TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

OUTREACH CLINICAL DIAGNOSIS SYSTEM

By:

Aashish Acharya (066BCT501)

Dicchya Shakya (066BCT511)

Supritam Raj Shrestha (066BCT542)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

SEPTEMBER, 2012

COPYRIGHT

ACKNOWLEDGEMENT

Although this is a minor project of three individuals, this project would not have been a success without the help of some people and institutions. We would like to express our sincere thanks to the Department of Electronics and Computer Engineering for providing us the opportunity to explore our interest and ideas in the field of Computer Engineering through this project. We go to learn a lot of new things and got a chance to be involved in designing a system like never before. Our special thanks goes to Dr Arun Timalsina, HOD and Project Coordinator for his encouraging words and special support on us in course of making this project. We are grateful to Mr Shayak Raj Giri, our project coordinator for his help all along. Without his suggestions and periodic evaluations of our project, this work of ours would not have been a success.

We would like to thank Mr Samant Manadhar, our project mentor for his continuous support and guidance in the course of doing the project. Without his suggestions and experiences, we would not have been able to coordinate properly amongst ourselves while working for this project of ours. We also would like to extend thank you to all our friends for helping us decide this topic for Minor Project.

We would like to thank Nick Simon's Institute Nepal for providing us with the medical algorithms and support materials that were required for the completion of the project. Lastly, we would like to thank each and everyone who, directly and indirectly, have been a part of this project. Your participation has taken this project to where it is now. May god bless you all!

Regards,
Aashish, Dichhya, Supritam

ABSTRACT

Health is a significant and very essential measure of life. The 'Outreach Clinical Diagnosis System' is an app that serves to provide health services to the community. In a country like ours, where due to the wide variation in the geographic formation and demographic variety, not all the people are properly facilitated with the basic health services. The basic reason for this is due to the lack of availability of heath facilities and health experts in the rural and geographically challenged part of the nation. Since the geographically rural part of the country also lags in terms of availability of ICT, implementation of e-Health is also of a challenge. The application of ours is a inference engine based decision support system that supports a health worker while in the process of diagnosis of a patient in a health care station. This application is designed to be used by a trained health professional only. The built of our application is inspired from a popular medical record system called the Open-MRS. Our application flow starts with a chief complaint, followed by a series of questionnaires and finally exits with the diagnosis of the diseases. This is acquired through the use of Nick Simon's Algorithms for Medical Diagnosis. The details of patient's visit (encounter) to the health care station and the whole of the diagnosis process is stored in the database for future use. The details can be obtained, transferred or further analyzed as per the requirements. Due to the shortage of time, we could not add a built in feature to incorporate the international conventions for medical data transfer over a network. But this does not compromise the performance of the system. We have tried to bridge the gap between the patients and health care workers by designing a highly efficient decision support system through Outreach Clinical Diagnosis System.

TABLE OF CONTENTS

Table of Contents

TABLE OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

ASP	Active Server Page
EDI	Electronic Data Interchange
EDMX	Electronic Data and Metadata Exchange
HL7	Health Level 7
LEMR	Longitudinal Electronic Medical Records
Microsoft PHVC	Microsoft Patient Health Vault Connect
MVC	Model View Controller
NSA	Nick Simon's Algorithm
NSI	Nick Simon's Institute
OCDS	Outreach Clinical Diagnosis System
Open-MRS	Open Medical Record System
XML	Extensible Mark-up Language

OBJECTIVES:

- Implementing Nick Simons Algorithm into a complete Computer-Based System and provide a pilot testing environment to see for its feasibility.
- To complete information of diagnosis process followed by the health-worker to the senior doctors so that communication gap can be bridged.
- To automate the process of reports that needs to be generated and provided to the concerned authorities.
- To provide a platform for the development of the relevant Information System that provides a better solution to manage the information existing in any health institution.
- To provide the diagnostic path followed by the patient for the ease of future diagnosis.
- To facilitate the generation of the health reports as per the format specified.
- To bridge the digital divide by commencing an easier method to health processes.

INTRODUCTION

Outreach Clinical Diagnosis System:

With the intention to eradicate all the health related problems in the life of the individual, to support the health workers, we tend to implement a computerized medical diagnosis application "Outreach Clinical Diagnosis System", which deals with the individual health status and results in synopsis prescription, reports and feedbacks.

Most of the working population of our country is centred in the cities and urban areas. There is a significant population in the rural part of the country that is deprived of most of the services and facilities including the health. There are children, mothers and sexagenarians in these parts of the country whose health status is more vulnerable than that of the city people. So, community health in the rural areas is more challenging.

Outreach Clinical Diagnosis System is a patient-centric application that captures interactions between a health care provider and a patient. It is designed to manage patient data longitudinally, linking multiple interactions over time into a single patient chart. Having this complete patient history available empowers clinicians to make more informed decisions about health care, while also enabling a deeper analysis of patient health in order to draw more meaningful conclusions on improving outcomes

It gives an individual the ability to view, store, and update and communicate his/her personal information. The application contains data about allergies, medications, illness, vaccinations, and social history. Outreach Clinical Diagnosis System can be a better approach to reach out to a greater number of people in regards to the health services. With a basic knowledge of the computer and the internet, any trained health worker can monitor the diagnostic procedures both at the rural as well as the urban parts of the country, thereby benefitting a larger population. The decision support system of the Outreach Clinical Diagnosis System is built to be as accurate as possible but does not contain of any treatment procedures. Outreach Clinical Diagnosis System contains the basic health problems and the diagnostic methodologies that can be monitored by a health worker. The overall diagnostic procedures can be monitored by the doctors and the specialists to assure the effective health practices.

Project Background:

In the country with a large disparity in urban and rural development, a critical challenge exists in medical personnel retention in rural areas. The current health situation of our country is such that there are 10,000 people per doctor to be looked after. The situation of health facilities at the rural parts is even fierce. There are not enough manpower and resources for the country to mobilize in the health sector. The sector of health science has not been that fancy to the young students like us to research on and do some project on the same. Yet, every year, millions of dollars are invested in a quest to find better and easier approach to provide health services to the

people of the world. Our country being a third world country does not have effective health service anywhere in its priority. We being the technological literates and close to the situation of the country, we attempted to dig on the available technologies and find the best of the possible implementations. In this way, we finally got into doing the project.

Problem Statement:

A healthy person is a happy person. Taking risk with our health is taking one step further towards death. Every country requires healthy human resource. Health of the citizens depends upon the availability of health services. In the urban areas, there is no problem with regard to the availability of the health services,-hospitals and doctors. But in case of the remote areas, finding health posts is very difficult. People in the rural areas have been deprived of health services, in the absence of health workers, despite the establishment of health posts. Doctors and health workers hesitate to work at the remote areas. People in remote and rural areas themselves are less interested to get treatment in hospitals or health posts. Rather they prefer the traditional healing methods. It is sheer negligence on towards those people by not deploying doctors and health centre. The figure below the pattern of the migration of health workers within a country like ours:

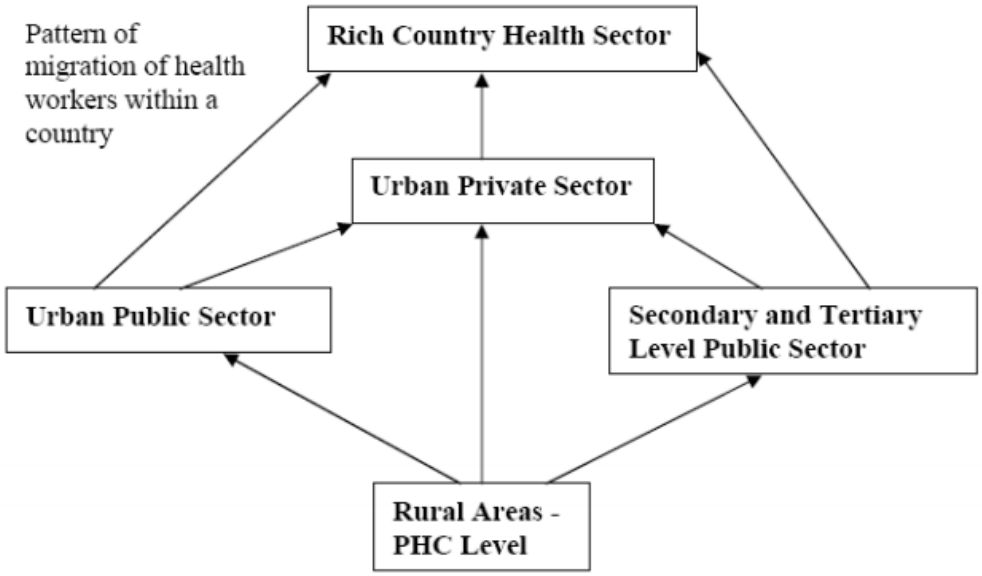

Figure 1 Migration Pattern of rural healthcare workers in Nepal

The limited numbers of health posts that have been established across the country have become the last stop for most of the people at the rural part of the country but not the first choice. The health workers there do not get to interact frequently the doctors and hence their communication to the central resource centre is highly compromised. They cannot keep up with the updates and latest breath through. The health care condition is made even worse due to the lack of transportation and awareness among the people.

LITERATURE REVIEW

Our project was not a design of functionality. Our problem domain was the health diagnosis process. We needed to analyze it, spot the problems and build the conceptual solution first. Then, based on that, we needed to present an implementation prototype to address the problem of medical diagnosis. In this course, we studied various alternatives and existing technologies to effectively model our problem domain. The following are the list of technologies that we incorporated, in some way to model the diagnosis process:

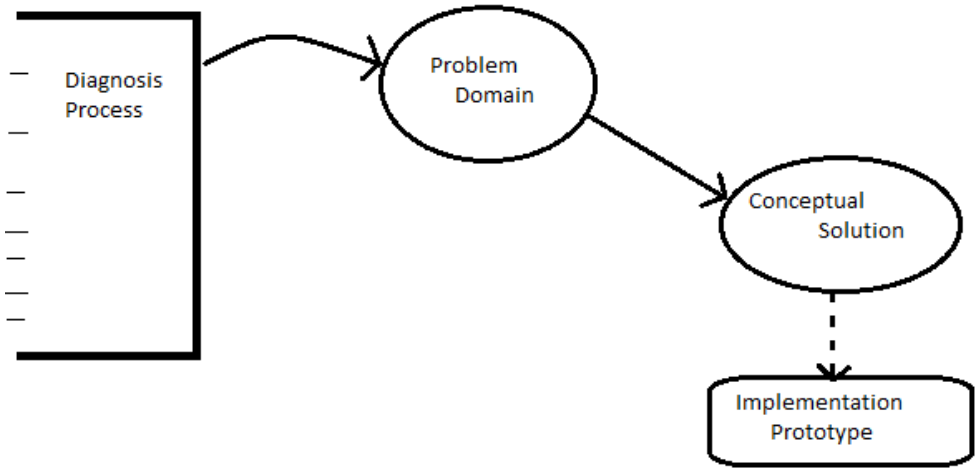

Figure 2 Process of OCDS Design

Microsoft PHVC:

Microsoft Health Vault is web-based health data storage and sharing platform that allows us to store copies of our health records online. The data that is being stored into the health vault may be acquired from various resources like hospitals, pharmacies, health post etc. Health Vault supports various web-based applications thus allowing access to the patient's data with consent from the patient. The details can be exported to any medical personnel if required by the person to support his diagnosis process. We studied the data model of this product and learnt how we could model the various entities of diagnosis process.

The Health Vault platform offers:

- An online storage facility for consumers to keep health-related information.
- A set of features consumers can use to collect, view, share, and transfer health-related information.
- A set of features various health-related service providers can use to offer their consumers services connected with their customers' health-related information stored in Health Vault.

Health Vault Data Model:

a. Interoperable: Designed to work well with industry standards
b. Inclusive: Use structure data when available but supports unstructured data
c. Just In Time: A Growth data model based on real-world requirements

d. Independent data elements: Minimizes relationship and allows connection without enforcing referential integrity.

Developing Health Vault Applications:

1. Platform XML Services
2. Health Vault .Net SDK
   - Encapsulates all XML functionality
   - Strongly typed object model
   - Source code available
   - Includes a variety of sample applications
3. Community created libraries
   - Java, PHP, Ruby and Cold Fusion projects exist today
   - Integrated into public repositories (Codeplex, Sourceforge, Rubyforge, etc.)

Health care providers should not use Health Vault as an electronic medical record system. Health care providers should import a copy of any Health Vault data into their own medical record system prior to using that data to make clinical decisions. Since the data in Health Vault accounts is managed and controlled by individuals (consumers and patients), health care providers should evaluate its completeness and accuracy as they would other information received from their patients or their patients' families.

Some problems in implementing Health Vault:

i. To implement Health vault into an application, an agreement must be signed with the company regarding their policies and privacy concerns
ii. Applications can access the records only if the applications have been authorized by the users and also they must be registered to the Health Vault
iii. For users to be able to approve the authorization, they must have a registered account at the health vault (compulsory)
iv. Regarding point ii and iii, we have a major problem. Health vault is currently being implemented at selected areas only. Users from Nepal are not authorized to create an account in Heath Vault, thus it won't be possible to create an application since it will have problems with authorization and user authentication.

OpenMRS:

OpenMRS is a popular open source electronic medical record which has been a striking choice for clinics and research institutions due to its open nature, low deployment costs, and ease of extending the system through modules. It is a client-server application based on the principle that information should be stored in a way which makes it easy to summarize and analyze, i.e., minimal use of free text and maximum use of coded information. We studied OpenMRS to design the database models for the storage of information obtained in course of diagnosis process.

OpenMRS is programmed in Java and the core application works through a web-browser. Hibernate is used as an interface layer to the database. Tomcat is used as the web application server. The back end database is currently in MySQL. The system creates XML schemas for

form design. Form design and form data entry is currently done in Microsoft Infopath, HTML, or XForms. When form data entered is submitted, it is converted into a HL7 message before going into the database.

The OpenMRS API is designed to sit between the web application and the database and simplify access to the data model. OpenMRS can run on any platform that supports Java technology, such as Linux, Windows, and Mac OS X.

Features of OPENMRS:

- **Central concept dictionary**.
- **Security:** User authentication.
- **Privilege-based access:** User roles and permission system
- **Patient repository:** Creation and maintenance of patient data, including demographics, clinical observations, encounter data, orders, etc.
- **Multiple identifiers per patient:** A single patient may have multiple medical record numbers
- **Data entry:** With the FormEntry module, clients with InfoPath (included in Microsoft Office 2003 and later) can design and enter data using flexible, electronic forms. With the HTML FormEntry module, forms can be created with customized HTML and run directly within the web application.
- **Data export:** Data can be exported into a spreadsheet format for use in other tools (Excel, Access, etc.)
- **Standards support:** HL7 engine for data import
- **Modular architecture:** An OpenMRS Module can extend and add any type of functionality to the existing API and web app.
- **Patient workflows:** An embedded patient workflow service allows patient to be put into programs (studies, treatment programs, etc.) and tracked through various states.
- **Relationships:** Relationships between any two people (patients, relatives, caretakers, etc.)
- **Patient merging:** Merging duplicate patients
- **Localization / internationalization:** Multiple language support and the possibility to extend to other languages with full UTF-8 support.
- **Support for complex data:** Radiology images, sound files, etc. can be stored as "complex" observations

  OpenMRS can be integrated with SMS messaging, allowing community health workers to add information about adherence to medication regimens to a patient's record, as they make rounds through villages in rural areas.

Nick Simon's Algorithm:

Nick Simon's Algorithm is a medical decision making tool used to go from symptoms to diagnosis. This is different from most medical algorithms found in text books, which go from diagnosis to treatment. These clinical algorithms are devised by a group of experts at Nick Simon's Institute. Nick Simon's Institute is an organization working in Nepal with a mission to train and support skilled, compassionate rural health care workers.

These clinical algorithms have been developed to aid the health care worker in diagnosing

common problems seen in a primary health care setting. The algorithms are based on common chief complaints. Algorithms are primarily based on history questions with confirmation by physical examination. There is no need for any laboratory or other diagnostic tests in using these algorithms.

The implementation of these algorithms is commenced by determining the chief complaint and then the related specific algorithm is searched. The latter processes are simply the implementation of a decision tree on the basis of 'yes' and 'no' questions. Final result will yield the disease of the given symptom.

It is to be noted that this algorithm is only for the diagnosis and not for the treatment. It is also important that the diagnosis using these algorithms be monitored by a doctor for safe keeping.

Evaluation of the Nick-Simon's Algorithm:

1. The diagnosis is based on the typical Yes/No query done by the medical practitioner with the patient. The practitioner follows a set of query and based on the patient's answers, the problem is carried out.
2. Usually each diagnosis consist of a single entry point, basically a yes/no query relevant to patient's problems. (There are some exceptions where the entry point may be another algorithm or a warning signs.)
3. Following the binary path, the diagnosis usually ends up at a single terminating point at which the problem/disease is identified. (There are some cases that may require follow ups or further evaluation.)
4. At the end of diagnosis, the result acquired is represented using color codes, each color code have its own meaning and is used to define the severity of the situation and further action.

Each step of the diagnosis query can be:

a. A Starting point
b. A Terminating point
c. A Diagnosis Query requiring a Yes/No answer
d. A Warning Sign Query that requires information supplied by the patient
e. A Path to a different algorithm for further diagnosis

Problems in implementing generic algorithm:

1. Sometimes the algorithm begins with a set of warning sign query; this might be little difficult to implement as the information required is more than that in a single yes/no query.
2. Some algorithms have extensive branching depending upon the time constraint of the disease. E.g. Fever Algorithm. The query is not a yes/no query but based on the time.
3. Some algorithms like the extremity trauma have multiple outcomes despite being a yes/no query. The result of a query has multiple choices.
4. Some algorithms like the burn algorithm does not comprise of the typical yes/no query. Require s different forms of query.

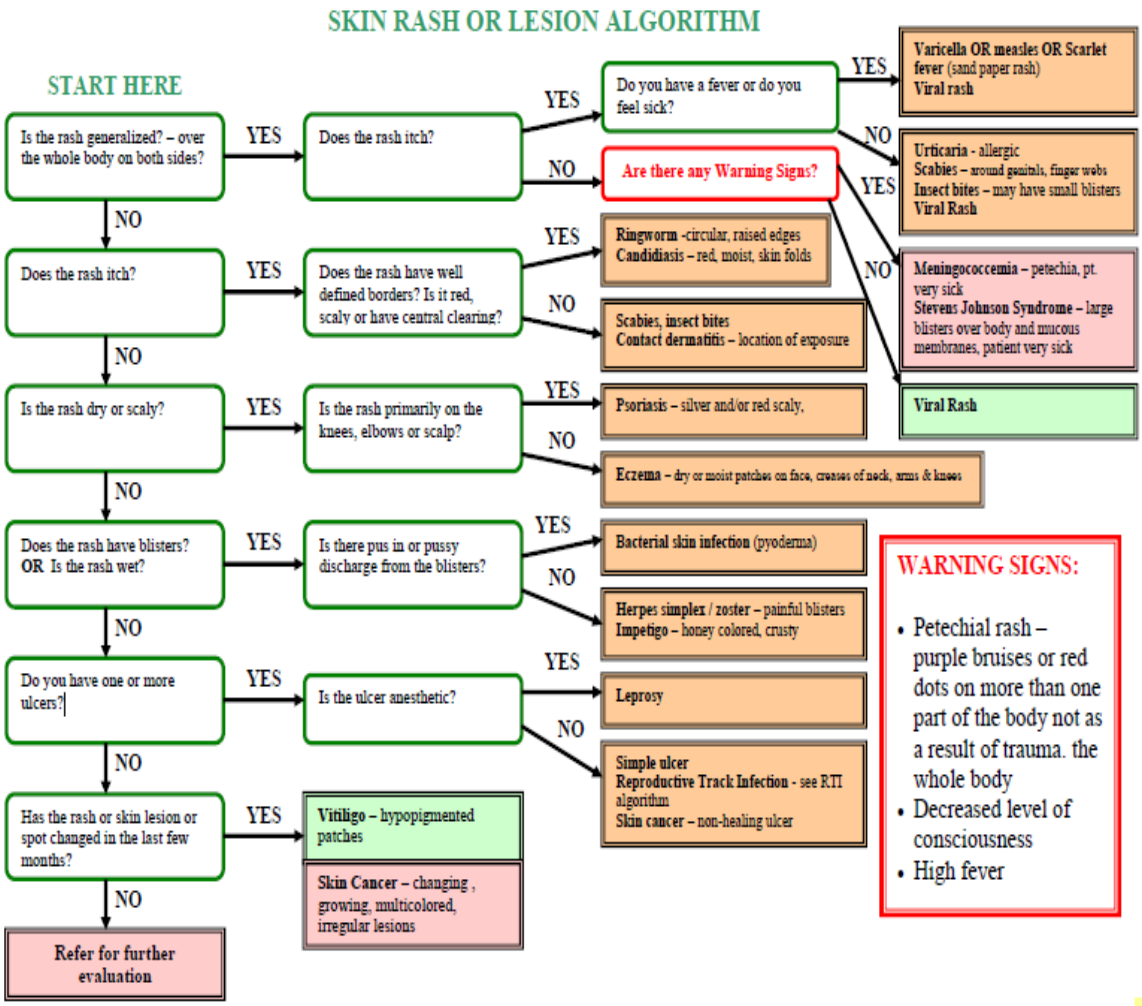

Figure 3 Nick Simon's Algorithm for Skin Rash

TECHNOLOGY USAGE OVERVIEW

ASP.NET programming platform

ASP implies for the Active Server Page developed by the Microsoft Corporation. It was developed for the dynamic creation of standard web based solutions. And ASP.NET is the next version of the ASP which is the programming framework to develop the enterprise level web application providing the rich set of features and easy integration with the databases, web services and web application with the features of web forms, MVC, dynamic data scaffolding, AJAX framework, templates and themes and a lot of such features enables the web developer to build the web application better and faster. As being a part of the Microsoft.NET framework, it offers a built-in security, reliability and robustness while using the various styles and patterns based on the needs of the projects.ASP.NET introduces the 2 main features:

- Web services
- Web forms

ASP.NET uses the Common Language Runtime (CLR) which is provided by the .NET framework to manage the execution of the code and removes the language barrier. It introduces the advantages to the web application by improved performance, flexibility, configuration settings and security.

MVC3 design pattern

The ASP.NET MVC 3 Framework is the latest evolution of Microsoft's ASP.NET web platform. It provides a high-productivity programming model that promotes cleaner code architecture, test-driven development, and powerful extensibility, combined with all the benefits of ASP.NET4.

In the third edition, the core model-view-controller (MVC) architectural concepts are not simply explained or discussed in isolation, but are demonstrated in action. The new features added can be organized as below

- Extensible scaffolding with MvcScaffold Integration
- HTML5 enabled project templates
- The razor view engine
- Support for multiple view engines
- Controller Improvements
- JavaScript and Ajax
- Model Validation Improvements
- Dependency Injection Improvements

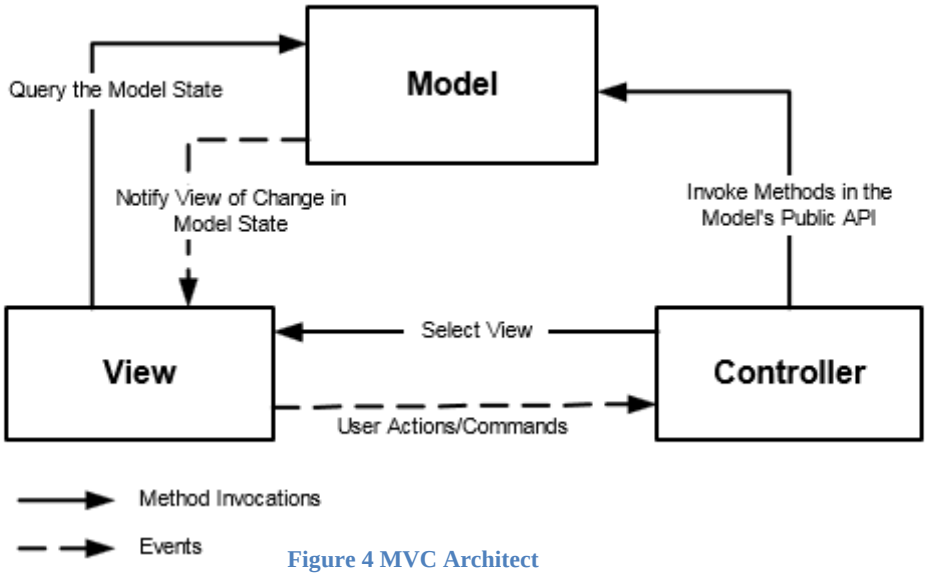

Figure 4 MVC Architect

a. **Model**: The domain-specific representation of the information on which the application operates. The model is another name for the application logic layer (sometimes also called the domain layer). Application (or domain) logic adds meaning to raw data (e.g., calculating if today is the user's birthday, or the totals, taxes and shipping charges for shopping cart items). Many applications use a persistent storage mechanism (such as a database) to store data. MVC does not specifically mention the resource management layer because it is understood to be underneath or encapsulated by the Model.
b. **View**: Renders the model into a form suitable for interaction, typically a user interface element. MVC is often seen in web applications, where the view is the HTML page and the code which gathers dynamic data for the page.
c. **Controller**: Processes and responds to events, typically user actions, and may invoke changes on the model and view.

Microsoft Visual Studio 2010 IDE

Visual Studio is a suite of component-based software development tools and other technologies for building powerful, high-performance applications. In addition, Visual Studio is optimized for team-based design, development, and deployment using Team Foundation Service or Team Foundation Server. The IDE is also improved a lot with the previous version clearing the UI organization and reducing the clutter and complexity. This IDE comes with the .NET Framework 4 and the applications developed are targeted to support windows 7. It was designed to develop the Microsoft Silverlight Applications. This version supports the IBM DB2 and oracle databases including Microsoft SQL Server. Visual Studio 2010 offers several tools to make parallel programming simpler: in addition to the Parallel Extensions for the .NET Framework and the Parallel Patterns Library for native code, Visual Studio 2010 includes tools for debugging parallel applications. The new tools allow the visualization of parallel Tasks and their runtime stacks.

XML mark ups

XML stands for the Extensible Mark-up Language which is the metalanguage that describes the content of the document or in short we can say that it is self-describing data.XML is all about preserving useful information - information that computers can use to be more intelligent about what they do with our data. The best part of XML is that it liberates information from the shackles of a fixed-tag set. XML provides a standard approach for describing, capturing, processing and publishing information. It is a language that has significant benefits over HTML. Unlike most mark-up languages, XML is a flexible framework in which we can create our own customized mark-up languages. All XML-based languages will share the same look and feel and they share a common basic syntax. Some XML based languages already exist in fields such as Push Technologies, Electronic Commerce and Mathematics. XML is used is for the different reasons, some of which are as follows:

- Used in the configuration files
- Used as the media for the data interchange
- Used in B2B transaction on the web
- Web development
- Documentation
- Database development

Altova XMLSpy for XML designs

AltovaXMLSpy is the XML editor and development environment for modelling, editing, transforming, and debugging XML-related technologies. It offers the graphical schema designer, Smart Fix validation, a code generator, file converters, debuggers, profilers, full database integration, support for XSLT, XPath, XQuery, WSDL, SOAP, XBRL, and Office Open XML (OOXML) documents, plus Visual Studio and Eclipse plug-ins, and more. XMLSpy delivers the power we need to create the most advanced XML and Web services applications, yet at the same time it's flexible enough to allow us to work with XML using the views and options that best suit your business needs and working preferences.

Microsoft SQL Server Management Studio

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network. SQL Server Management Studio is an integrated environment for accessing, configuring, managing, administering, and developing all components of SQL Server. SQL Server Management Studio combines a broad group of graphical tools with a number of rich script editors to provide access to SQL Server to developers and administrators of all skill levels.

METHODOLOGY

The process of designing an implementation prototype took place in various phases. The milestones for each phase were set by us internally. The final design occurred in following major phases in overall.

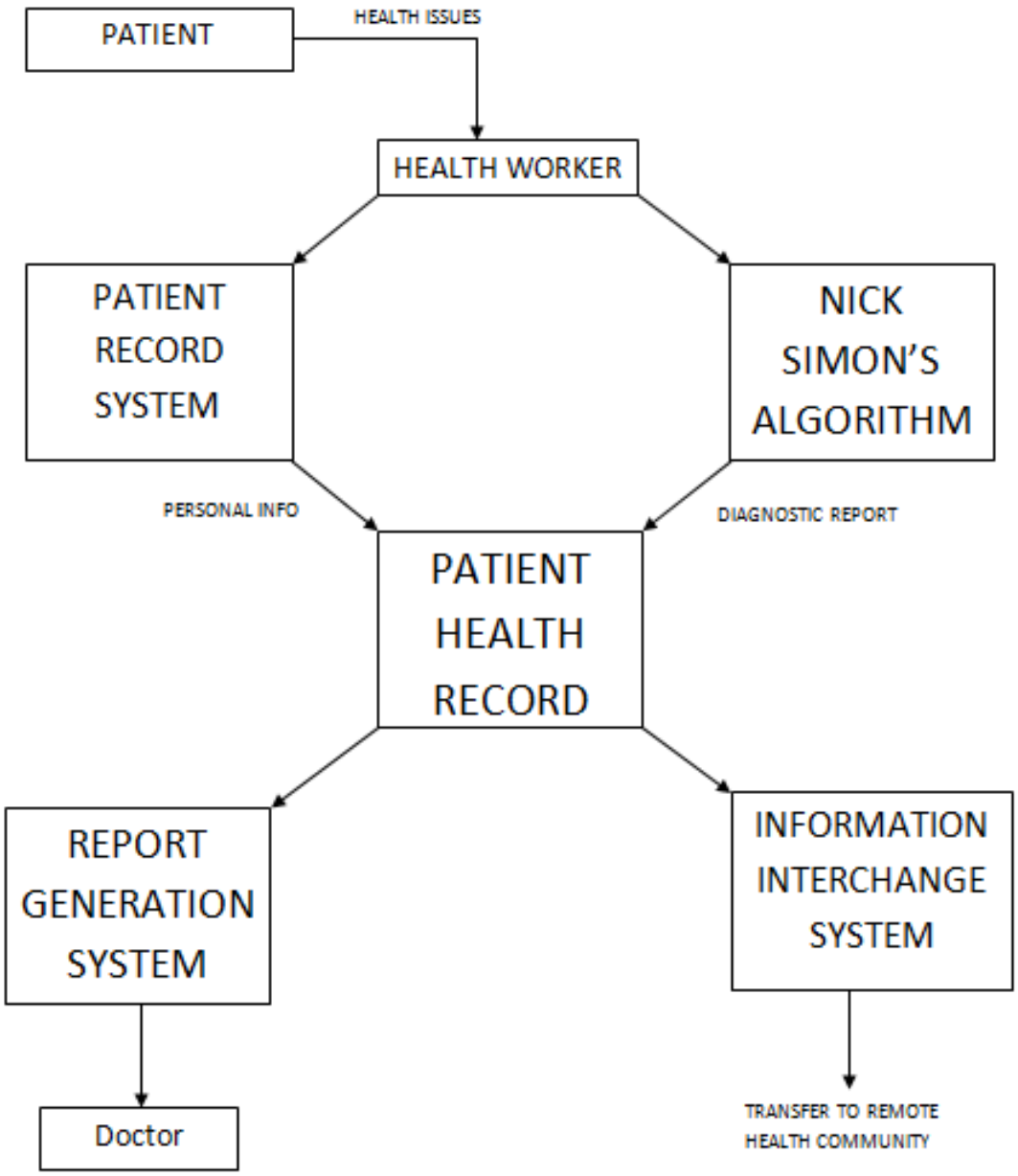

Figure 5 System Block Diagram

Phase-1 Design in sketch flow

After studying the problem domain and specifying the constraints of the conceptual solution, the first version our application came on a sketchflow model. With this model, we were able to correctly identify the program flow and the control flow. This enabled us to design the view, controllers, service layers and models. The sketchflow diagrams were updated and re-designed as we found flaws in the preceding designs. Sketchflow models enabled us to efficiently design the user interfaces and the data repositories. Following shows the snapshot of one of the sketchflow models that we designed.

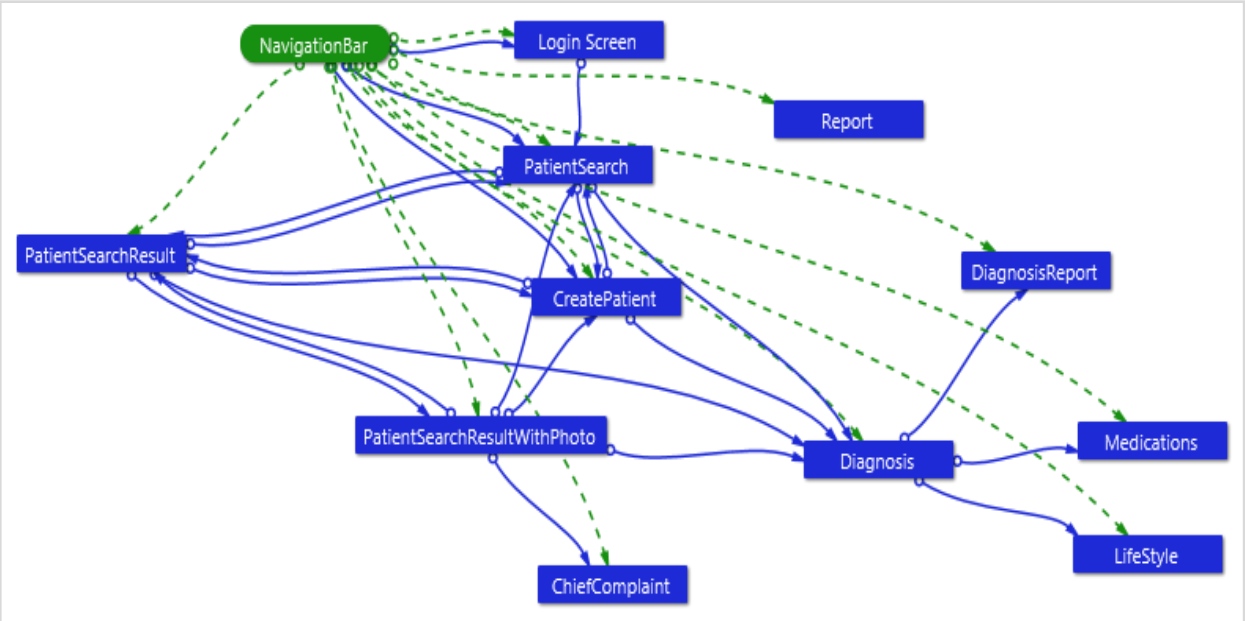

Figure 6 Sketchflow Model of OCDS

Phase-2 UI design

As we obtained a stable sketchflow model, we started to work on designing UI to match the sketchflow model. The UI design was implemented using ASP.NET MVC platform. The UI design process continued here forth to incorporate the communication of the user with the data access layer and the service layer.

Phase-3 XML and database design

Once the preliminary UI was finalized we started to work on the design of the rule base and the database for the system. The rule base for the diagnosis contains the rules from Nick Simon's Algorithm, implemented using XML schemas. The figure below shows some of the elements of XML schema.

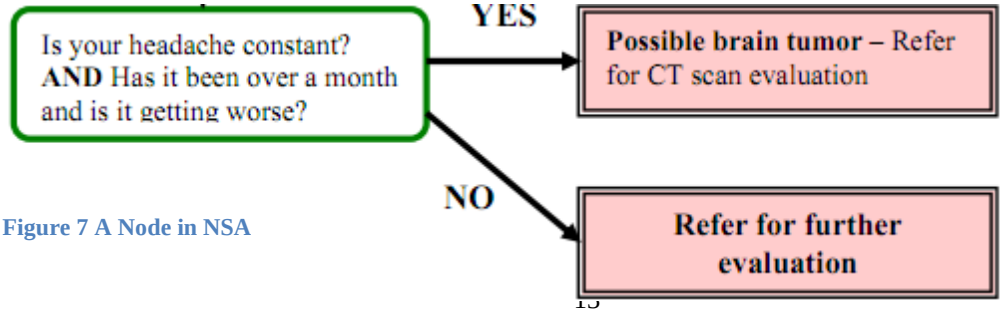

Figure 7 A Node in NSA

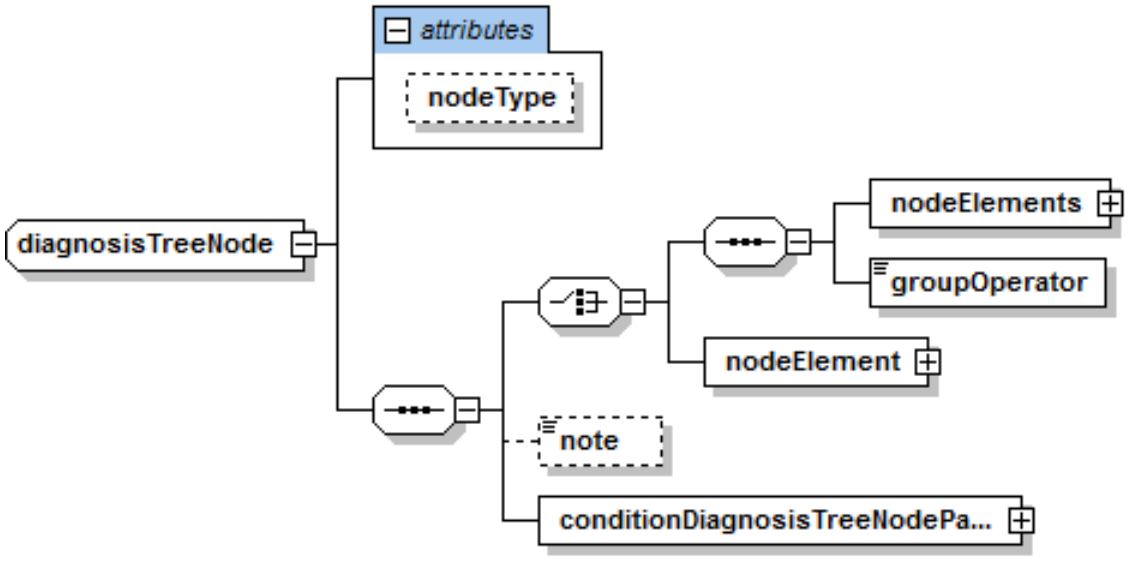

Figure 8 XML Schema for diagnosisTreeNode

Each rectangular box in 5 represents one step of the Clinical Algorithm. Each of them is represented by a *diagnosisTreeNode* in our XML Schema. The schema structure of *diagnosisTreeNode* is shown in 6. Each diagnosis node has a *nodeType* as its attribute. This means that the content of a diagnosis node can represent History of Present Illness, Family History, and Physical Examination and so on. In the first node in 5, we have two statements, separated by AND operator; while the other nodes both have only one statement contained in them. Hence, a diagnosis tree node can have either one *nodeElement*, or have a list of *nodeElements*, along with a *groupOperator* (AND or OR condition) as shown in 6. Also, a *diagnosisTreeNode* has *conditionDiagnosisTreeNodePairs* which maps a condition to the next step, deciding which step to follow when a specific condition matches.

The database of the system was designed to store the details of the results of diagnosis process of a patient. Different tables were created to adjust different details into the database. The figure below shows the database schema of what we have implemented:

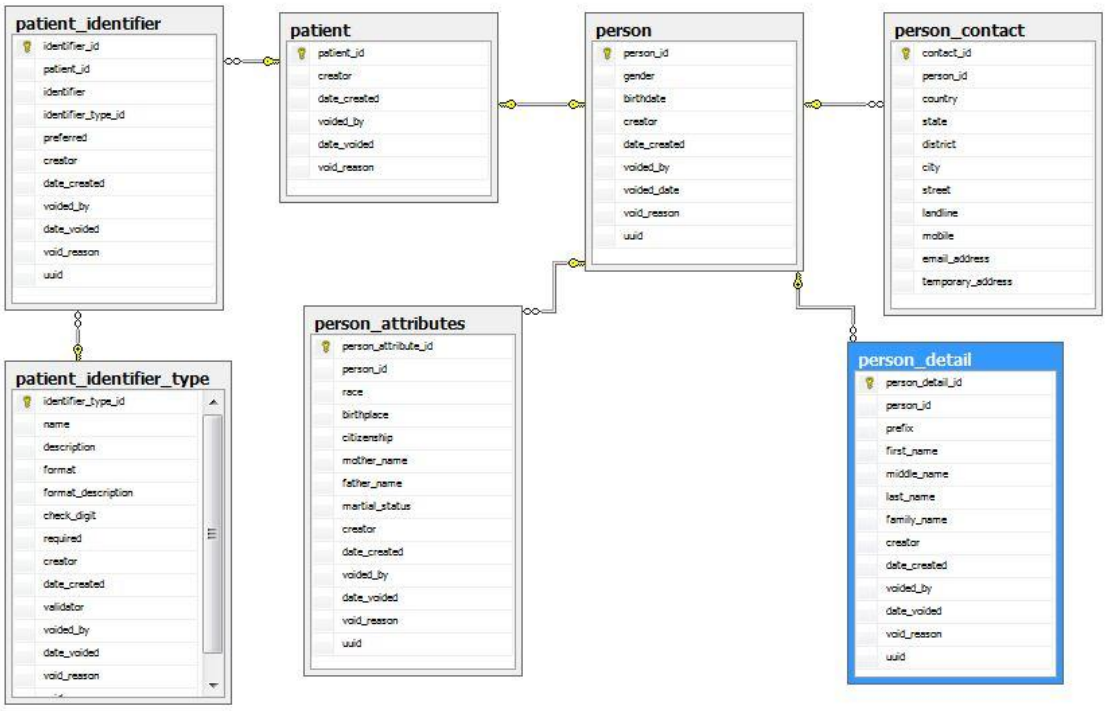

Figure 10 Database Schema for Patient Details Storage

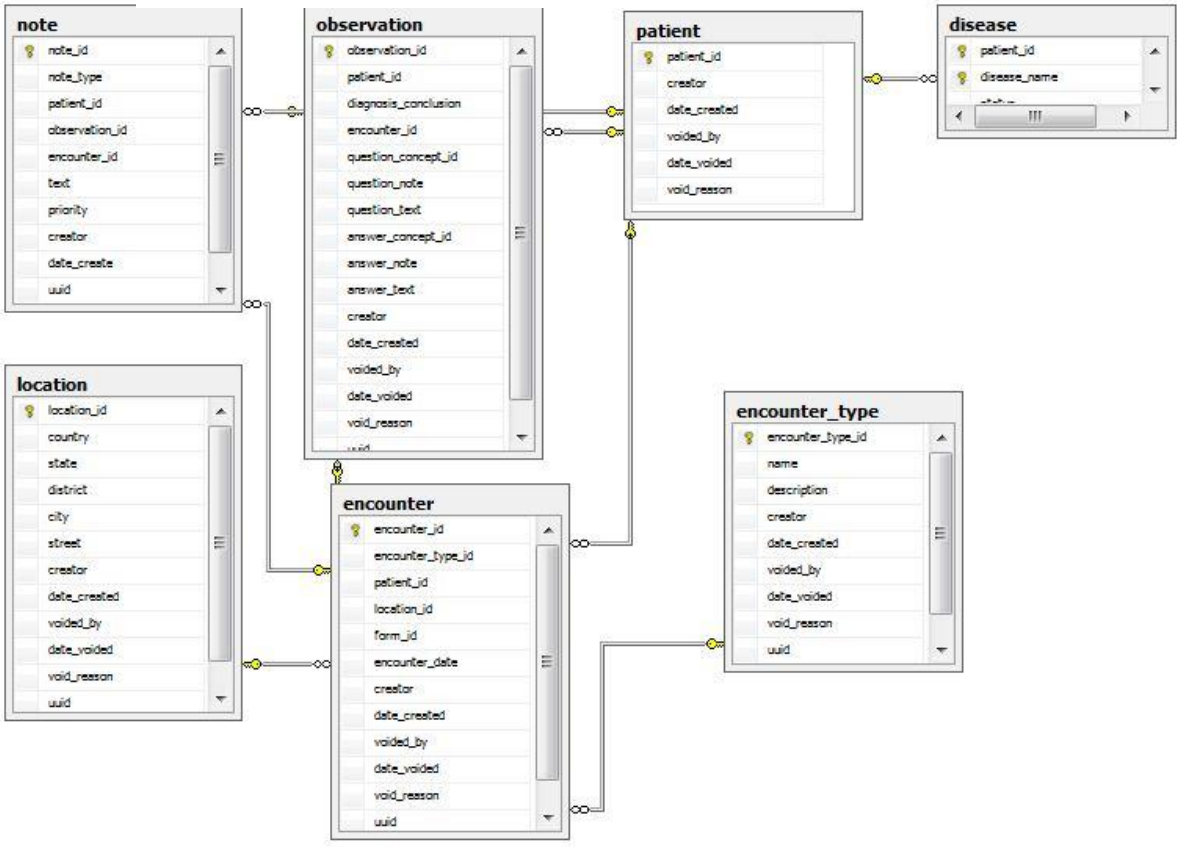

Figure 9 Database Schema for Patient Encounter

Phase-4 Integration

Finally, when each of the phases was taken to the final level, we started to integrate them into a single, self sustaining system. The complete system diagram is shown in the system block diagram above.

DESIGN TECHNOLOGY

Layered Architecture

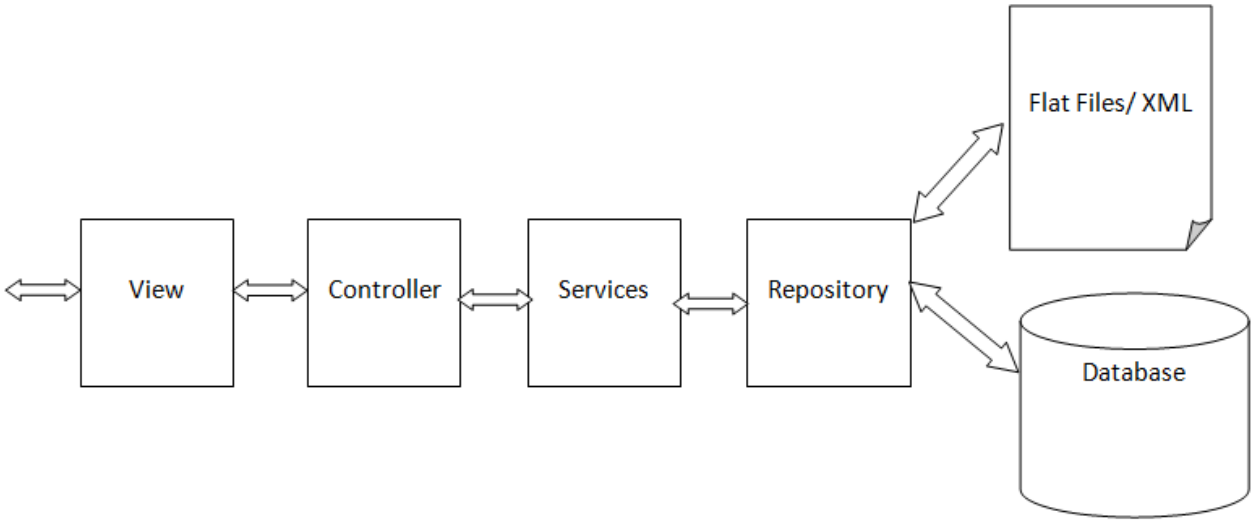

Figure 11 Abstraction of MVC in OCDS

i. **View**:

View is the highest level of abstraction provided by the application. This layer contains all the necessary interfaces, forms and features that users can use to interact with the application. It is the doorway to insert the medical observations and symptoms seen on a patient into the system for analysis and diagnosis process. The user will have nothing to do with the underlying logic of the system. The view captures the chief complaint and based on the chief complaint, requests the controller to load the respective algorithm. Once algorithm is loaded, it presents the questions to the user and reads the answer of the corresponding question. The answers are then sent to the controller. At the end of the diagnosis process, the view presents the final result of the diagnosis. The final result consists of the name of the disease that the person is suffering from.

ii. **Controller**:

Controller is the layer that acts as a bridge between the interface and application. Controller is an action driven layer i.e. it triggers various events and calls various functions depending upon the actions either triggered implicitly by the program or explicitly by the users. When it gets the name of the algorithm sent by the view, it uses the service layer to invoke the first question content of the diagnosis process. It also redirects the answers for each question on the nodeElement to the respective service layer to get processed. The answers when sent to the service layer yields the content of the next diagnosis question. These questions are again redirected to the view by the controller. Controller also exports the data content of current diagnosis process to the database to be stored for future use.

iii. **Services**:

The Service layer consists of all the functions that will operate on the data. Service layer works with repository and controller to transform data. Based on function calls made by actions in controllers, this layer processes the data either provided by the users or that stored in the database (or flat file) and forward it to the next layer for further processing.

The functions of service layer either work on the database or on the flat files. The flat files contain the rules for the diagnosis process. The contents of current diagnosis process are processed by the service layer to find the contents of next diagnosis process. These contents are then located in the respective flat file and the corresponding content set is exported to the controller to be passed to the view.

The data contents of completed diagnosis process are obtained by the service layer and are then exported to be saved into the database.

iv. **Repository:**

Repository is the lowest level of abstraction. It directly communicates with the storage units. Repository consists of all the necessary logic to map the data that are stored in various flat files and database into the classes defined within the application. It also correctly stores data provided by service layer to appropriate database.

When a data content to be interacted with the flat file is sent by the service layer, repository uses the required logic to allow the interfacing. In case the data content corresponds to the content of completed diagnosis process, then the repository allows the access to the database and insures the required operation is carried our properly, that is to say, the content is saved well.

Unit Testing

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Unit testing allows programmers to find out about errors and bugs earlier during development stage and facilitate changes. Unit testing are ideal for projects that implements modular programming.

Reasons for implementing Unit Testing:

Our project comprises of numerous modular elements; each module has independent operations but each module heavily relied on data and events triggered by users. The only way to determine that whether all the modules are in good form and no bugs were present in the code was to implement unit testing. Unit testing allowed us to find out various logical and run-time errors present in our program and debug them.

Implementing Unit Test in our Project:

Unit Tests were written mainly for modules that implemented data validation and data mapping to the database. Writing test allowed us to use in-program data and validate them against the logic of the program and also check whether they were successfully implemented in the database and XML files.

Advantages of Using Unit Testing:

1. **Instant Gratification:** Unit testing allows programmers to execute the code right away and know that it works. This is a very powerful tool. Unit testing allows us to instantly try out your code and see it working.
2. **Code Against Your API Before it is Built:** Unit testing gives us the chance to code against our API/class before it is built. The test then becomes a template for future classes that use the API. They will interact with it the same way that a client class will.
3. **Leads to a Better Design:** Since unit testing should start very early in the process of writing code, it can help programmers to design an API. Since the tests will become templates as to how to use the class under test, we can get a feel for how natural the API feels. This ultimately leads to a better design and helps us to choose good names for methods and also to determine which methods are really needed.
4. **Understand How Our Code Works:** Unit tests really work out the code and tell us how our code will act under many different scenarios. Whether the tests pass or fail, we will learn something about how our code executes.

The code below demonstrates a sample usage of unit testing in our application:

```
using System;
using System.Text;
using System.Collections.Generic;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OCDS.Services.Interfaces;
using OCDS.Services.Implementation;
using OCDS.Domain.Model;
namespace OCDS.DataAccessTest
{
    [TestClass]
    public class DiagnosisServicesTest          //This is the unit test class
    {
        private IDiagnosisService _diagnose = new DiagnosisService();
        [TestMethod]
        public void DiagnosisServiceTest()      //This is the unit test function
        {
            string actual = "D:\Database\fever.xml";
            string algo = _diagnose.GetFilePathByID(1);
            Assert.AreEqual(actual,algo,"Your implementation is incorrect");
        }
    }
}
```

The code snippet is the implementation of Unit Testing. The code below is the code that we are going to test in the above unit test class.

```
public string GetFilePathById(long id)          //This is the function being
tested
{
      string path = entitySet.diagnosis_algorithms
                  .Where(algo => algo.diagnosis_id == id)
                  .Select(question=>question.file_path).Single();
      return path;
}
```

Unit Test Description:

This is one of the sample from number of unit testing codes used in our project. This simple unit testing code called "DiagnosisServicesTest" performs validation of a function called "GetFilePathById".

To check the validity of the code, an object that can access the function (_diagnose) is created in the unit test. This object is used to invoke the function while maintaining the functional integrity and necessary working environment of the function. Once the function is invoked, the output of the function is tested with the desired output and both are asserted to a test. If they meet the requirements and produce no error, the test can be concluded successfully. Otherwise, there may be occurrence of some logical errors or bugs that need to be removed from the code

Data Contracts

Our built of implementation prototype started with analysis of problem domain- medical diagnosis process. We designed the conceptual solution and fixed the objectives of our application. We defined concrete layers to work on with. We defined software classes and assigned attributes and properties to them. We defined how our instances look like and behave in a particular situation. For this, we explicitly defined interfaces for various layers. The implementation classes then followed these signatures. While each one of us was busy with our part of work, we strictly followed these signatures and defined the behaviour of the methods that we defined for all the software classes. Some of the interfaces are shown below:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using OCDS.Domain.DiagnosisAlgorithm.Implementations;

namespace OCDS.Services.Interfaces
{
    public interface IInferenceEngineService
    {
        /// <summary>
        /// Gets the first Diagnosis Node of the given algorithm
        /// </summary>
        /// <param name="algorithmInfo">Diagnosis Algorithm</param>
        /// <returns>first diagnosis node</returns>
        diagnosis_node FindFirstDiagnosisNode(string algorithmFilePath)
        /// <summary>
        /// From the given currentNodeId, find the next DiagnosisNode
        /// </summary>
        /// <param name="algorithmFilePath"></param>
        /// <param name="currentNode"></param>
        /// <returns></returns>
        diagnosis_node FindNextDiagnosisNode(string algorithmFilePath, diagnosis_node
currentDiagnosisTreeNode);
        /// <summary>
        ///
        /// </summary>
        /// <param name="algorithmfilePath"></param>
        /// <param name="currentDiagnosisTreeNode"></param>
        /// <returns></returns>
        diagnosis_node GetReportForCurrentDiagnosis(string algorithmfilePath,
diagnosis_node currentDiagnosisTreeNode);
    }
}
```

While we restricted our designs with the signatures such as above, it became way too easy at the time of integration of individual works to make a single functional implementation prototype. There were no any code conflicts and coding was efficiently completed.

ANALYSIS OF NSA

NSAs (Nick Simon's Algorithms) are based on a set of symptoms driving the diagnosis process towards a reasonable conclusion. Here, we try to analyze and demonstrate how a conclusion is reached in case of a diagnosis algorithm for Headaches as chief complaint.

The lists of diseases that are caused due to headaches are as follows:

- Intracranial haemorrhage
- Cervical Spodilytis
- Brain Tumour
- Meningitis/Typhoid/Malaria
- Influenza/Typhoid/Malaria
- Migraine Headache
- Depression/Poor Vision
- Severe Head Injury
- Moderate Head Injuries
- Head injury with concussions
- Mild Head Injuries

The list of symptoms following the chief complaint 'headache' is as follows:

- The patient has injury in head.
- There is no injury in patient's head.
- The patient has fever
- The patient does not have fever
- Sudden onset of headache
- Not the first time that this kind of headache has happened
- The patient is dizzy, unable to walk or talk
- No dizziness, the patient can walk and talk
- Frequent and irregular headaches.
- Regular and less frequent headaches.
- Pain in the head when moved side to side.
- No pain, numbness or tingling in the head when moved side to side.
- Constant headache for last 1 month and is getting worse.
- No constant headaches
- Loss of consciousness
- No loss of consciousness
- Ongoing loss of consciousness
- Decreased level of consciousness.
- Convulsions/Syncope (fainting).
- Stiffness in the neck
- Constant Vomiting
- First encounter after age 40
- Speech/Movement Impairment
- Headache with a fever and a petechial rash.
- Bothered by light
- Not bothered by light. Nausea with Headaches.

- Nausea and Vomiting.

Sample Diagnosis Process Evaluation

Now, let us consider a patient with the following complaint:

"*...I am having headaches. I think these headaches are coming to me on regular basis and it has got worse in last couple of week. I didn't have any recent head injuries and the headaches are not causing any fever as well I don't think I am going to have paralysis because my spinal chord is good and I don't have problems while I am at the gym or when I am playing....*"

From the description above the facts that we can draw are:

- No head injuries
- No fever
- Headache was not sudden
- Headaches come and go often
- No pain when moving around, playing and jumping
- Headache is constant and has got worse

First league of diseases:

The first fact cuts off all the diseases that are the results of head injuries forming the first league of diseases. These diseases that are not cut off are shown in the chart below:

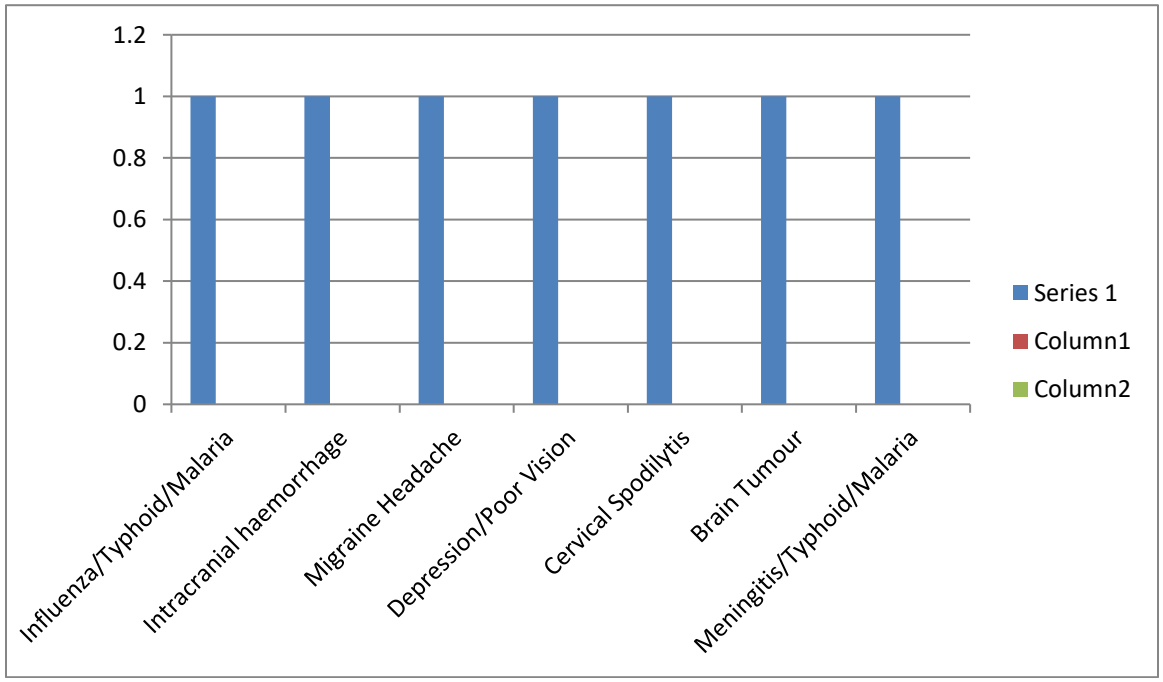

Figure 12 First League of Diseases

All these seven diseases are not caused by head injuries and with the single fact with us, all there diseases are likely to occur.

Second League of Diseases

The fact that the person does not have fever puts on some fluctuations on the occurrence of different diseases. The changes are illustrated as follows:

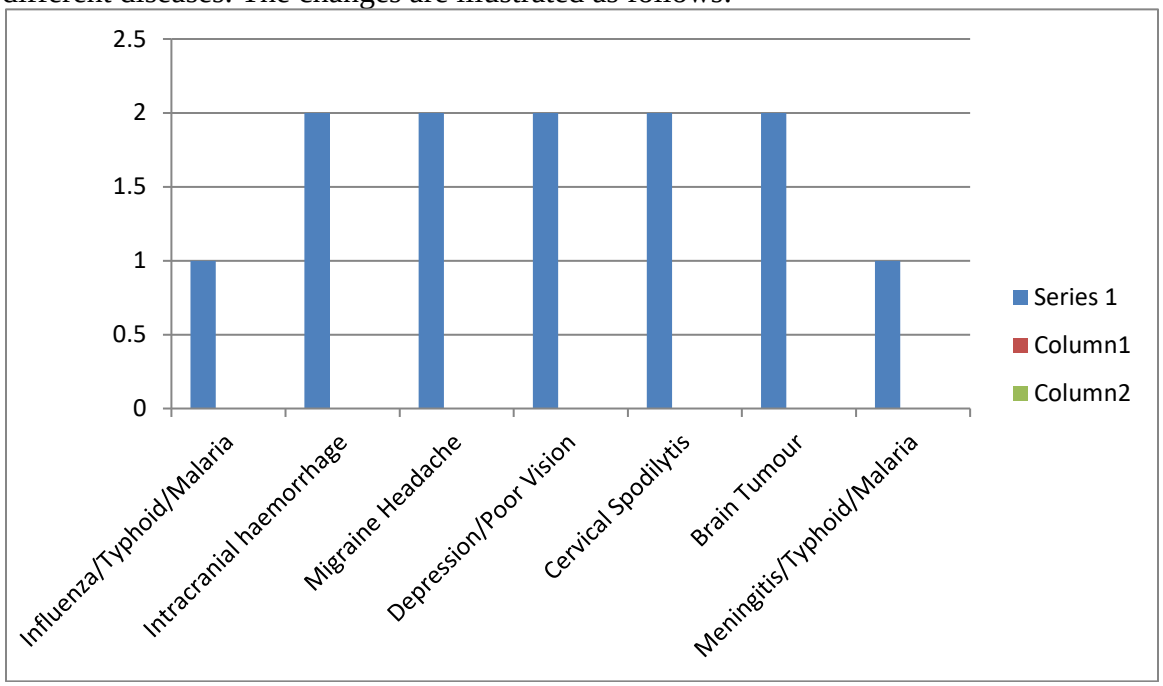

Figure 13 Second League of Diseases

The second league has clearly illustrated that the diseases like Influenza, Typhoid, Malaria, and Meningitis are less likely to occur when a patient does not have headaches due to head injuries and there is no sign of fever.

Third League of Diseases

The fact that the headache was not a sudden onset again changes the scenario giving up new figures to be analyzed. The new league becomes as follows:

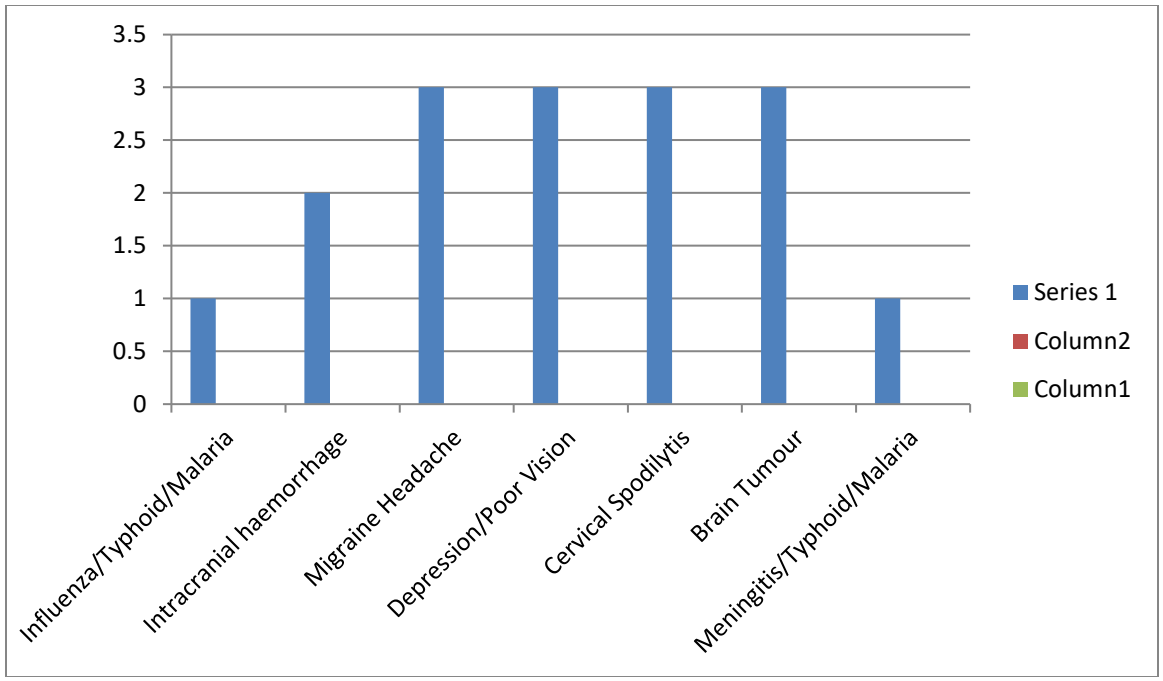

Figure 14 Third League of Diseases

Now, we can see the chances of occurrences of different diseases such as Migraine and Brain tumour has increased while the chance of Intracranial Haemorrhage has stayed the same.

Fourth League of Diseases

The fact that the patient often has the headaches will commence the following changes in the figure above:

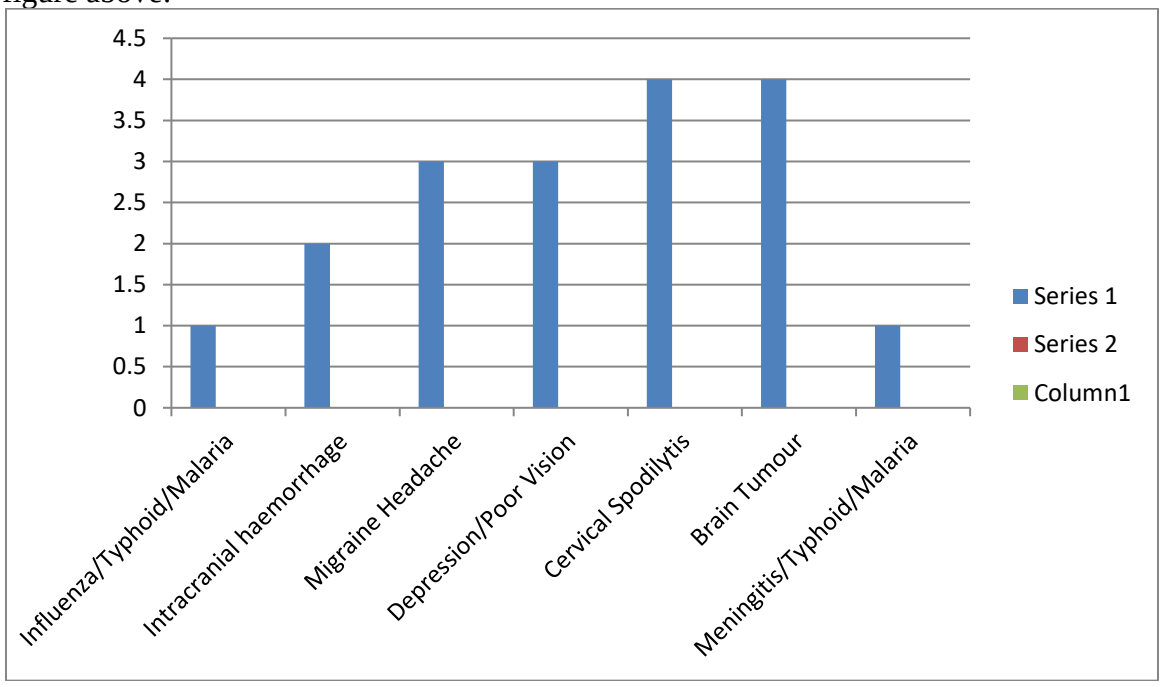

Figure 15 Fourth League of Diseases

Fifth League of Diseases

The fact that the person has no problems in doing physical exercises implies that his neck is not numb and that he has been having constant headaches. This fact provides a new context to the diagnosis, modifying the table as follows:

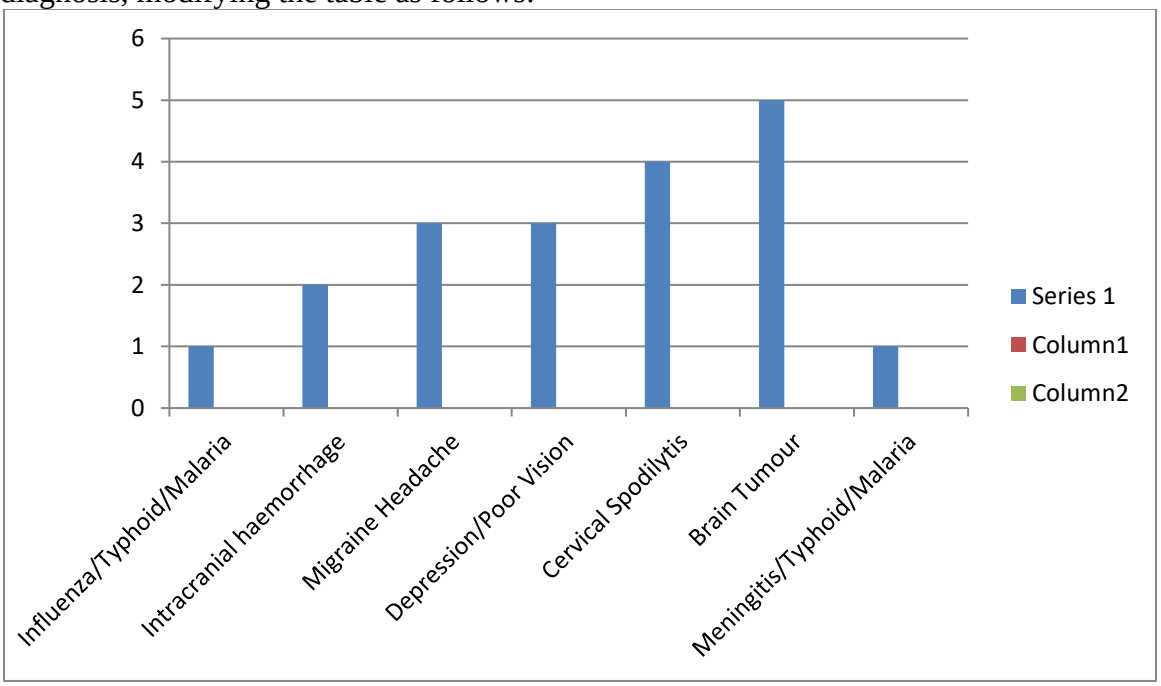

Figure 16 Fifth League of Diseases

Coming to the fifth league the NSA has concluded that the patient has brain tumour. It could appear to the reader that the diagnosis is more of a probability game, but since the medical diagnosis process is all about the symptoms and observations made on the patient, these symptoms form a foundation to take a decision regarding the diagnosis of the patient. These algorithms have been tested and have been proven to be greatly useful and highly accurate. Till date, there implementation of these algorithms have been recorded to be only based on papers and tradition equipments. We have tried to generate a new approach to facilitate the diagnosis process. This is how our application reaches to the diagnosis from chief complaint.

FUTURE EXTENDIBILITY

While on this project, we designed a simple module of implementation of diagnosis using NSA. This diagnosis system has following possible areas of extension:

- Implementation of system over the web, that is run able both online and offline, is one possible are of extension of our application. The system can be switched to online mode to update the rule base and the data base of the system. While the system is in offline mode, it is capable of providing complete diagnosis on the list of chief complaints it has in its rule base.
- Design of Data Synchronization System for the application that can run both in online and offline mode can also be done. This provides a easy update of the system with the newer rule bases.
- The report that our system currently generates contains only the details of the results obtained in the single diagnosis process. The diagnosis results can be further analyzed to generate reports according to the requirement of various units of a Health Care Institution. There reports are generated based on a set of constraints that are specified the respective unit of the Health Care Institution
- The reports and diagnosis details of the person can be transmitted over the web using standard EDIs such as HL7 or EDMX. For these, the protocols pre-specified by the messaging standards need to be followed. HL7 and EDMX provide documentations for the protocols and these protocols are revised every now and then.
- The diagnosis module of the system alone can be integrated to OpenMRS system to facilitate more efficient data storage systems. The storage of LEMRs provided by OpenMRS is popular and widely implemented, in more than 130 countries. If the system is implemented together with OpenMRS, a large number of medical institutions, especially in the underdeveloped countries, will get benefitted with the diagnosis system.

CONCLUSION

The work that our team did together in course of the completion of our Minor Project OCDS brought us together as a team. The implementation of diagnosis system in the context of country like ours needs to be taken with greater interest. There are a lot of people being deprive from effective health services and our application has the potentiality to fulfil the promise. The usage of computer system design for health can benefit a lot of people.

We aim to take this initiation of ours to a logical conclusion. We are ready to work further on this project, if we get proper resources and a good working environment. With the implementations of above specified extensions, Outreach Clinical Diagnosis System has the potentiality to advent a new era of e-Health where people are diagnosed by computer and expert system. Then no one would have to solely depend on qualified doctors to get better health services. Our intelligent system is the doctor of the next generation.

REFERNCES

Margolis, C.Z., et al., Clinical algorithms teach pediatric decision making more effectively than prose. Med Care, 1989. 27(6): p. 576-92.

Lowery, J.C., et al., *Comparison of professional judgment versus an algorithm for nutrition* status classification. Med Care, 1998. 36(11): p. 1578-88.

Neale, E.J. and A.M. Chang, *Clinical algorithms.* Med Teach, 1991. **13**(4): p. 317-22.

Ambrose, R.F., et al., *Rheumatology algorithms for primary care physicians.* Arthritis Care Res, 1990. **3**(2): p. 71-7.

http://www.layoutgalaxy.com/xml/intro.php4
http://www.w3.org/XML/
http://msdn.microsoft.com/en-us/library/ms174173.aspx
http://msdn.microsoft.com/en-us/library/hh213248.aspx
http://msdn.microsoft.com/en-us/centrum-asp-net.aspx
http://www.asp.net/mvc/tutorials/getting-started-with-aspnet-mvc3/cs/intro-to-aspnet-mvc-3