=!0;

t.src=v;s=b.getElementsByTagName(e)[0];s.parentNode.insertBefore(t,s)}

(window,document,'script','https://connect.facebook.net/en_US/fbevents.js');

fbq('init','1.955795021178967e+15');fbq('track','PageView');fbq('track','StartTrial');

</script><noscript><img height="1" width="1" style="display:none"

src="https://www.facebook.com/tr?id=1.955795021178967e+15&ev=PageView&noscript=1"/>

</noscript></head><body ng-app="hf.prescreening"
```

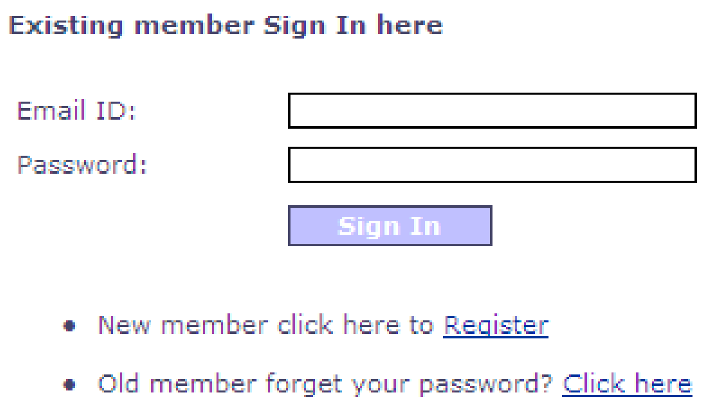
```
ng-init="formToken='form_en';interviewId='9c8d0bf2-f6ad-4b5b-9797-b67c840e32d7';

questionNodes={};"><div id="initialization-loader"><div class="wrapper"

><div class="loader big"></div><p>Loading...</p>

<div class="font-prefetch"><span class="span-normal">Abc</span>

<span class="span-bold">Abc</span>

<span class="span-bolder">Abc</span></div></div></div><div

id="content-wrapper" ng-controller="PrescreeningController"

ng-class="{'content-wrapper-fluid': form.settings.fluid_layout}"

><header class="prescreening-header" ng-if="showHeader()">


ng-if="form.settings.side_menu"><side-menu questions="form.questions"

 current-screen="controls.currentScreen"

progress="controls.progressBar.progress"

on-block-click="navigation.goToScreen(position)">

</side-menu></div><div class="form "

 ng-class="{'has-side-menu': form.settings.side_menu}"

 ng-cloak><div class="form-content invisible" ng-class="{'has-footer':

 controls.showFooter, 'back': controls.directionBack}"

ng-style="controls.fromContentStyle">

<div class="screen-container"

ng-repeat="question in form.questions | orderBy: 'position'"

ng-show="question.position == controls.currentScreen"

screen-id="{{ question.position }}"

screen-type="{{ question.type }}"></div></div>

<div class="form-footer hidden-print" ng-show="controls.showFooter"
```

```
ng-class="{'has-side-menu': form.settings.side_menu}"

ng-cloak><div class="footer-progress"

ng-style="{ width: controls.progressBar.progress + '%' }">

</div><div class="footer-content"><div class="status ng-cloak">

<button class="btn btn-link btn-back-footer hidden-print ng-cloak"

ng-class="{ 'hidden' : !controls.showPrevious }"

ng-disabled="controls.animationInProgress"

ng-click="controls.animationInProgress

|| navigation.previousScreen()">

Back

</button><button class="btn btn-primary btn-next hidden-print"

ng-class="{ 'hidden': !controls.showNext, 'disabled

': controls.animationInProgress

 || !controls.isScreenValid() }"

ng-click="controls.animationInProgress |

| navigation.nextScreen()">

{{controls.nextButtonLabel || 'Next'}}

</button><span class="choose-answer-above text-muted hidden-xs"

 ng-class="{ 'hidden': !controls.showClickAboveHint }">

Select an answer above

</span><a href="/"

class="btn btn-primary btn-finish hidden-print" rel="noopener"

ng-class="{ 'hidden' : controls.animationInProgress

 || !controls.prescreeningFinish.success }"
```

```
ng-disabled="controls.animationInProgress">

Finish

</a><button class="btn btn-primary btn-retry hidden-print"

ng-click="navigation.resubmitPrescreeningForm()"

ng-show="controls.prescreeningFinish.error">

Retry

</button></div>

</div></div>

</div><div

 class="footer-links"><span>&copy; 2020&nbsp;</span><a

 href="http://infermedica.com/" target="_blank" rel="noopener">Infermedica</a>

<a ng-if="form.settings.for_business"

 ng-click="showForBusiness()" href="" target="_blank" rel="noopener">

For business</a><a ng-if="form.settings.ce_marking"

 ng-click="showCEMarking()" href=""

target="_blank" rel="noopener">CE Marking</a><a

 ng-if="form.terms_of_service.length"

 ng-click="showTermsOfService()"

href="" target="_blank" rel="noopener">

Terms of Service</a><a ng-if="form.privacy_policy.length"

ng-click="showPrivacyPolicy()"

href="" target="_blank" rel="noopener">Privacy Policy</a>

<a ng-repeat="panel in form.panels" ng-if="panel.id != 'pp' && panel.id != 'tos'"

ng-click="openSidePanel(panel.title, panel.content)"
```

```
 href>{{ ::panel.title }}</a><a ng-repeat="link in form.settings.footer_links"

ng-href="{{ ::link.url }}" ng-attr-title="{{ ::link.title }}"

target="_blank" rel="noopener">{{ ::link.title }}</a></div>

</div><side-panel panel-config="sidePanelConfig"

 form="form" go-to-language="goToLanguage"

open="sidePanelOpen" close-func="closeSidePanel">

</side-panel></div><noscript

 id="prescreening-styles">

<link href=https://fonts.googleapis.com/css?family=Roboto:400,500&subset=latin-ext

 rel="stylesheet">

<link rel="stylesheet" href="/static/CACHE/css/287097e71d92.css" type="text/css" /></noscript><script

type="text/javascript">

(function(){window.TOUCH_SCREEN=!!

('ontouchstart'in window||navigator.maxTouchPoints);var

 bodyClass=window.TOUCH_SCREEN?'touch':'no-touch';if(window.document.body.classList)

{window.document.body.classList.add(bodyClass);}

else{window.document.body.className+=' '+bodyClass;}

window.API_ALIAS='en';window.INTERFACE_LANG='en';})();/*! loadCSS.

 [c]2017 Filament Group, Inc. MIT License */(function(w

){"use strict";var loadCSS=function(href,before,media){var

doc=w.document;var ss=doc.createElement("link");var ref;if(before){ref=before;}

else{var refs=(doc.body||

doc.getElementsByTagName("head")[0]).childNodes;ref=refs[refs.length-1];}

var sheets=doc.styleSheets;ss.rel="stylesheet";

ss.href=href;ss.media="only x";function ready(cb){if(doc.body){return cb();}
setTimeout(function(){ready(cb);});}

ready(function()

{ref.parentNode.insertBefore(ss,(before?ref:ref.nextSibling));})

;var onloadcssdefined=function(cb)

{var resolvedHref=ss.href;var i=sheets.length;while(i--)

{if(sheets[i].href===resolvedHref){return cb();}}

setTimeout(function()

{onloadcssdefined(cb);});};

function loadCB(){if(ss.addEventListener)

{ss.removeEventListener("load",loadCB);}

ss.media=media||"all";}

if(ss.addEventListener){ss.addEventListener("load",loadCB);}

ss.onloadcssdefined=onloadcssdefined;onloadcssdefined(loadCB);return

 ss;};if(typeof exports!=="undefined"){exports.loadCSS=loadCSS;}

else{w.loadCSS=loadCSS;}}(typeof

global!=="undefined"?global:this));(function()

{var tmpDiv=document.createElement('div');tmpDiv.

innerHTML=document.getElementById('prescreening-styles').

textContent;tmpDiv.hasChildNodes()&&[].

forEach.call(tmpDiv.childNodes,function(child)

{child.tagName==='LINK'&&loadCSS(child.href);});})();</script><script

 src="/jslang/en/"></script><script

src="/jsreverse/"></script><script

type="text/javascript"
```

```
src="/static/CACHE/js/b2db562af246.js">
</script><script type="text/javascript">
Raven.config('https://ae60dc8d358243ea81c0a719f3137394@sentry.io/121824',
{debug:false,environment:'diseaseion',release:'hf-sym@7daefb1',
tags:{instance:window.location.hostname,event_layer:'frontend'}})
.addPlugin(Raven.Plugins.Angular).install();
</script><script
src="/prescreening/jstemplates/en/">
</script>
<scripttype="text/javascript">gtag('event','
conversion',{'send_to':'AW-1001570007/5-EYCLiq8aQBENf9yt0D'});</script><script
type="text/javascript" src="/static/CACHE/js/f837a52ac718.js"></script></body></html>
```


Existing member Sign In here

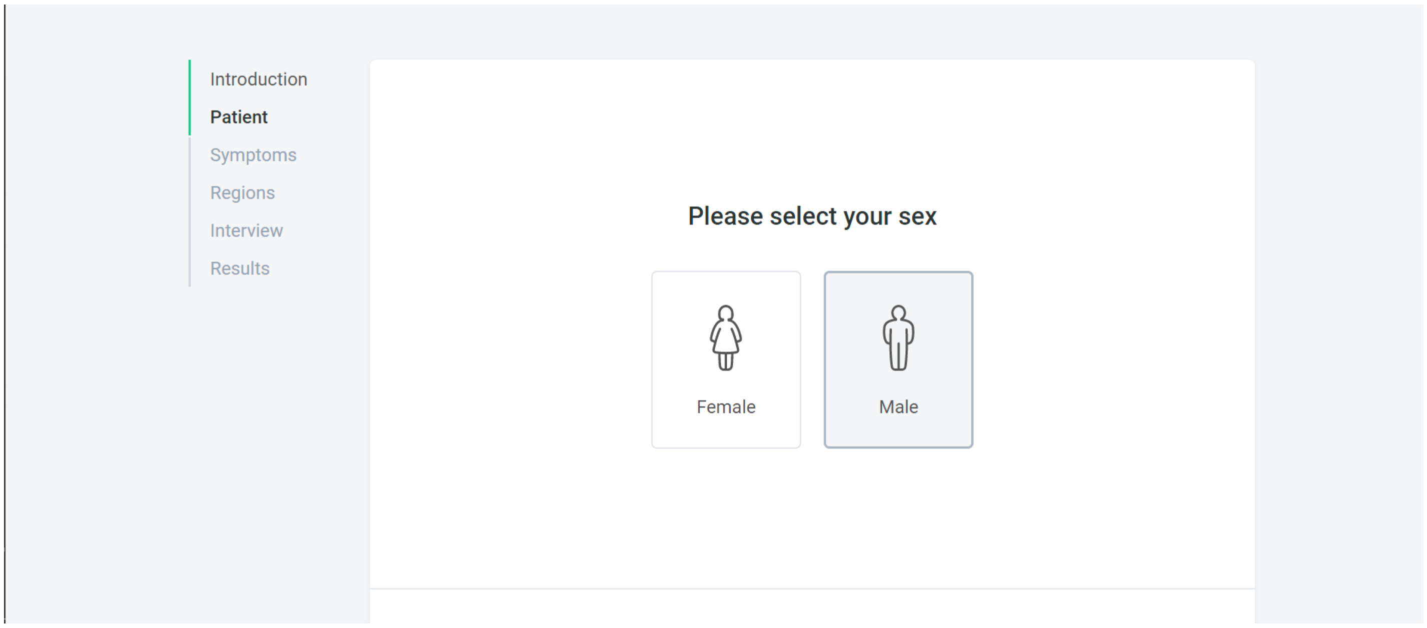
Email ID:	
Password:	

- New member click here to Register
- Old member forget your password? Click here

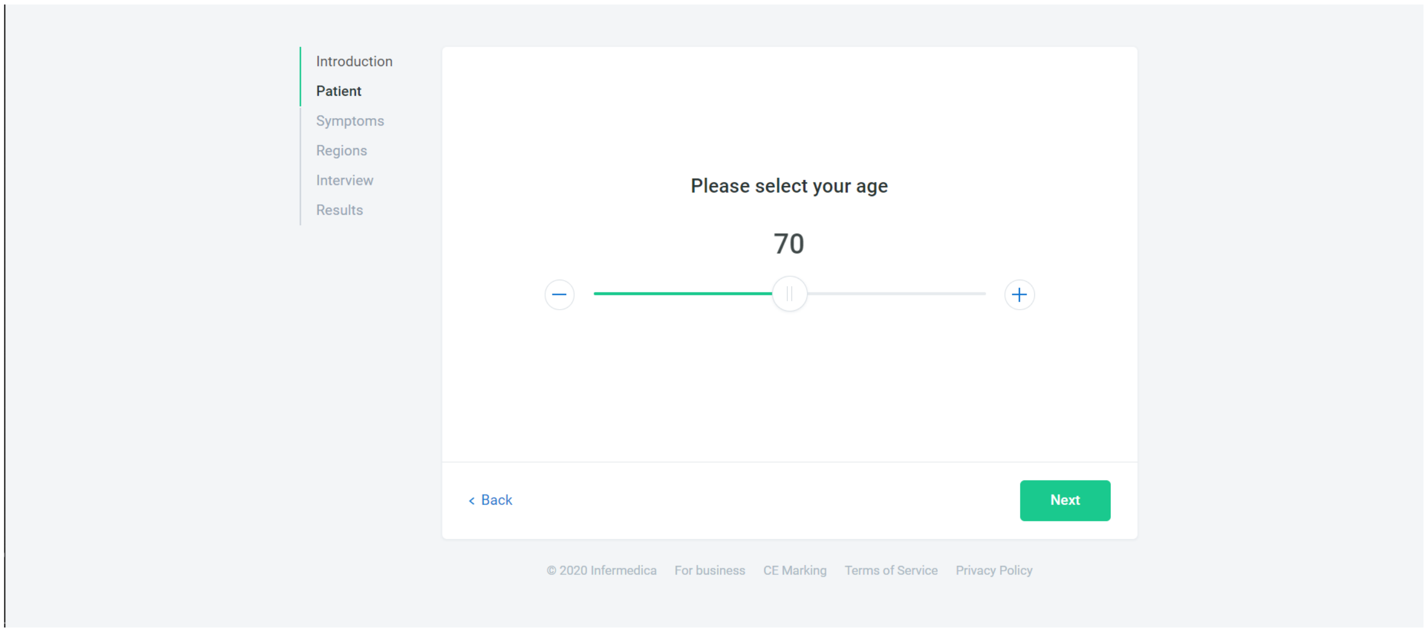
login.aspx.cs

```
using System;
using System.Data;

using System.Data.SqlClient;
using System.Configuration;
using System.Collections;
using System.Web;

using System.Web.Security;
using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
```

```
public partial class login : System.Web.UI.Page

{

    protected void Page_Load(object sender, EventArgs e)

    {

        if (Page.IsPostBack == true)

        {

            return;

        }

        string connectionInfo =
ConfigurationManager.AppSettings["ConnectionInfo"];

        SqlConnection cn = new SqlConnection();
        cn.ConnectionString = connectionInfo;
        if (cn.State == ConnectionState.Closed)

        {

            cn.Open();

        }


        //get category
        string StrCat = "";
        SqlDataReader dr;

        SqlCommand com = new SqlCommand();
        com.CommandText = "GetCategory";
        com.CommandType = CommandType.StoredProcedure;
        com.Connection = cn;

        dr = com.ExecuteReader();
        while (dr.Read() == true)

        {

            StrCat = StrCat + "<tr><td width='126'><a href='buy.aspx?cat=" +
dr["cat_type"].ToString() + "'><font color='#434367'>" +
dr["cat_type"].ToString() + "</font></a></td><tr>";

        }


        sp_cat.InnerHtml = StrCat;
        dr.Close();


    }

    protected void BtnGo_Click(object sender, ImageClickEventArgs e)

    {

        Response.Redirect("buy.aspx?search=" + TxtSearch.Text);

    }

    protected void Button1_Click(object sender, EventArgs e)
    {

        //login and return page call
        if (TxtEmailID.Text == "")

        {

            MyClass.MyAlert(this, "Enter email id.", "123");
            return;

        }

        if (TxtPassword.Text == "")

        {

            MyClass.MyAlert(this, "Enter password.", "123");
            return;

        }

        string connectionInfo =
ConfigurationManager.AppSettings["ConnectionInfo"];

        SqlConnection cn = new SqlConnection();
        cn.ConnectionString = connectionInfo;
        if (cn.State == ConnectionState.Closed)
        {
            cn.Open();
        }


        SqlDataAdapter da = new SqlDataAdapter("select * from member_master 
where mem_email_id='" + TxtEmailID.Text + "' and mem_password='" + 
TxtPassword.Text + "'" , cn);

        DataTable dt = new DataTable();
        da.Fill(dt);

        if (dt.Rows.Count == 1)

        {

            //Alert.Show("ok");


            Session["mem_id"] = dt.Rows[0]["mem_id"].ToString();
            Session["mem_name"] = dt.Rows[0]["mem_name"].ToString();
            Session["mem_email_id"] = dt.Rows[0]["mem_email_id"].ToString();
            Session["mem_last_login"] = 

dt.Rows[0]["mem_last_login_date2"].ToString();


            SqlCommand com = new SqlCommand("update member_master set 
mem_last_login_date2=mem_last_login_date1, mem_last_login_date1=getdate() where 
mem_id = " + dt.Rows[0]["mem_id"].ToString(), cn);

            com.ExecuteNonQuery();

            if (Session["ret_url"]==null)

            {

                Response.Redirect("myaccount.aspx");

                //Server.Transfer("myaccount.aspx");

                //this.ClientScript.RegisterStartupScript(this.GetType(), 
"Alert", "<script language=\"javaScript\">" + "alert('Login Successfully!');" + 
"window.location.href='default.aspx';" + "<" + "/script>");

            }

            else

            {

                Response.Redirect(Session["ret_url"].ToString());

                //this.ClientScript.RegisterStartupScript(this.GetType(), 
"Alert", "<script language=\"javaScript\">" + "alert('Login Successfully!');" + 
"window.location.href='default.aspx';" + "<" + "/script>");


            }

        }

        else
        {
            MyClass.MyAlert(this, "Can not login, Inavlid email id/password.",
"123");
            return;
        }



    }
}
```

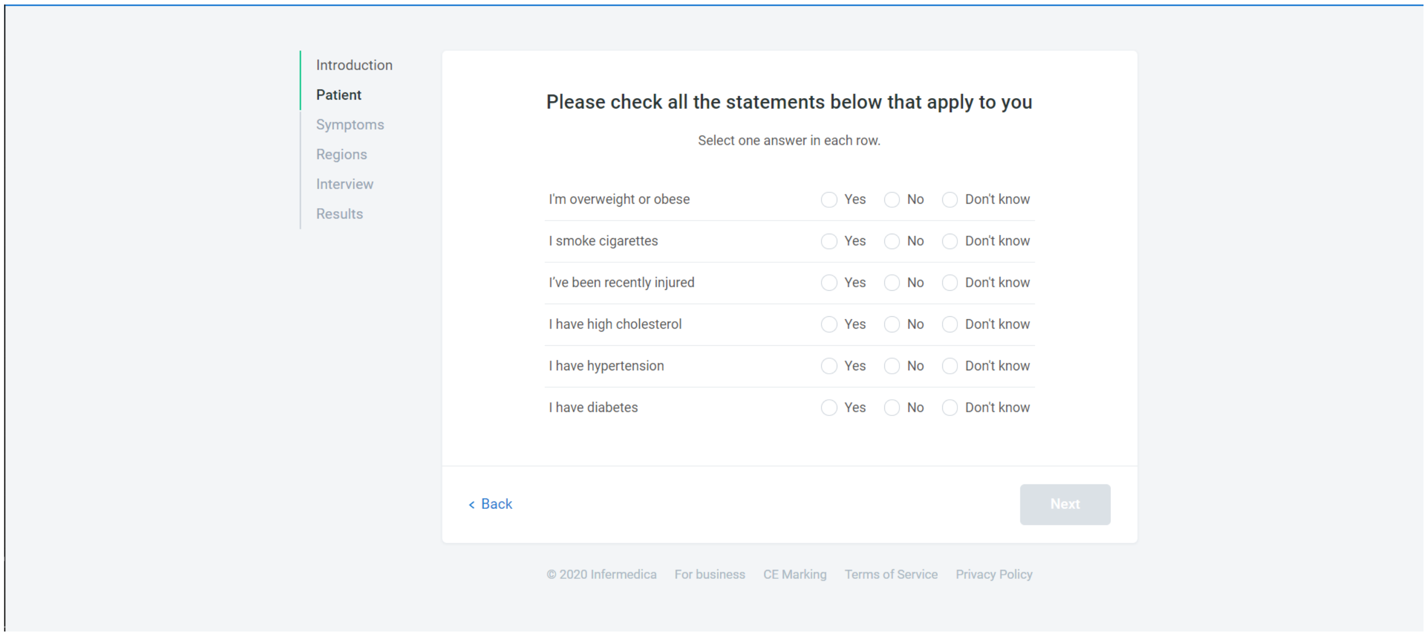


```
<div class="container"><a href="https://DIseasesymptomsanalysis.com"

 class="brand" title="Disease Symptoms analysis – Check your symptoms online">

Disease symptoms analyis</a>

<a class="icon-hamburger" ontouchstart=""

 ng-click="openSidePanelMenu()" ng-if="form.panels.length || form.settings.languages">

<div class="icon-hamburger__inner"></div></a><div class="language-dropdown"

 ng-if="form.settings.languages" ng-class="{open:

languageDropdownOpen}"

ng-mouseenter="languageDropdownOpen = true"

ng-mouseleave="languageDropdownOpen = false"

ng-click="languageDropdownOpen = !languageDropdownOpen">

<a class="lang-switch" ng-bind="getCurrentLanguage().label"></a>

<ul class="languages">

<li ng-repeat="lang in form.settings.languages |
orderBy:'label'"><a ng-class="{'languages__item--active': lang.code ===

form.settings.interface_language}" data-lang-code="{{lang.code}}"

 class="languages__item" ng-bind="lang.label"

 href ng-click="lang.code === form.settings.interface_language ?

 null : goToLanguage(lang)" ontouchstart=""></a></li>

</ul></div></div></header><div ng-class="{'multi-part-screen':

controls.multiPartScreen, 'container-fluid':

 form.settings.fluid_layout, 'container':

 !form.settings.fluid_layout}"
class="main-container"><div class="side-menu"
```



Please select your age

70

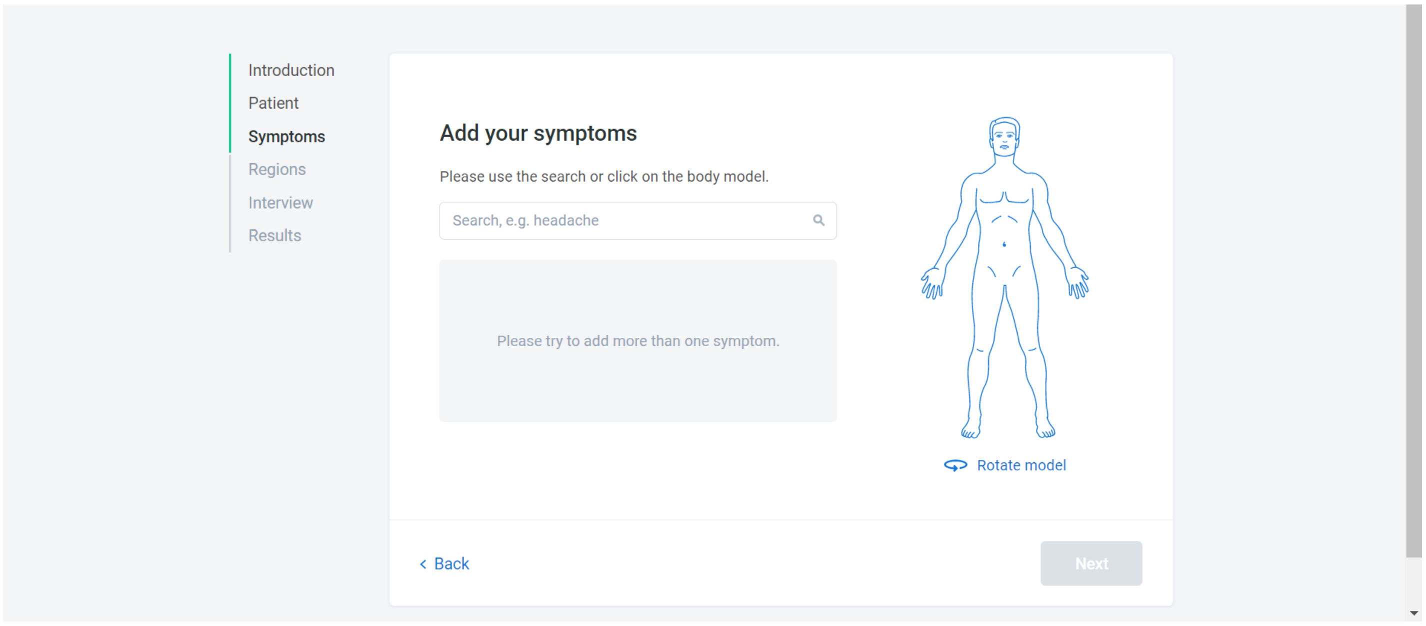


Please check all the statements below that apply to you

Select one answer in each row.

I'm overweight or obese	Yes	No	Don't know
I smoke cigarettes	Yes	No	Don't know
I've been recently injured	Yes	No	Don't know
I have high cholesterol	Yes	No	Don't know
I have hypertension	Yes	No	Don't know
I have diabetes	Yes	No	Don't know

© 2020 Infermedica For business CE Marking Terms of Service Privacy Policy



Introduction
Patient
Symptoms
Regions
Interview
Results

Add your symptoms

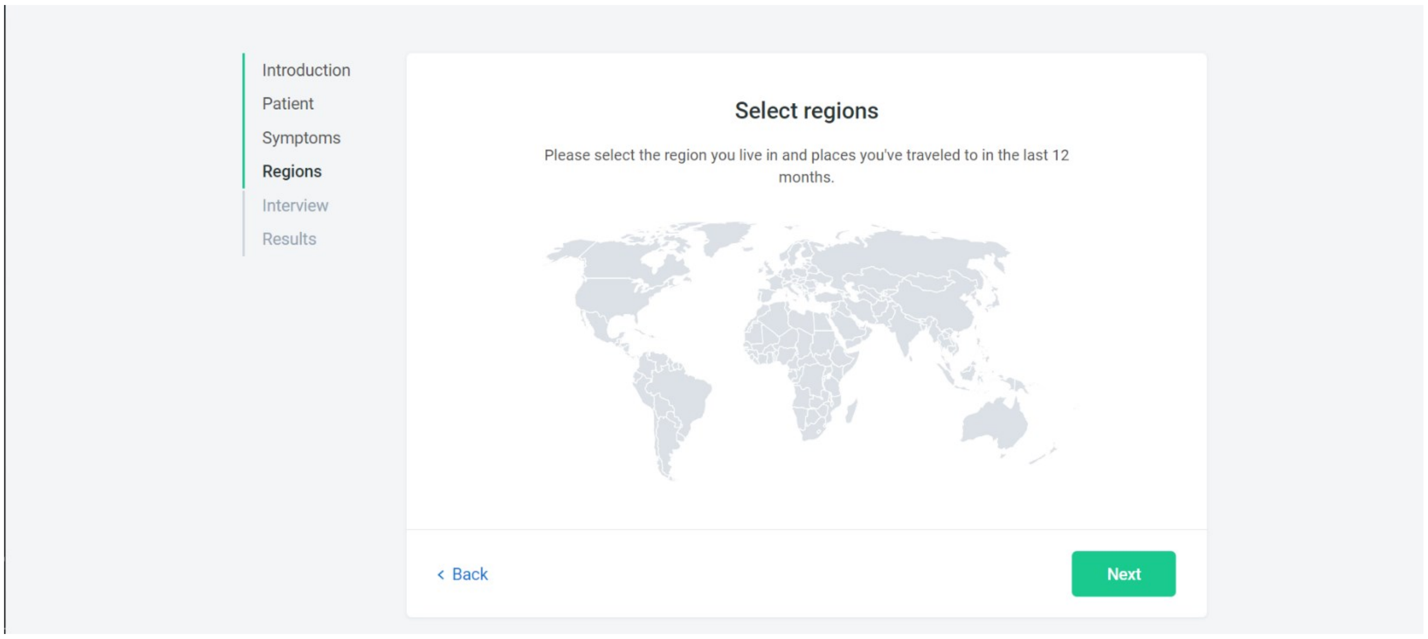
Please use the search or click on the body model.

Search, e.g. headache

Please try to add more than one symptom.

‹ Back

Next



Select regions

Please select the region you live in and places you've traveled to in the last 12 months.

Literature Survey

Monika Gandhi used Naïve Bayes, Decision tree and neural network algorithms and analysed the medical dataset. There are a huge number of features involved. So, there is a need to reduce the number of features. This can be done by feature selection. On doing this, they say that time is reduced. They made use of decision tree and neural networks.

J Thomas, R Theresa made use of K nearest neighbour algorithm, neural network, naïve Bayes and decision tree for heart disease prediction. They made use of data mining techniques to detect the heart disease risk rate.

Sana Bharti, Shailendra Narayan Singh made use of Particle Swarm Optimization, Artificial neural network, Genetic algorithm for prediction. Associative classification is a new and efficient technique which integrates association rule mining and classification to a model for prediction and achieved good accuracy.

Purushottam proposed "An automated system in medical diagnosis would enhance medical care and it can also reduce costs. In this study, we have designed a system that can efficiently discover the rules to predict the risk level of patients based on the given parameter about their health. The rules can be prioritized based on the user's requirement. The performance of the system is evaluated in terms of classification accuracy and the results shows that the system has great potential in predicting the heart disease risk level more accurately".

Sellappan Palaniyappan, Rafiah made use of decision tree Naïve Bayes, Decision tree, Artificial Neural Networks to build Intelligent Heart Disease Prediction Systems (IHDPS).To enhance visualization and ease of interpretation, it displays the results both in tabular and graphical forms. By providing effective treatments, it also helps to reduce treatment costs. Discovery of hidden patterns and relationships often has gone unexploited. Advanced data mining techniques helped remedy this situation.

Himanshu Sharma, M A Rizvi made use of Decision tree, support vector machine, deep learning, K nearest neighbour algorithms. Since the datasets contain noise, they tried to reduce the noise by cleaning and pre-processing the dataset and also tried to reduce the dimensionality of the dataset. They found that good accuracy can be achieved with neural networks.

J.Vijayashree and N.Ch.Sriman Narayana Iyengar used data mining. A huge amount of data is produced on a daily basis. As such, it cannot be interpreted manually. Data mining can be effectively used to predict diseases from these datasets. In this paper, different data mining techniques are analysed on heart disease database. In conclusion, this paper analyses and compares how different classification algorithms work on a heart disease database.
Ramandeep Kaur, Er.Prabhsharn Kaur have showed that the heart disease data contains unnecessary, duplicate information. This has to be pre-processed. Also, they say that feature selection has to be done on the dataset for achieving better results.

J.Vijayashree and N.Ch. Sriman Narayana Iyengar used data mining. A huge amount of data is produced on a daily basis. As such, it cannot be interpreted manually. Data mining can be effectively used to predict diseases from these datasets. In this paper, different data mining techniques are analysed on heart disease database. In conclusion, this paper analyses and compares how different classification algorithms work on a heart disease database.

Existing System

There are various projects exist in this field of machine learning which are capable of doing the analysis of any kind of data by using the different algorithm of machine learning on the given data set.

Health sector generates an immense amount of data and it is also used in some of the research and also gives the desired result to it's users. But no system exist that do the analysis with the more than one type of machine learning algorithm. They only use one type of predefined algorithm that was given to the system but that is not possible that only one type of algorithm can full fill the need of all type of data set.

If that was possible why would be need so many types of machine learning algorithms that exist in our machine learning field like:

- Linear Regression
- Logistic Regression
- Support Vector Machine
- Random Forest
- Naïve Bayes Classification
- K Means
- Decision Tree
- KNN
- Independent Component Analysis

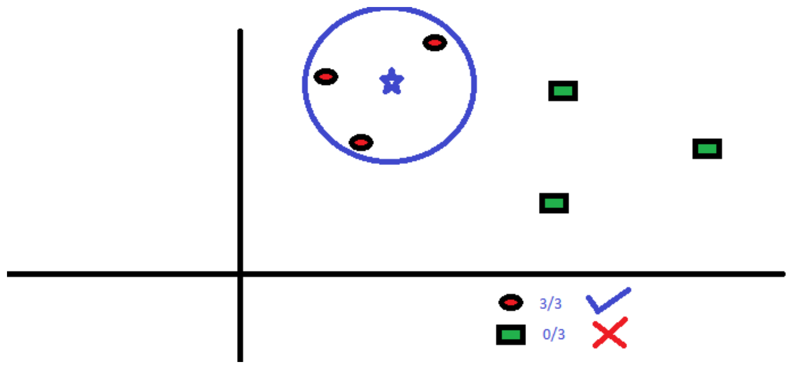
K-Nearest Neighbors (KNN)

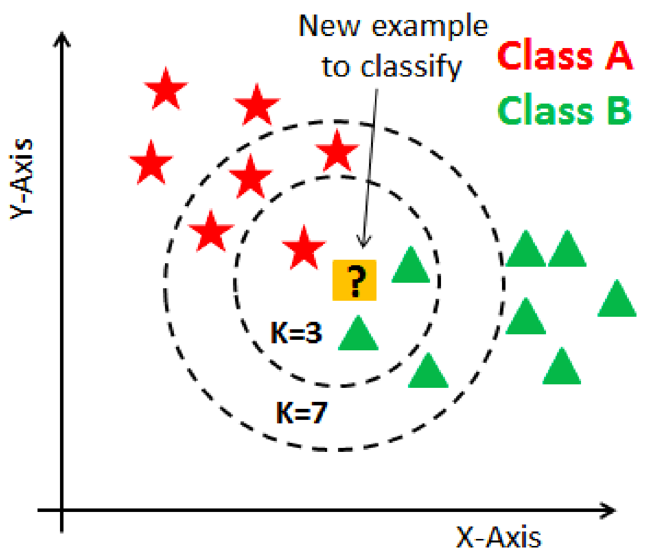
KNN is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. KNN doesn’t have a discriminative function from the training data but it memorizes the training the training data, there is no learning phase of the model and all the work is done at the time of prediction is requested.

A supervised classification algorithm in which we have some data points or data vectors which is separated into different number of several categories and tries to predict the classification of new sample from that particular population set. It classifies any new points on similarity measures that similarity measures for continuous variables, Euclidean distance, Manhattan distance and Minkowski distance measures can be used.
However, the commonly used measure is Euclidean distance. The formula for Euclidean distance is as follows:

$$d = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

The data is divided into training and test sets. The train set is used for model building and training. A k- value is decided which is often the square root of the number of observations. Now the test data is predicted on the model built. There are different distance measures. For continuous variables, Euclidean distance, Manhattan distance and Minkowski distance measures can be used.




New example to classify
Class A
Class B
Y-Axis
X-Axis
?
K=3
K=7

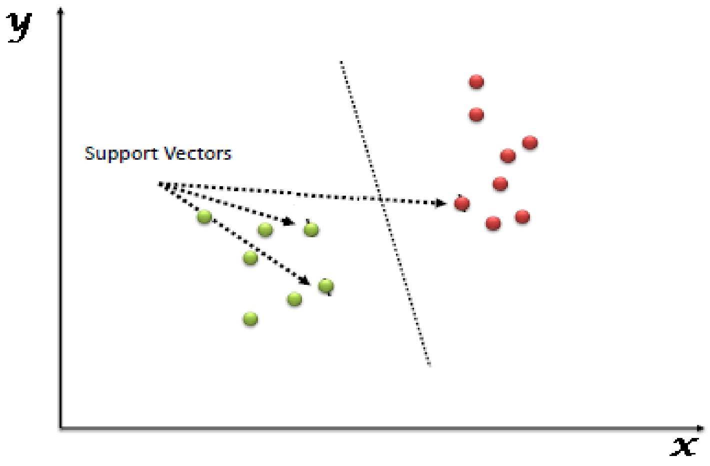
Support Vector Machine (SVM)

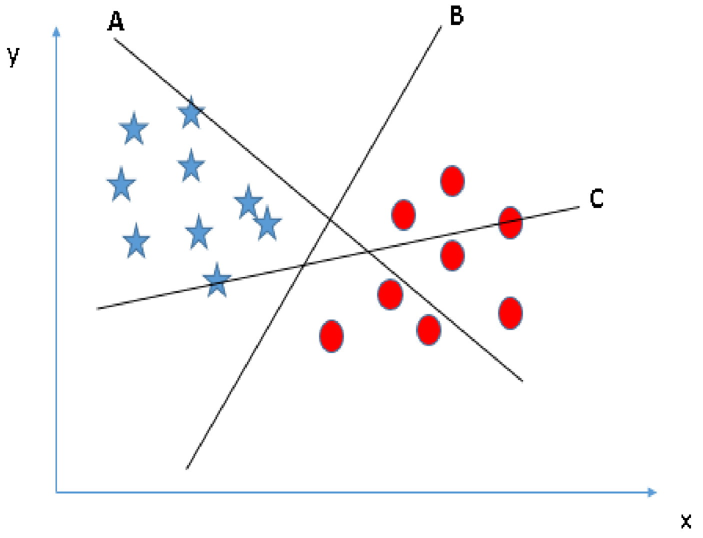
Support Vector Machine is a discriminative classifier that is formally designed by a separative hyperplane. It is a representation of examples points in space that are mapped so that the points of different categories are separated by a gap as wide as possible.

The main aim of the SVM is to segregate the given data in the best possible way. When the segregation is done the distance between the nearest points known as margin. The approach is to select a hyperplane with maximum possible margin between the support vector in the given dataset.

In some datasets the hyperplane may not be useful in that the SVM uses a kernel trick to transform the input into a higher dimensional space. Kernel is a set of mathematical functions. In this proposed methodology, linear kernel is used.

$$K(x\ ,x') = \exp((-||x-x'||^2)/2\sigma^2)$$





Naive Bayes algorithm (NB)

Naïve Bayes is a simple but surprisingly powerful algorithms for predictive analysis. It is a classification technique based on Bayes theorem with an assumption of independent amount of predictors. It comprises of two parts Naïve and Bayes. Naïve Bayes classifiers assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Even if this feature depends on each other or upon to the adjacent of the other feature all these properties independently contribute to the probability. It is easy to built and particularly useful for very large datasets.

Given a hypothesis X and Evidence E, Bayes theorem states that the relationship between the hypothesis before getting the evidence P(X) and the probability of the hypothesis after getting the evidence P(X/Y) is as follows:

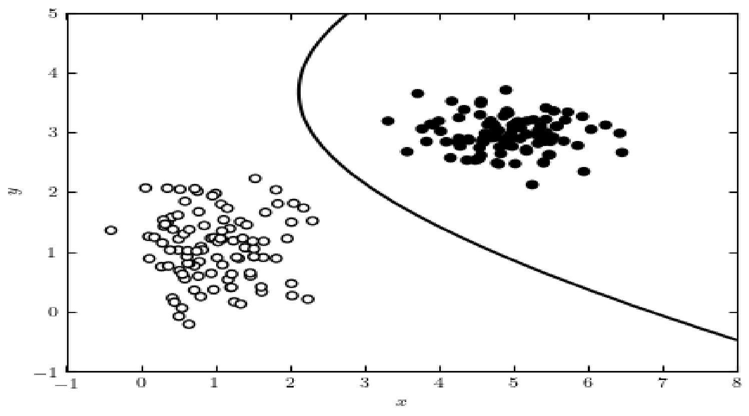
P(Y/X) = P(X/Y) P(X)

This calculates the probability of Y given X where X is the prior event and Y is the dependence event.

Code in Python to plot the simple Gaussian Naive Bayes Classification

```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import colors

from sklearn.naive_bayes import GaussianNB

if "setup_text_plots" not in globals():
    from astroML.plotting import setup_text plots
setup_text_plots(fontsize=8, usetex=True)


np.random.seed(0)
mu1 = [1, 1]
cov1 = 0.3 * np.eye(2)

mu2 = [5, 3]
cov2 = np.eye(2) * np.array([0.4, 0.1])

X = np.concatenate([np.random.multivariate_normal(mu1, cov1, 100),
                    np.random.multivariate_normal(mu2, cov2, 100)])
y = np.zeros(200)
y[100:] = 1

clf = GaussianNB()
clf.fit(X, y)

xlim = (-1, 8)
ylim = (-1, 5)
xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 71),
                     np.linspace(ylim[0], ylim[1], 81))
Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])
Z = Z[:, 1].reshape(xx.shape)
```

```
fig = plt.figure(figsize=(5, 3.75))
ax = fig.add_subplot(111)
ax.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.binary, zorder=2)

ax.contour(xx, yy, Z, [0.5], colors='k')

ax.set_xlim(xlim)
ax.set_ylim(ylim)

ax.set_xlabel('$x$')
ax.set_ylabel('$y$')

plt.show()
```

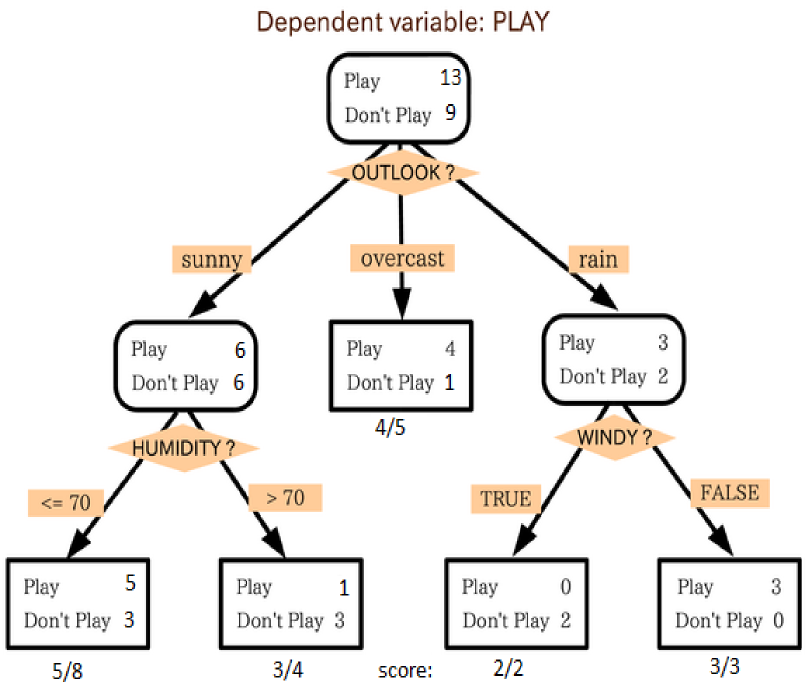


.

Decision trees

A decision tree is a graphical representation of all the possible solutions to a decision based on certain conditions. It is called so because it starts with a root and then branches off to a number of solutions just like a tree. Even the tree starts growing its branches once it gets bigger and bigger, similarly in a decision tree it has a root which keeps on growing with increasing number of decisions and the conditions.

In this research paper we have chosen a dataset which have several factors such as smoking, age, gender. The factor used in root node must clearly classify the data. We make use of age as the root node. The decision tree is easy to interpret. They are non-parametric and they implicitly do feature selection.

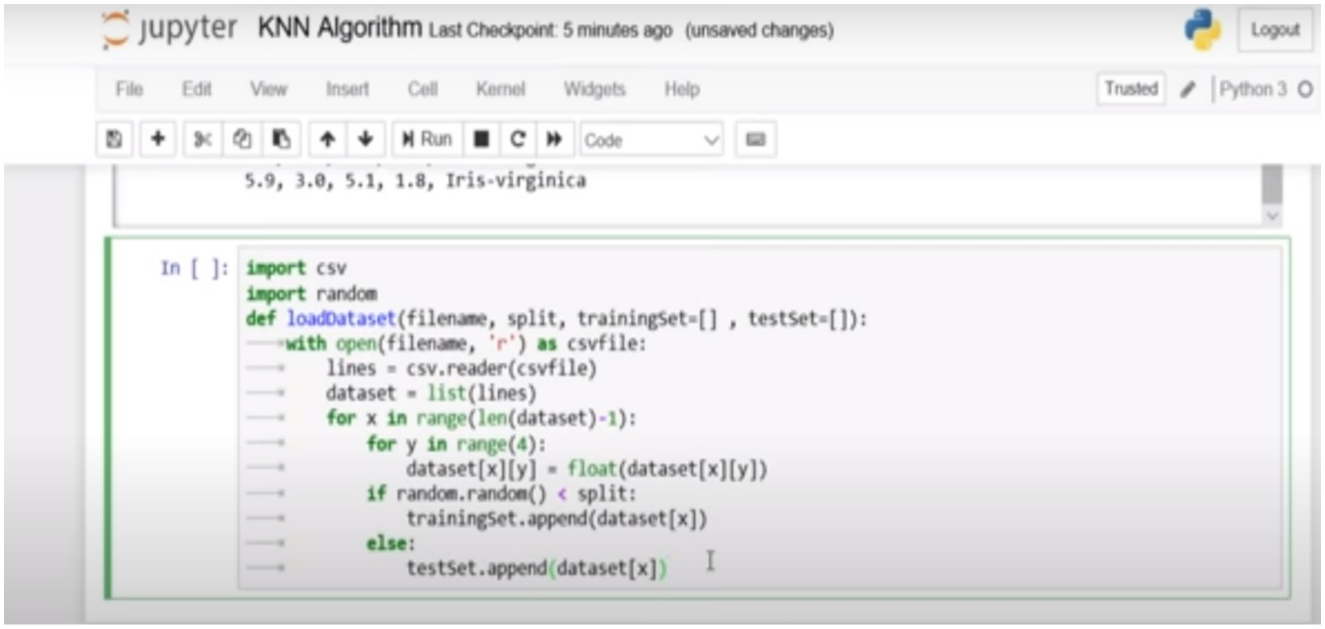


Codes and Screenshots

This section contains the various codes that were used to achieve the accuracy of the models.

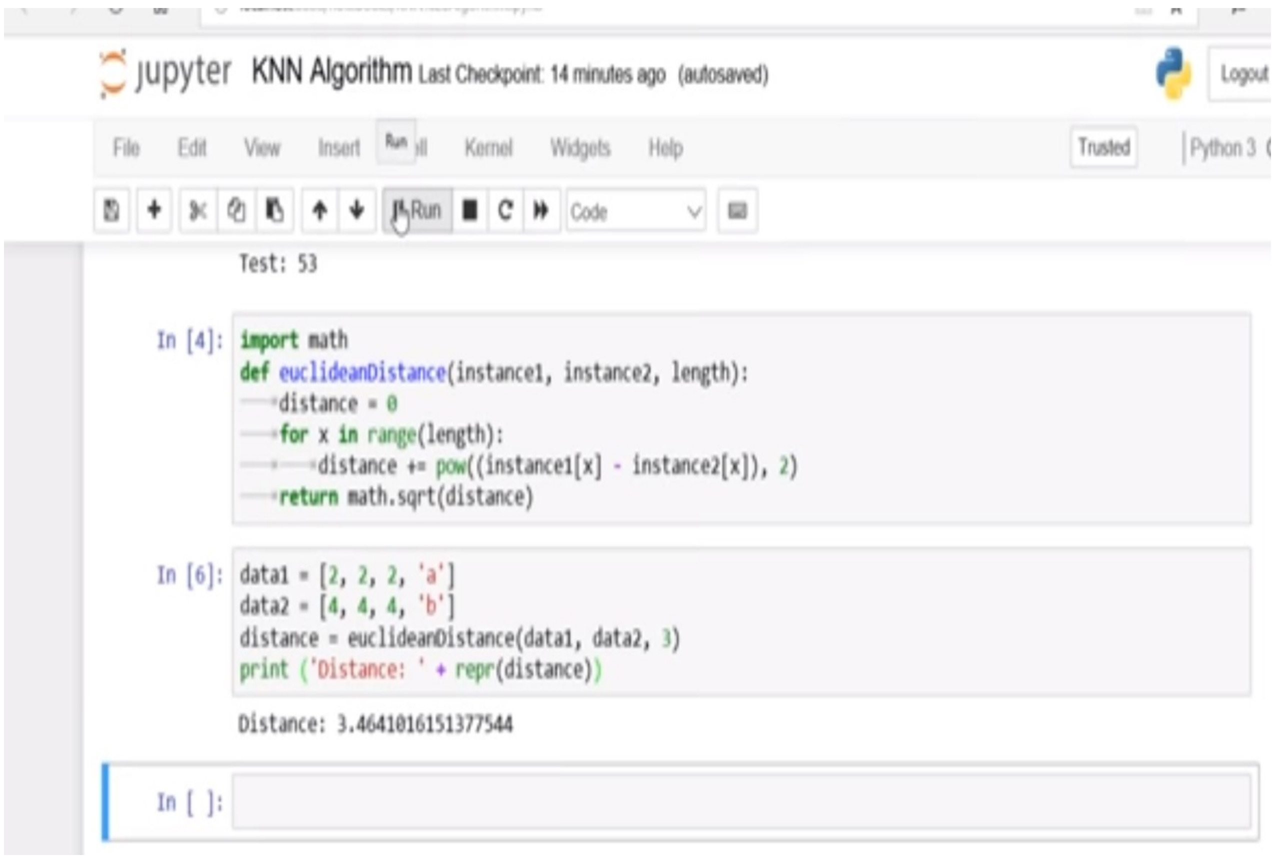
K-Nearest Neighbors (KNN)

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np
df = pd.read_csv("cardiac_train", index_col = 0)
  df
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df.drop('TARGET CLASS', axis = 1))
scaled_features = scaler.transform(df.drop('TARGET CLASS', axis = 1))

df_feat = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_feat.head()
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
      scaled_features, df['TARGET CLASS'], test_size = 0.30)

  from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 1)

knn.fit(X_train, y_train)
pred = knn.predict(X_test)

# Predictions and Evaluations
# Let's evaluate our KNN model !
from sklearn.metrics import classification_report,
confusion_matrix
print(confusion_matrix(y_test, pred))

print(classification_report(y_test, pred))
```


jupyter KNN Algorithm Last Checkpoint: 5 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
5.9, 3.0, 5.1, 1.8, Iris-virginica
```

```
In [ ]: import csv
        import random
        def loadDataset(filename, split, trainingSet=[] , testSet=[]):
            with open(filename, 'r') as csvfile:
                lines = csv.reader(csvfile)
                dataset = list(lines)
                for x in range(len(dataset)-1):
                    for y in range(4):
                        dataset[x][y] = float(dataset[x][y])
                    if random.random() < split:
                        trainingSet.append(dataset[x])
                    else:
                        testSet.append(dataset[x])
```


jupyter KNN Algorithm Last Checkpoint: 14 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
Test: 53
```

In [4]:

```python
import math
def euclideanDistance(instance1, instance2, length):
    distance = 0
    for x in range(length):
        distance += pow((instance1[x] - instance2[x]), 2)
    return math.sqrt(distance)
```

In [6]:

```python
data1 = [2, 2, 2, 'a']
data2 = [4, 4, 4, 'b']
distance = euclideanDistance(data1, data2, 3)
print ('Distance: ' + repr(distance))
```

```
Distance: 3.4641016151377544
```

In [ ]:

Support Vector Machine (SVM)

```
# importing scikit learn with make_blobs
from sklearn.datasets.samples_generator import make_blobs

# creating datasets X containing n_samples
# Y containing two classes
X, Y = make_blobs(n_samples=500, centers=2,
                  random_state=0, cluster_std=0.40)
import matplotlib.pyplot as plt
# plotting scatters
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring');
plt.show()
# creating line space between -1 to 3.5
xfit = np.linspace(-1, 3.5)

# plotting scatter
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring')

# plot a line between the different sets of data
for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
    color='#AAAAAA', alpha=0.4)

plt.xlim(-1, 3.5);
plt.show()
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# reading csv file and extracting class column to y.
x = pd.read_csv("C:\...\cancer.csv")
a = np.array(x)
y  = a[:,30] # classes having 0 and 1

# extracting two features
x = np.column_stack((x.malignant,x.benign))
x.shape # 569 samples and 2 features

print (x),(y)
# import support vector classifier
from sklearn.svm import SVC # "Support Vector Classifier"
clf = SVC(kernel='linear')

# fitting x samples and y classes
clf.fit(x, y)


clf.predict([[120, 990]])

clf.predict([[85, 550]])
```

In [1]:
```python
import numpy as np
import pandas as pd
from IPython.display import Image
from IPython.core.display import HTML
%matplotlib  inline
```

In [4]:
```python
df = pd.read_csv("cardio_train.csv")
df
```

Out[4]:

	id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio
0	0;18393;2;168;62.0;110;80;1;1;0;0;1;0
1	1;20228;1;156;85.0;140;90;3;1;0;0;1;1
2	2;18857;1;165;64.0;130;70;3;1;0;0;0;1
3	3;17623;2;169;82.0;150;100;1;1;0;0;1;1
4	4;17474;1;156;56.0;100;60;1;1;0;0;0;0
...	...
69995	99993;19240;2;168;76.0;120;80;1;1;1;0;1;0
69996	99995;22601;1;158;126.0;140;90;2;2;0;0;1;1
69997	99996;19066;2;183;105.0;180;90;3;1;0;1;0;1
69998	99998;22431;1;163;72.0;135;80;1;2;0;0;0;1
69999	99999;20540;1;170;72.0;120;80;2;1;0;0;1;0

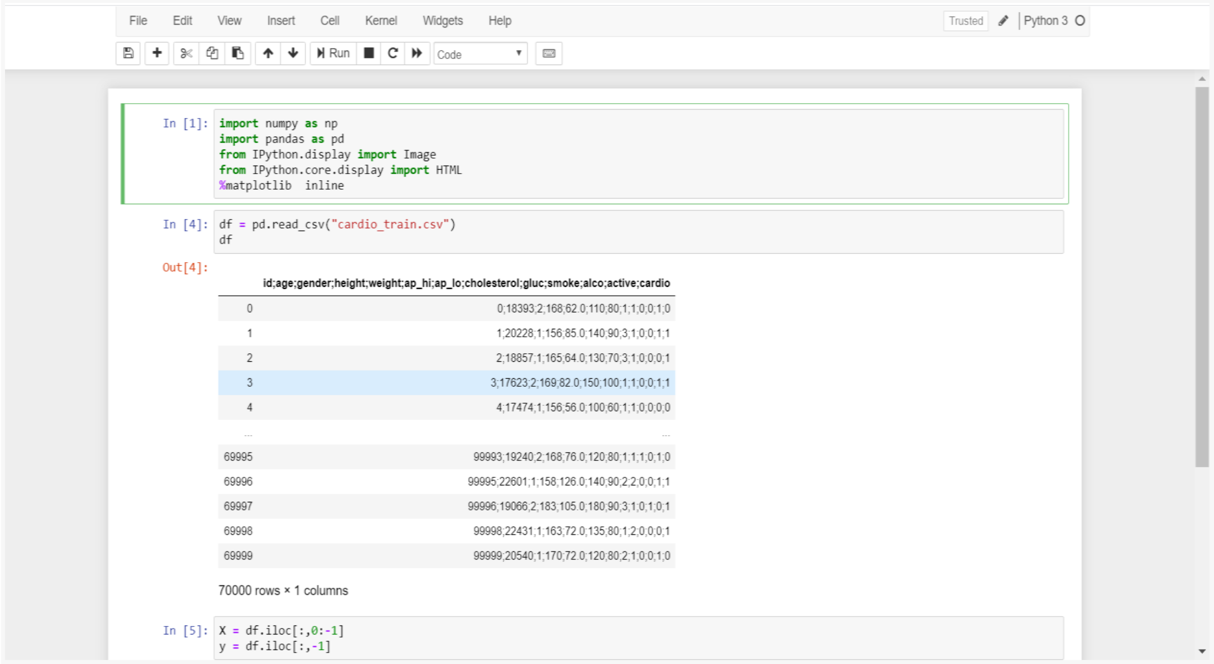
70000 rows × 1 columns

In [5]:
```python
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

Naive Bayes algorithm (NB)

```
import numpy as np

import panda as pd
```

```
from IPython.display import Image
from IPython.core.display import HTML
%matplotlib  inline
df= pd.read_csv(“cardio_train.csv”,names=column)
df
```



```
In [1]: import numpy as np
        import pandas as pd
        from IPython.display import Image
        from IPython.core.display import HTML
        %matplotlib  inline

In [4]: df = pd.read_csv("cardio_train.csv")
        df

Out[4]:
            id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio
      0                                            0;18393;2;168;62.0;110;80;1;1;0;0;1;0
      1                                            1;20228;1;156;85.0;140;90;3;1;0;0;1;1
      2                                            2;18857;1;165;64.0;130;70;3;1;0;0;0;1
      3                                           3;17623;2;169;82.0;150;100;1;1;0;0;1;1
      4                                            4;17474;1;156;56.0;100;60;1;1;0;0;0;0
    ...                                                                               ...
  69995                                        99993;19240;2;168;76.0;120;80;1;1;1;0;1;0
  69996                                       99995;22601;1;158;126.0;140;90;2;2;0;0;1;1
  69997                                       99996;19066;2;183;105.0;180;90;3;1;0;1;0;1
  69998                                        99998;22431;1;163;72.0;135;80;1;2;0;0;0;1
  69999                                        99999;20540;1;170;72.0;120;80;2;1;0;0;1;0

70000 rows × 1 columns

In [5]: X = df.iloc[:,0:-1]
        y = df.iloc[:,-1]
```

In [1]:
```python
import numpy as np
import pandas as pd
from IPython.display import Image
from IPython.core.display import HTML
%matplotlib  inline
```

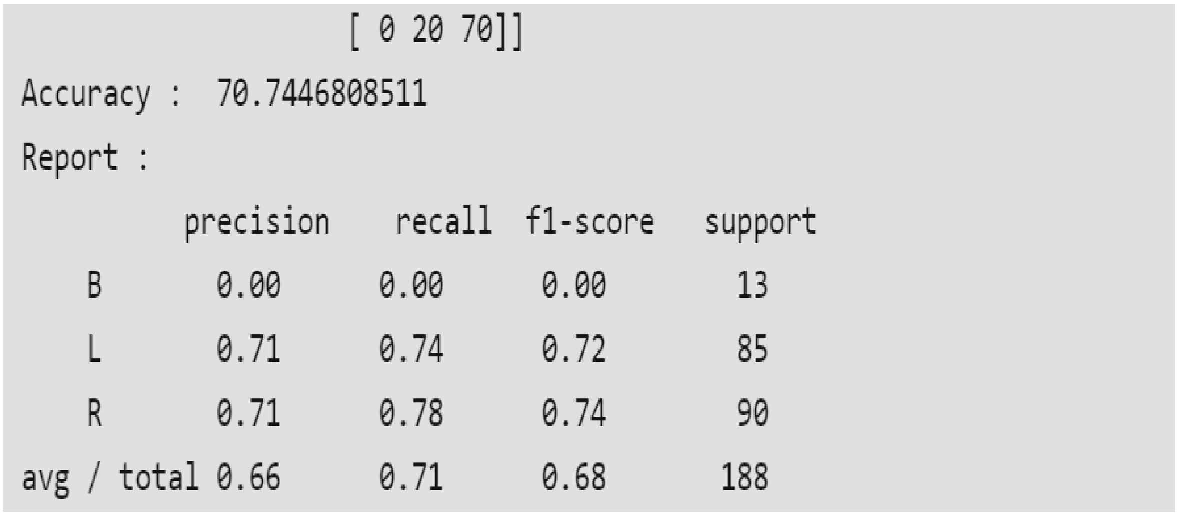
In [4]:
```python
df = pd.read_csv("cardio_train.csv")
df
```

Out[4]:
```
          id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio
    0                                    0;18393;2;168;62.0;110;80;1;1;0;0;1;0
    1                                    1;20228;1;156;85.0;140;90;3;1;0;0;1;1
    2                                    2;18857;1;165;64.0;130;70;3;1;0;0;0;1
    3                                   3;17623;2;169;82.0;150;100;1;1;0;0;1;1
    4                                    4;17474;1;156;56.0;100;60;1;1;0;0;0;0
  ...                                                                      ...
69995                                99993;19240;2;168;76.0;120;80;1;1;1;0;1;0
69996                               99995;22601;1;158;126.0;140;90;2;2;0;0;1;1
69997                               99996;19066;2;183;105.0;180;90;3;1;0;1;0;1
69998                                99998;22431;1;163;72.0;135;80;1;2;0;0;0;1
69999                                99999;20540;1;170;72.0;120;80;2;1;0;0;1;0

70000 rows × 1 columns
```

In [5]:
```python
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

Decision trees

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

def importdata():
    balance_data = pd.read_csv(“cardio_train.csv”,
    sep= ',', header = None)

    # Printing the dataswet shape
    print ("Dataset Length: ", len(balance_data))
    print ("Dataset Shape: ", balance_data.shape)

    # Printing the dataset obseravtions
    print ("Dataset: ",balance_data.head())
    return balance_data

# Function to split the dataset
def splitdataset(balance_data):

    # Separating the target variable
    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    # Splitting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 100)

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):

    print("Confusion Matrix: ",
        confusion_matrix(y_test, y_pred))

    print ("Accuracy : ",
    accuracy_score(y_test,y_pred)*100)

    print("Report : ",
    classification_report(y_test, y_pred))

# Driver code
def main():

    # Building Phase
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = tarin_using_entropy(X_train, X_test, y_train)
    # Operational Phase
    print("Results Using Gini Index:")
# Prediction using gini
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    print("Results Using Entropy:")
    # Prediction using entropy
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)


# Calling main function
if __name__=="__main__":
    main()
```



```
                  [ 0 20 70]]
Accuracy :  70.7446808511
Report :
             precision    recall  f1-score   support
    B          0.00        0.00      0.00        13
    L          0.71        0.74      0.72        85
    R          0.71        0.78      0.74        90
avg / total    0.66        0.71      0.68       188
```

Result and Discussion

Data source

The dataset used is taken from Kaggle named as cardio train.

There were various attributes that were taken into account such as: Gender Age, Gender, Height, Weight, Cholesterol, Glucose, Smoke, Alcohol, Active, Cardio.

Results

After downloading the data the data from Kaggle the same dataset is processes using jupyter notebook on the anaconda navigator. The dataset is uploaded and the data is processed using all the four algorithm we chose before and then their accuracy is calculated

The accuracy results are tabulated as follows:

Method	Accuracy
KNN	80.60%
NB	78.66%
Decision tree	70.58%
SVM	62.56%

The accuracy of K-nearest neighbor algorithm is good when compared to other algorithms.

9. Test Plan

1.3 Introduction

This document describes the user acceptance test plan for the Disease Symptoms Analysis Application. The complete test strategy for the Disease Symptoms Analysis Application is to perform the following kinds of tests, in sequence:

1. **Component testing** of each component that makes up the Disease Symptoms Analysis Application
2. **Integration testing** of the D Application, to ensure the correct interworking of its components
3. **Validation testing** of the Disease Symptoms Analysis Application, to ensure that it works correctly in a pseudo-live environment
4. **User acceptance testing** of the Disease Symptoms Analysis Application, to ensure that its function is acceptable to its users

Acceptance testing is the last set of tests to be performed before the application goes officially live.

9.1 Test Scope

The scope of the user acceptance testing covers:

- Version 1 of the Disease Symptoms Analysis Application
- User-facing functionality defined by a set of use cases
- Administrator-facing functionality defined by a set of use cases

The aim of the testing is to determine how well the application meets its functional requirements from the perspective of the user, and to identify any issues so they can be resolved. Also, the testing serves to compile a set of test data and results that can be used during subsequent test cycles, to test for non-regression of the software in later releases or after the application is in maintenance.

Working practices might vary from user to user and are considered outside the scope of the testing.

9.2 Test Strategy

The basis of user acceptance testing is that other tests were completed successfully, so the application and its required infrastructure are considered to be stable and reliable.
Acceptance testing concentrates on the application from the user's perspective, that is,
how the application is used and whether it meets the necessary quality criteria.

Change requests will be sent to the development team as the actionable documentation. Change criteria will be determined by the Test team and the Development team prior to the beginning of testing. For instance, criteria may include *impact to desired functionality, amount of code impacted by proposed change,* and *design required by proposed change*. The tester will evaluate the criteria. The test lead will determine Change Required or not. Once a bug has been determined as Change Required, the bug report will be translated into a Change Request and passed on to development.

The Member of the acceptance testing is the System Users, Supervisor, Manager and MIS Executive for Disease Symptoms Analysis Application. The progress of the acceptance testing will be reported to the Member, together with any issues that are discovered and their planned resolutions. Sign-off of the tests, and therefore the acceptance of the application, will be performed by the Member or a selected representative.

9.3 Preconditions

The following items are required before testing can take place:

- A complete and coherent functional specification of the Disease Symptoms Analysis Application expressed as use cases and usage scenarios
- A complete and validation-tested release of Disease Symptoms Analysis Application, delivered according to the delivery plan
- An agreed-upon procedure for dealing with any anomalies that are discovered during the testing process
- A set of test specifications describing how each functional area of the Disease Symptoms Analysis Application is to be acceptance tested
- An implemented test environment for the testing
- Sufficient, suitable resources to carry out the testing
- Available standards for the acceptance testing

9.4 Test Priorities

During testing of the Disease Symptoms Analysis Application, the following qualities will be tested in order of priority:

- Functionality—whether the required functions are available and working as expected
- Usability—how user-friendly and intuitive the Disease Symptoms Analysis Application is
- Security—how well-protected and guaranteed corporate and user data is
- Performance—whether the response times are within acceptable limits
- Customization—how straightforward it is to use the application in new, unpredicted ways

9.5 Test Techniques

The following techniques will be applied:

- Scripted tests—sequences of user interactions (based on the use case and usage scenarios) using predefined data sets against predicted results
- Unscripted tests—based on scripted tests, the tester tries to modify the scenarios to explore what-if possibilities
- Penetration tests—scripted tests to attempt unauthorized entry into the system
- Usability checklists—tests to determine the complexity of interactions
- Performance statistics—generation of performance information to check against desired performance criteria

9.6 Test Organization

1.3.1 Roles and Responsibilities

The following roles are defined:

- QA lead/test manager—responsible for planning and ensuring the smooth running of the test process
- Tester—carries out the tests according to the test plan, and then reports the results
- Disease manager—ensures that the tests are carried out successfully from a user perspective
- Project sponsor/client—acts as main stakeholder, and ensures that the needs of the Member community as a whole are considered
- Test support—provides technical assistance, such as test environment configuration, and non-technical assistance, such as methodological support

Weekly team meetings will be held involving the test manager, testers, and disease managers. At these meetings, the progress of the testing process will be reported, any issues will be discussed, and actions will be agreed upon.

9.7 Deliverables

The following deliverables will be expected from the user acceptance testing process:

- Test plan—this document, together with any updates that have occurred during the testing process
- Change requests—any bugs, defects, or other changes required to the Disease Symptoms Analysis Application as a result of the testing process
- Weekly reports—progress reports to enable the status of the testing process to be determined
- Completion report—a report to be signed off by the Member, to signify the successful completion of the user acceptance testing

9.8 Test Environment

1.3.2 Hardware and Software

The test environment will consist of:

1.3.2.1 Server

A single Intel-based computer running:

- Microsoft Windows
- Disease Symptoms Analysis Application components

Client Workstations

Two Intel-based client laptop computers, each running:

- Microsoft Windows XP Professional
- Microsoft Office
- Internet Explorer 6 or greater

The following additional hardware will be required:

- One laser printer to print reports
- One colour printer (laser or inkjet) to print screen dumps
- One CD-ROM drive to enable clean installation of the Disease Symptoms Analysis Application
- Networking connectivity to permit interconnection of the server, clients.

9.9 **Testing Automation Software**

No testing automation software packages are selected at present.

9.10 **Application Configuration**

The following user accounts will be configured on the server:

- System Administrator
- System Users 1
- System Users 2
- Member 1
- Member 2

9.11 Test Management
Tests shall be managed according to the corporate test management standards, which cover:

- Conduct of tests
- Reporting of test results
- Defect tracking and resolution
- Configuration management of the test environment
- Configuration control of test deliverables.

9.12 Testing Schedules
The user acceptance testing schedules are shown in the project structure document and resulting Gantt charts.

9.13 Threats to Testing
Potential threats to the testing process are as follows:

- **Insufficient resources available for testing.** Testing resources have been seconded from the development departments , whose time is at a premium. Mitigation: ensure department heads apply a high priority to the testing of the Disease Symptoms Analysis Application.

Conclusion and Future Work

This paper discusses the various machine learning algorithms such as support vector machine, Naïve Bayes, decision tree and k- nearest neighbour which were applied to the data set. It utilizes the data such as age, cholesterol, alcohol consumptions and then tries to predict the possible coronary heart disease patient in coming years.

Family history of heart disease can also be a reason for developing a heart disease as mentioned earlier. So, this data of the patient can also be included for further increasing the accuracy of the model.

This work will definitely help the current generation and the generations to come in identifying the possible patients who may suffer from heart disease in the coming years. This may help in taking preventive measures and hence try to avoid the possibility of heart disease for the patient. So when a patient is predicted as positive for heart disease, then the medical data for the patient can be closely analysed by the doctors.

In this research paper we had only used four supervised algorithm in future the other types of supervised algorithm and also the different types of machine learning algorithm can be used to predict the data and the accuracy of those algorithm can be better than the algorithm used in this research paper.

References

Monika Gandhi, Shailendra Narayanan Singh Predictions in heart disease using techniques of data mining (2015)

J Thomas, R Theresa Princy Human heart disease prediction system using data mining techniques (2016)

Sana Bharti, Shailendra Narayan Singh, Amity university, Noida, India Analytical study of heart disease prediction comparing with different algorithms (May 2015)

Purushottam, Kanak Saxena, Richa Sharma Efficient heart disease prediction system using Decision tree (2015)

Sellappan Palaniyappan, Rafiah Awang Intelligent heart disease prediction using data mining techniques (August 2008)

Himanshu Sharma,M A Rizvi Prediction of Heart Disease using Machine Learning Algorithms: A Survey (August 2017)

Animesh Hazra, Subrata Kumar Mandal, Amit Gupta, Arkomita Mukherjee and Asmita Mukherjee Heart Disease Diagnosis and Prediction Using Machine Learning and Data Mining Techniques: A Review (2017)

V.Krishnaiah, G.Narsimha, N.Subhash Chandra Heart Disease Prediction System using Data Mining Techniques and Intelligent Fuzzy Approach: A Review (February 2016)

Ramandeep Kaur, 2Er. Prabhsharn Kaur A Review - Heart Disease Forecasting Pattern using Various Data Mining Techniques (June 2016)

J.Vijayashree and N.Ch. SrimanNarayanaIyengar Heart Disease Prediction System Using Data Mining and Hybrid Intelligent Techniques: A Review (2016)