

1. INTRODUCTION

1.1 OBJECTIVE

COLLABORATIVE INFORMATION SYSTEMS (CISs) allow groups of users to communicate and cooperate over common tasks. They have long been called upon to support and coordinate activities related to the domain of “computer supported and cooperative work”. Recent breakthroughs in networking, storage, and ubiquitous computing have facilitated an explosion in the deployment of CIS across a wide range of environments.

Beyond computational support, the adoption of CIS has been spurred on by the observation that such systems can increase organizational efficiency through streamlined workflows, shave administrative costs, assist innovation through brainstorming sessions, and facilitate social engagement. On the Internet, for instance, the notion of CIS is typified in wikis, video conferencing, document sharing and editing, as well as dynamic bookmarking. At the same time, CIS are increasingly relied upon to manage sensitive information.

Intelligence agencies, for example, have adopted CIS to enable timely access and collaboration between groups of analysts using data on personal relationships, financial transactions, and surveillance activities. Additionally, hospitals have adopted electronic health record (EHR) systems to decrease healthcare costs, strengthen care provider productivity, and increase patient safety, using vast quantities of personal medical data. However, at the same time, the detail and sensitive nature of the information in such CIS make them attractive to numerous adversaries. This is a concern because the unauthorized dissemination of information from such systems can be catastrophic to both the managing agencies and the individuals (or organizations) to whom the information corresponds. It is believed that the greatest security threat to information systems stems from insiders. In this work, we focus on the insider threat to centralized CIS which are managed by a sole organization.

A suspicious insider in this setting corresponds to an authenticated user whose actions run counter to the organization’s policies. Various approaches have been developed to address the insider threat in collaborative environments. Formal access control frameworks, for instance, have been adapted to model team and contextual scenarios recognizing that access control is necessary, but not sufficient to guarantee protection, anomaly detection methods have been proposed to detect deviations from expected behavior.

Detecting Anomalous Insiders in Collaborative Information Systems

In particular, certain data structures based on network analysis have shown promise. But wish to highlight several limitations of these approaches up front.

First, access control models assume a user's role (or their relationship to a group) is known a priori. However, CIS often violate this principle because teams can be constructed on the fly, based on the shifting needs of the operation and the availability of the users. Second, the current array of access control and anomaly detection methods tend to neglect the Meta information associated with the subjects.

2. LITERATURE SURVEY

2.1 RELATED WORK

Preventing Information Leakage between Collaborating Organizations

Imad M. Abbadi Group Technology Services

Information sharing and protection against leakage is a critical problem especially for organizations' having sensitive information. Sharing content between individuals in the same organization extends to exchanging and sharing content between collaborating organizations.

In this paper we

Propose a novel solution for preventing shared information between collaborating organizations from getting leaked to unauthorized users inside the destination organization or outside it. In addition, once the content is in the hands of authorized users our solution prevents unethical authorized users from leaking such content to other users in the same organizational third parties.

In this paper we provide a mechanism for a source organization to send content to another collaborating organization in such a way the sent content is either accessed by a specific group of users performing a specific task or it could be accessed by all devices member in the destination organization, which should be based on organization policy and requirements.

In the proposed solution we used trusted computing technology to provide a hardware-based root of trust for the master controller and organization devices.

ROLE PREDICTION USING Electronic MEDICAL RECORD SYSTEM AUDITS

EXTENDED ABSTRACT BY WEN ZHANG¹, CARL A.

There are two dominant strategies for limiting access to Electronic Medical Records (EMRs) within enterprises such as hospitals. One strategy, known as Role Based Access Control (RBAC) [SandhuCFY96], groups access privileges into collections called roles and then assigns users to roles to determine their access privileges. This is commonly achieved by reviewing the job positions in the enterprise and the tasks the employees in these positions need to perform, then assigning privileges to positions, or variants of them, to enable the employees to do their assigned tasks. A second strategy, which we group under the general heading of Experience Based Access Management (EBAM) [Gunter], emphasizes accountability and the use of audit data to reprimand abuse. An often referenced strategy for EBAM is to manually

review audit logs of VIPs to determine infractions. Another strategy, called “break-the-glass” security, discourages abuse by warning users that certain types of access are manually reviewed.

However, at the current point in time, RBAC and EBAM are used without much common foundation. This is unfortunate because there seems to be significant opportunities for synergy between the techniques. For example, audit data may provide valuable information about roles, such as whether a new role would be beneficial or whether two existing roles should be merged. On the other hand, auditing analytics can show how more appropriate definitions for roles, or roles that are context-specific, may be applied to restrict access so that fewer checks are required on audits.

Purpose Based Access Control for Privacy Protection in Relational Database Systems

-Won Byun, Ninghui Li

In this article, we present a comprehensive approach for privacy preserving access control based on the notion of purpose. In our model, purpose information associated with a given data element specifies the intended use of the data element. A key feature of our model is that it allows multiple purposes to be associated with each data element and also supports explicit prohibitions, thus allowing privacy officers to specify that some data should not be used for certain purposes. An important issue addressed in this article is the granularity of data labeling, that is, the units of data with which purposes can be associated. We address this issue in the context of relational databases and propose four different labeling schemes, each providing a different granularity.

We also propose an approach to representing purpose information, which results in low storage overhead, and we exploit query modification techniques to support access control based on purpose information. Another contribution of our work is that we address the problem of how to determine the purpose for which certain data are accessed by given user. Our proposed solution relies on Role-Based Access Control (RBAC) models as well as the notion of conditional role which is based on the notions

Collaborative information seeking:

Collaborative information seeking (CIS) is a field of research that involves studying situations, motivations, and methods for people working in collaborative groups for information seeking projects, as well as building systems for supporting such activities. Such projects often involve information searching or information retrieval (IR).

3. PROBLEM DEFINITION

3.1 ANALYSIS MODEL

The model that is basically being followed is the WATER FALL MODEL, which states that the phases are organized in a linear order. First of all the feasibility study is done. Once that part is over the requirement analysis and project planning begins. If system exists one and modification and addition of new module is needed, analysis of present system can be used as basic model.

The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model the sequence of activities performed in a software development project are: -

Requirement Analysis, Project Planning, System design, Detail design, Coding, Unit testing, System integration & testing

Here the linear ordering of these activities is critical. End of the phase and the output of one phase is the input of other phase. The output of each phase is to be consistent with the overall requirement of the system. Some of the qualities of spiral model are also incorporated like after the people concerned with the project review completion of each of the phase the work done.

WATER FALL MODEL was being chosen because all requirements were known beforehand and the objective of our software development is the computerization/automation of an already existing manual working system.

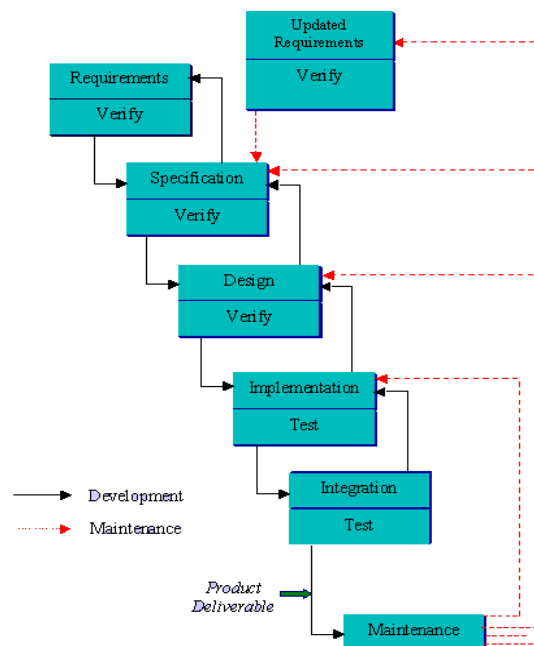


Fig 3.1: Water Fall

Model

3.2 EXISTING SYSTEM

It can be seen that the performance of the supervised classification models is significantly worse than the unsupervised models. The supervised models consistently have a lower true positive rate at all operating points. Second, unlike the previous experiment, HVU achieves comparable results to the supervised classification models. This is due to the fact that this model is correctly characterizing the intruders that access a larger number of records. Third, with respect to AUC, we observe the same trend as earlier regarding the dominance of the unsupervised models as a function of the mix rate. Specifically,

MetaCADS dominates when the mix rate is low, but CADS dominates when the mix rate is high. Notably the disparity between MetaCADS and CADS is more pronounced at the low mix rate (0.91 versus 0.88) in this setting than in the previous setting. However, at lower false positive operating points, CADS appears to dominate MetaCADS.

Disadvantages:

- 1) Supervised models consistently have a lower true positive rate at all operating points.
- 2) With respect to AUC, we observe the same trend as earlier regarding the dominance of the unsupervised models as a function of the mix rate
- 3) CADS dominates only when the mix rate is high

3.3 PROPOSED SYSTEM

Several notable approaches have been proposed to address this type of intruder. The first is nearest neighbor anomaly detection techniques, which are designed to measure the distances between instances by assessing their relationship to “close” instances. If the instance is not sufficiently close, then it may be classified as an anomaly. However, social structures in a CIS are not explicitly defined and need to be inferred from the utilization of system resources. If distance measurement procedures are not tuned to the way in which social structures have been constructed, the distances will not represent the structures well. Our experimental results confirm this notion.

The second approach is based on spectral anomaly detection. This approach estimates the principal components from the covariance matrix of the training data of “normal” events. The testing phase involves the comparison of each point with the components and assigning an anomaly score based on the point’s distance.

The model can reduce noise and redundancy, however, collaborative systems are team oriented, which can deteriorate performance of the model as our experiments demonstrate.

Advantages:

- 1) Proposed System Consistently Have a Higher True Positive Rate at All Operating Points
- 2) HVU Achieves More Comparable Results When Compared To Supervised Models
- 3) MetaCADS dominates when the mix rate is low, but CADS dominates when the mix rate is high

a. PROCESS DIAGRAM

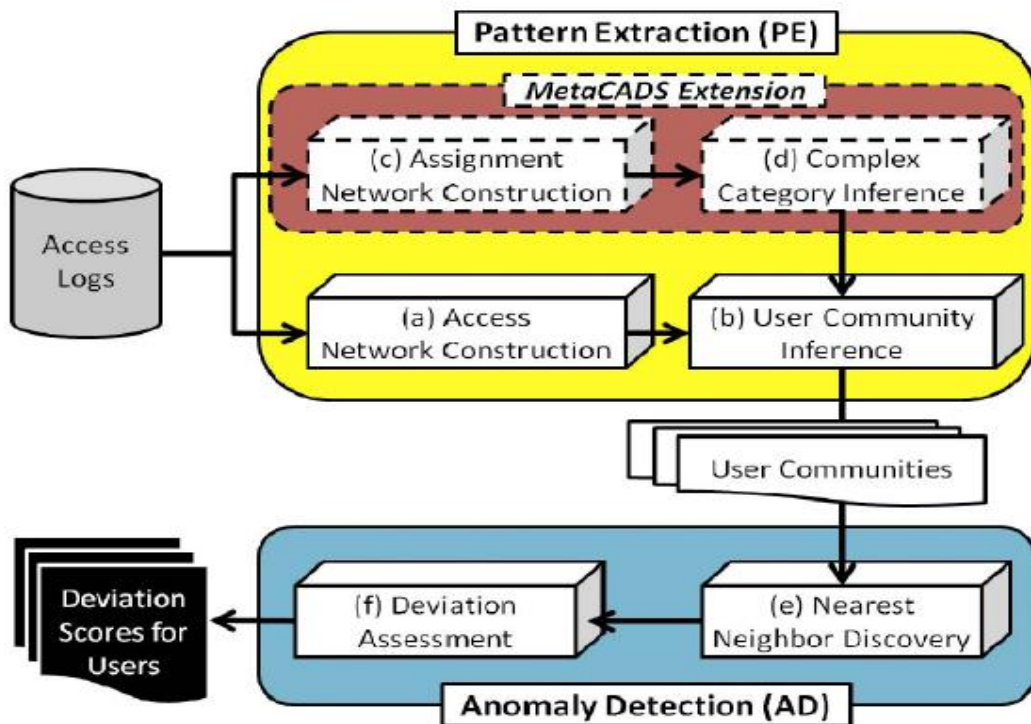


Fig 3.2: Process Diagram

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

3.4 SYSTEM DETAILS

The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution

In the flexibility of the user the interface has been developed a graphics concept in mind, associated through a browser interface. The GUI'S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The interfaces help the administrations with all the transactional states like Data insertion, Data deletion and Date updating along with the extensive data search capabilities.

The operational or generic user interface helps the users upon the system in transactions through the existing data and required services.

3.5 FEASIBILITY STUDY

Feasibility Study is a high level capsule version of the entire process intended to answer a number of questions like: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem clearly understood. Feasibility study is necessary to determine that the proposed system is Feasible by considering the technical, Operational, and Economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system.

The following feasibilities are considered for the project in order to ensure that the project is variable and it does not have any major obstructions. Feasibility study encompasses the following things:

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

In this phase, we study the feasibility of all proposed systems, and pick the best feasible solution for the problem. The feasibility is studied based on three main factors as follows

3.5.1 TECHNICAL FEASIBILITY

In this step, I verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not.

Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds.

- All necessary technology exists to develop the system.
- This system is too flexible and it can be expanded further.
- This system can give guarantees of accuracy, ease of use, reliability and the data security.
- This system can give instant response to inquire.
- Our project is technically feasible because, all the technology needed for our project is readily available.

3.5.2 ECONOMICAL FEASIBILITY

In this step, I verify which proposal is more economical. We compare the financial benefits of the new system with the investment. The new system is economically feasible only when the financial benefits are more than the investments and expenditure. Economic Feasibility determines whether the project goal can be within the resource limits allocated to it or not. It must determine whether it is worthwhile to process with the entire project or whether the benefits obtained from the new system are not worth the costs. Financial benefits must be equal or exceed the costs. In this issue, we should consider:

- The cost to conduct a full system investigation.
- The cost of hardware and software for the class of application being considered.
- The development tool.
- The cost of maintenance etc.,

My project is economically feasible because the cost of development is very minimal when compared to financial benefits of the application.

3.5.3 OPERATIONAL FEASIBILITY:-

In this step, I verify different operational factors of the proposed systems like man-power, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibility determines if the proposed system satisfied user objectives could be fitted into the current system operation. The present system Enterprise Resource Information System can be justified as Operationally Feasible based on the following grounds.

- The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.
- The clients have been involved in the planning and development of the system.
- The proposed system will not cause any problem under any circumstances.

Our project is operationally feasible because the time requirements and personnel requirements are satisfied. We are a team of four members and we worked on this project for three working months.

Project Instructions:-

- Based on the solution requirements, conceptualize the Solution Architecture. Depict the various architectural components, show interactions and connectedness and show internal and external elements. Discuss suitability of typical architectural types like Pipes, Filters, Event Handlers, and Layers etc.
- Identify the significant class entities and carry out class modeling.
- Carry out Detailed design of Classes, Database objects and other solution components.
- Distribute work specifications and carry out coding and testing

4. SYSTEM REQUIREMENTS & ANALYSIS

Software Requirements Specification plays an important role in creating quality software solutions. Specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation.

Requirements may be specified in a variety of ways. However there are some guidelines worth following: -

- Representation format and content should be relevant to the problem
- Information contained within the specification should be nested
- Diagrams and other notational forms should be restricted in number and consistent in use.
- Representations should be revisable.

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to the software as a part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, and indication of performance requirements and

Design constraints, appropriate validation criteria and other data pertinent to requirements.

4.1 SOFTWARE REQUIREMENT SPECIFICATIONS

4.1.1 FUNCTIONAL REQUIREMENTS

This section contains specification of all the functional requirements needed to develop this module or sub-module.

ID	Requirements
DAI_R_01	System should provide a provision to authenticate Admin Login.
DAI_R_02	System should provide a provision to the admin to View agent reports
DAI_R_03	System should provide a provision to the admin to Create new patient

DAI_R_04	System should provide a provision to the admin to Create new doctor.
DAI_R_05	System should provide a provision to the admin to Create new agent
DAI_R_06	System should provide a provision to the admin to View patient details.
DAI_R_07	System should provide a provision to the agent to register the new patient name.
DAI_R_08	System should provide a provision to the doctor to enter the patient all details.
DAI_R_09	System should provide a provision to generate secrete key
DAI_R_10	System should provide a provision to click on department by the doctor
DAI_R_11	System should provide a provision to the agent to view patient details
DAI_R_12	System should provide a provision to the agent to view doctor details.

Table 4.1: functional requirement table

4.1.2 NON FUNCTIONAL REQUIREMENTS

Performance Requirements:

Good band width, less congestion on the network. Identifying the shortest route to reach the destination will all improve performance.

Safety Requirements:

No harm is expected from the use of the product either to the OS or any data.

Product Security Requirements:

The product is protected from un-authorized users from using it. The system allows only authenticated users to work on the application. The users of this system are organization and ISP administrator.

Software Quality Attributes:

The product is user friendly and its accessibility is from the client. The application is reliable and ensures its functioning maintaining the ISP web service is accessible to the various organizations. As it is developed in .Net it is highly interoperable with OS that have provided support for MSIL (Server side). The system requires less maintenance as it is not installed on the client but hosted on the ISP. The firewall, antivirus protection etc. is provided by the ISP.

4.2 SOFTWARE REQUIREMENTS

Operating System	: Windows XP or 7
Language	: C#
Tool	: Visual Studio 2005 or latest
Data Base	: SQL Server 2005

4.3 HARDWARE REQUIREMENTS

System	: Pentium IV and above
Hard Disk	: 50 GB and above
Monitor	: VGA Colours
Ram	: 1GB and above

5. SYSTEM DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analyzed, system design is the first of the three technical activities - design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

5.1 ARCHITECTURE DIAGRAM:

"Tier" can be defined as "one of two or more rows, levels, or ranks arranged one above another". 2-Tier Architectures supply a basic network between a client and a server. For example, the basic web model is a 2-Tier Architecture. A web browser makes a request from a web server, which then processes the request and returns the desired response, in this case, web pages. This approach improves scalability and divides the user interface from the data layers. However, it does not divide application layers so they can be utilized separately. This makes them difficult to update and not specialized. The entire application must be updated because layers aren't separated.

5.1.1 High Level Design

System Design:-

Understanding bigger application with its external interfaces is called System Design.

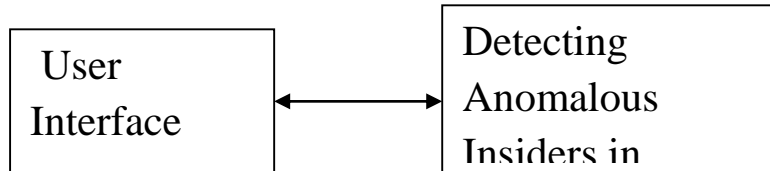


Fig 5.1: System Design

Sub system design:-

Understanding bigger system into smaller independent working systems is called subsystem design.

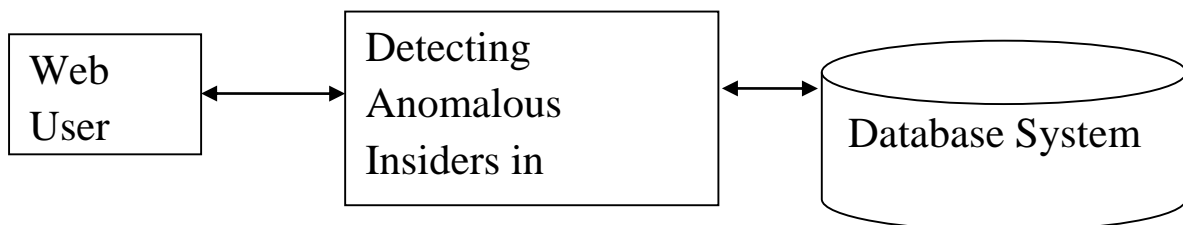


Fig 5.2: Sub System Design

Blocks Design:

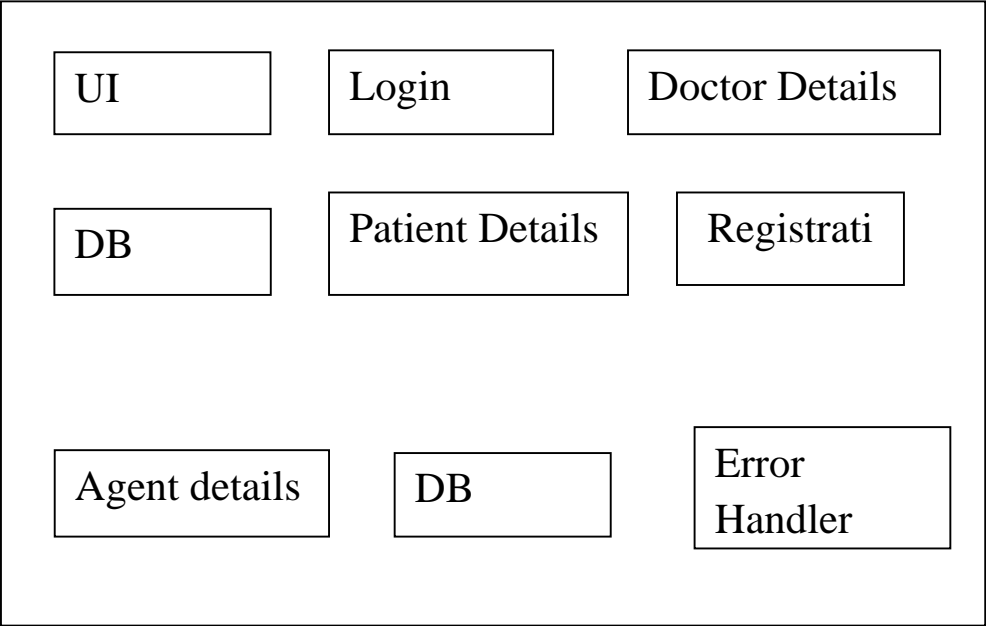


Fig 5.3: Block Design

5.2 MODULES

1 .Pattern Extraction:

CADS-PE infers communities from the relationships observed between users and subjects' records in the CIS access logs. The community extraction process consists of two primary steps: construction of the user-subject access network and user community inference. There are various aspects of a user's relationship to subjects that could be leveraged for measuring similarity. To mitigate bias and develop a generic approach, we focus our attention on the number of subjects a user accessed.

2 .Anomaly Detection:

CADS-AD predicts which users in the CIS are anomalous by discovering a user's nearest neighbors and calculating the deviation of each user from their neighbors. There are alternative anomaly detection models that have been proposed in the literature. Thus, in addition to CADS and MetaCADS, we evaluate three of the most related models. The first two are based on supervised classification and assume there exists a training set of anomalous and no anomalous user class labels, whereas the final model is an unsupervised heuristic. For each of these models, we treat real and simulated users as no anomalous and anomalous, respectively.

K-nearest neighbors. This model predicts the label for a user based on their k-nearest neighbors in the training set. The labels are weighted based on the cosine similarity of each neighbor to the user. For this work, we measure similarity via the vectors of the AI matrix.

Principle components analysis (PCA). This model predicts if a user is closer to normal or abnormal users according to the weighted principal components model. The components are derived from the AI matrix. **High volume users (HVUs).** This model is based on a rule invoked by privacy officials at several healthcare providers. It ranks users based on the number of subjects they accessed. The greater the number of subjects accessed, the higher the rank.

3 .Detection Performance Metrics:

We measure the performance of the models using the Receiver Operating Characteristic (ROC) curve. This is a plot of the true positive rate versus false positive rate for a binary classifier as its discrimination threshold is varied. The area under the ROC curve (AUC) reflects the relationship between sensitivity and specificity for a given test. A higher AUC indicates better overall performance. In the final two simulation settings, we report on the average AUC per simulation configuration.

4. Varying Number of Accessed Subjects:-

The first set of experiments focus on the sensitivity of anomaly detection models. To begin, we mixed a single simulated user with the real users. We varied the number of subjects accessed by the simulated user to investigate how volume impacts the deviation score and the performance of the anomaly detection models in general. For illustration, the MetaCADS and CADS deviation scores for the simulated users in the EHR data

5.3 UNIFIED MODELING LANGUAGE

Unified Modeling Language

- The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
- A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

User Model View:

- This view represents the system from the user's perspective.
- The analysis representation describes a usage scenario from the end-users perspective.

Structural model view:

- In this model the data and functionality are arrived from inside the system.
- This model view models the static structures.

Behavioral Model View:

- It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View:

- In this the structural and behavioral as parts of the system are represented as they are to be built.

Environmental Model View:

- In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.
- UML Analysis modeling, which focuses on the user model and structural model views of the system
- UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

The primary goals in the design of the UML were as follows:-

1. Provide users ready-to-use, expensive visual modeling languages so they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher level development concepts.
7. Integrate best practices and methodologies.

UML is a language used to:

“Visualize” the software system well-defined symbols. Thus a developer or tool can unambiguously interpret a model written by another developer, using UML

- “Specify the software system and help building precise, unambiguous and complete models.
- “Construct” the models of the software system that can directly communicate with a variety of programming languages.
- “Document” models of the software system during its development stages.

5.3.2 UML Diagrams

Every complex system is best approached through a small set of nearly independent views of a model; no single viewer is sufficient. Every model may be expressed at different levels of fidelity. The best models are connected to reality. The UML defines nine graphical diagrams.

1. Class diagram
2. Object diagram
3. Use-case diagram
4. Sequence diagram
5. Collaboration diagram
6. Activity diagram
7. ER diagram
8. State Chart diagram
9. Component diagram
10. Deployment diagram
11. Dataflow diagrams

Use Case Diagram:

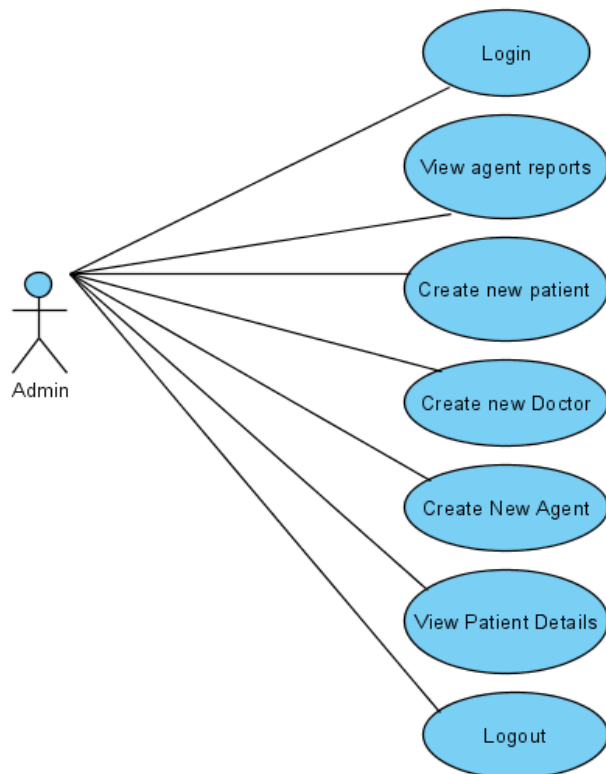


Fig 5.4: use case diagram of Admin

Agent:

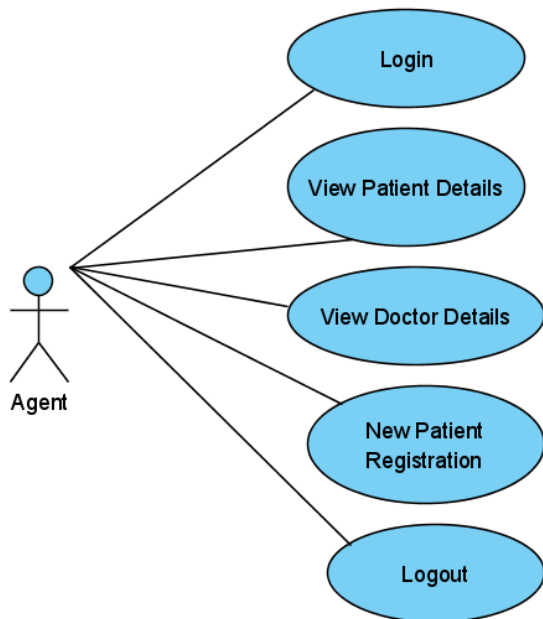


Fig 5.5: use case diagram of agent

Doctor:

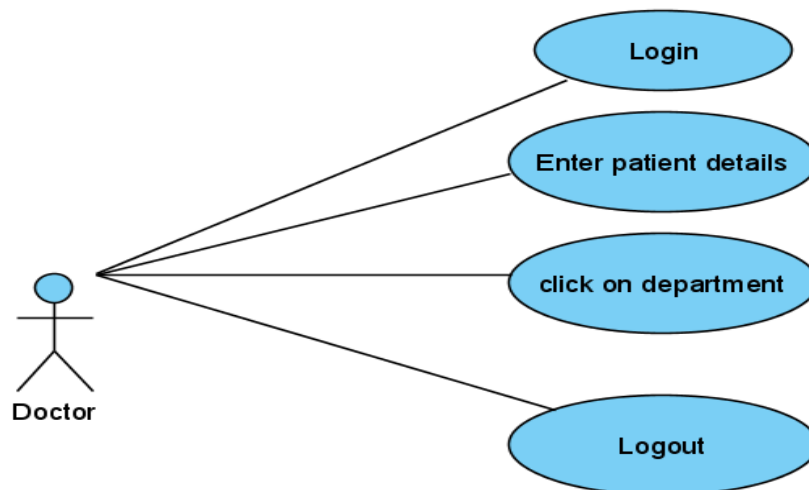


Fig 5.6 use case diagram of doctor

Sequence Diagrams:

Agent:

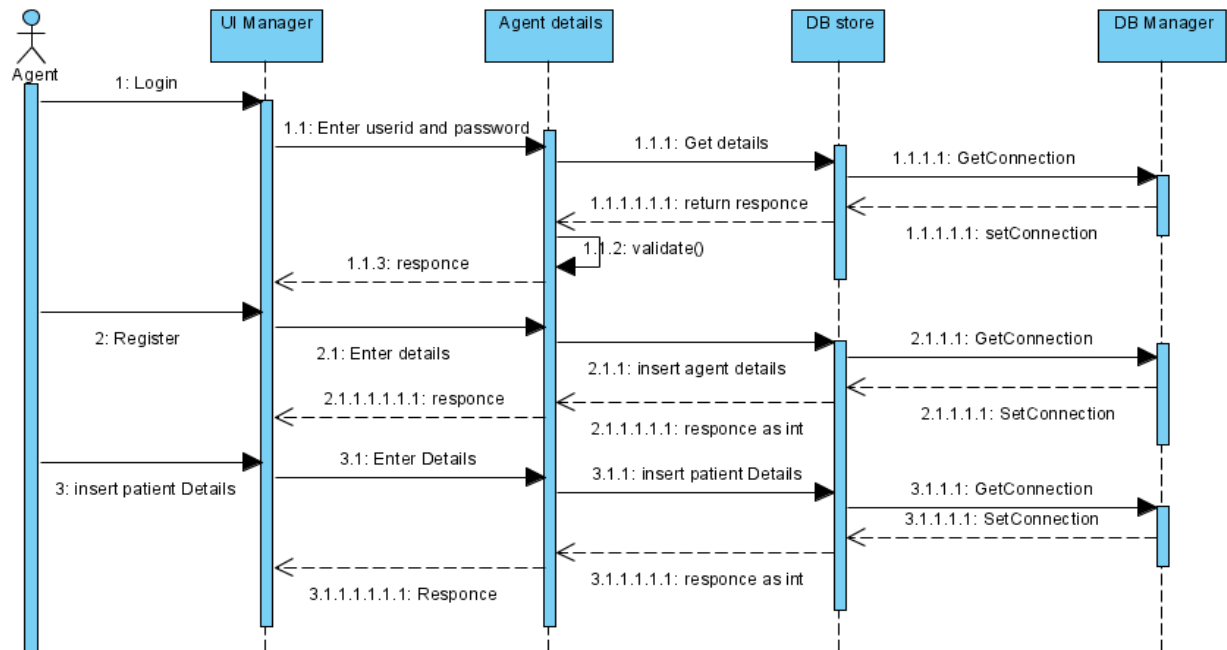


Fig 5.7: sequence diagram of agent

Admin:

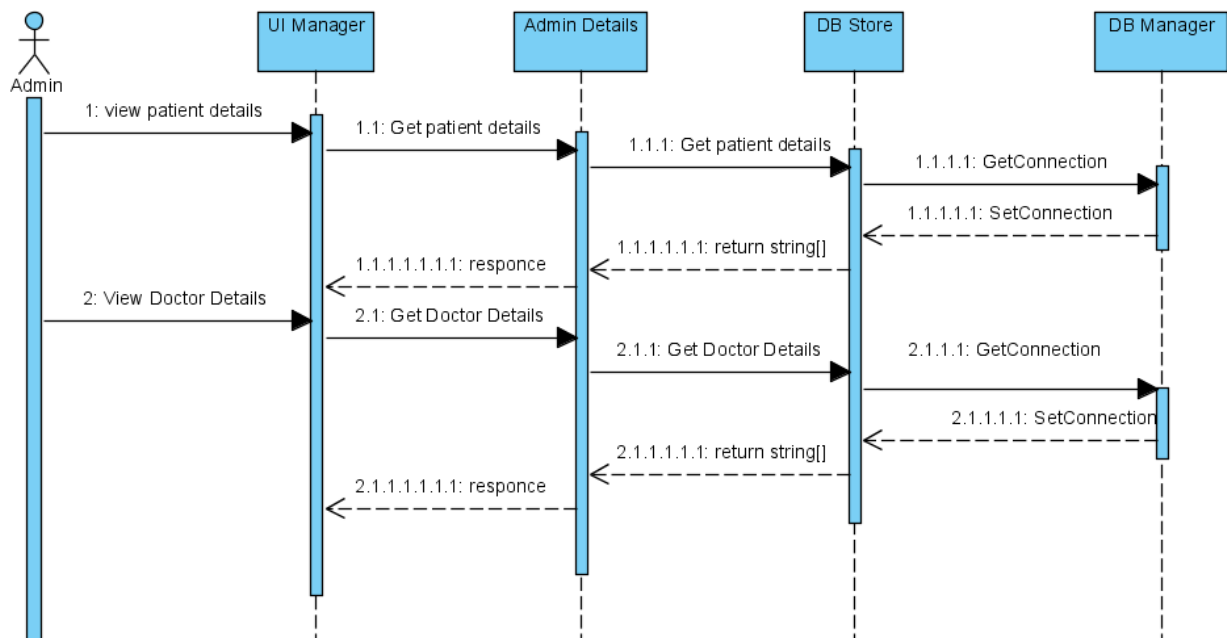


Fig 5.8: sequence diagram of admin

Collaboration Diagrams:

Agent:

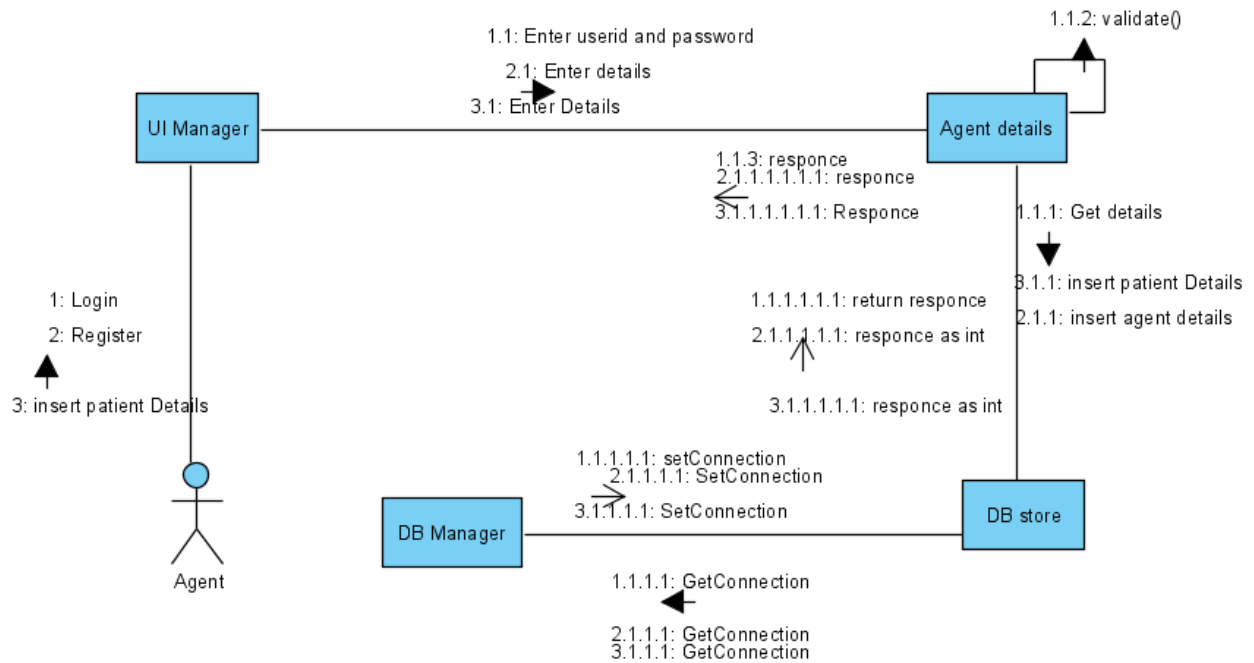


Fig 5.9: collaboration diagram of agent

Admin:

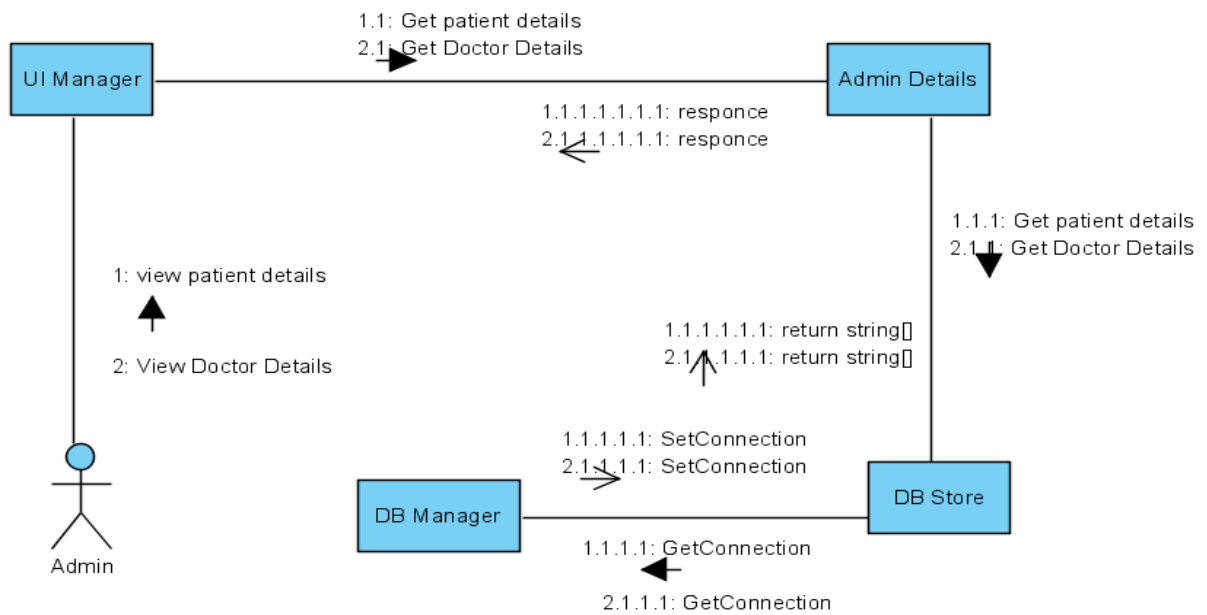


Fig 5.10: collaboration diagram of admin

Class Diagram:

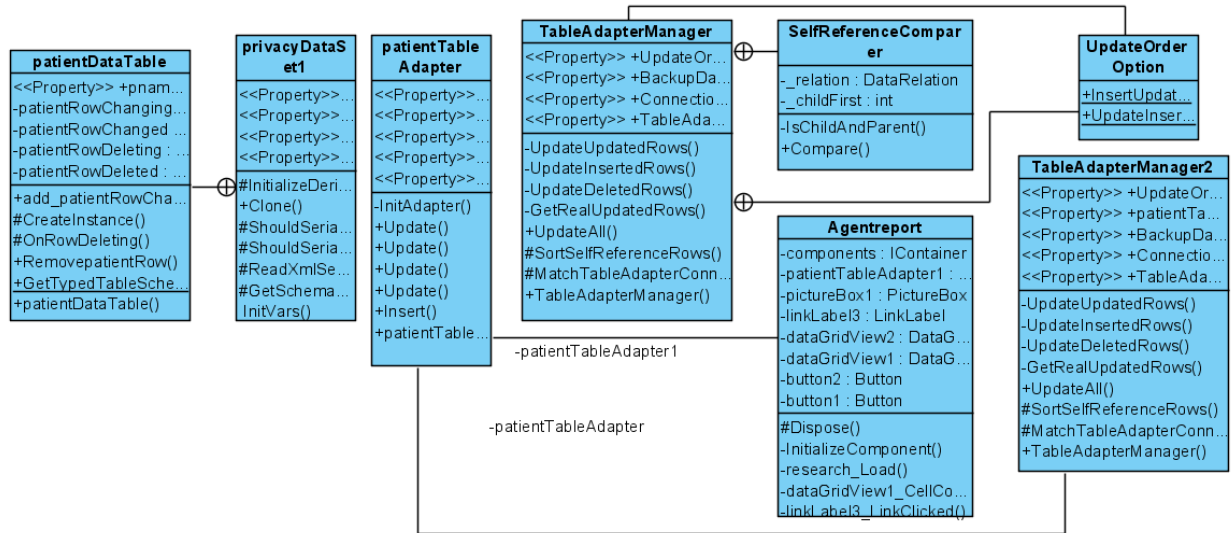


Fig 5.11: Class Diagram

Object Diagram:

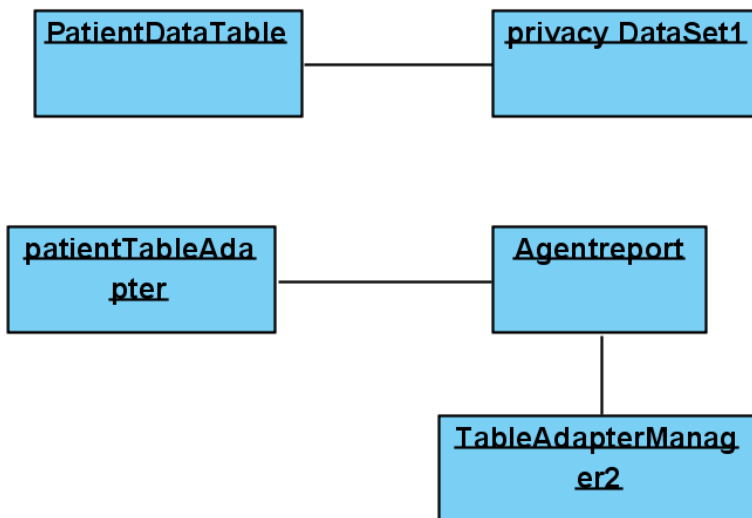


Fig 5.12: object diagram

Activity Diagram:

Patient:

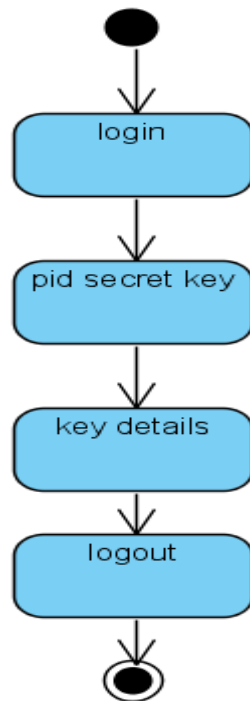


Fig 5.13: activity diagram of patient

Doctor:

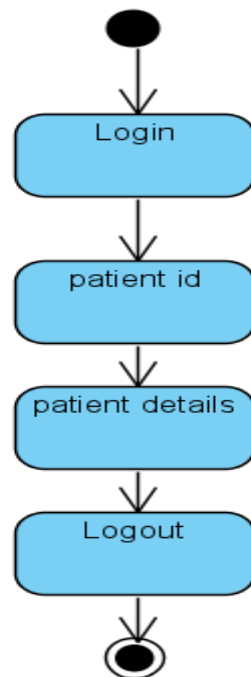


Fig 5.14: activity diagram of doctor

Agent:

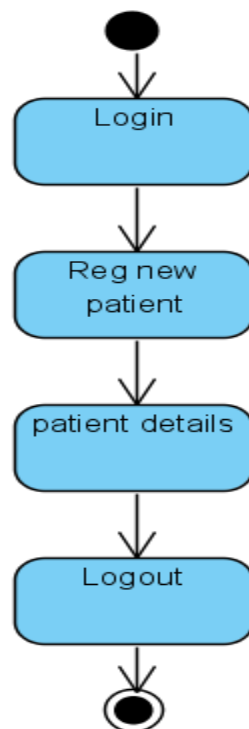


Fig 5.15: activity diagram of agent

Admin:

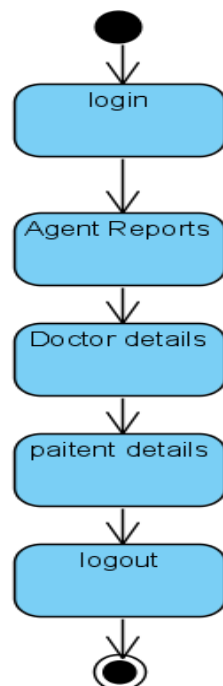


Fig 5.16: activity diagram of admin

State chart diagram:

Admin:

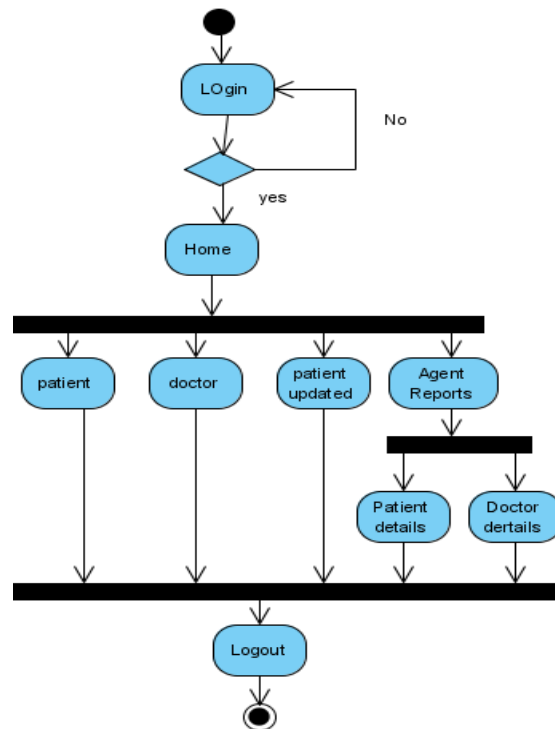


Fig 5.17: state chart diagram of admin

Agent

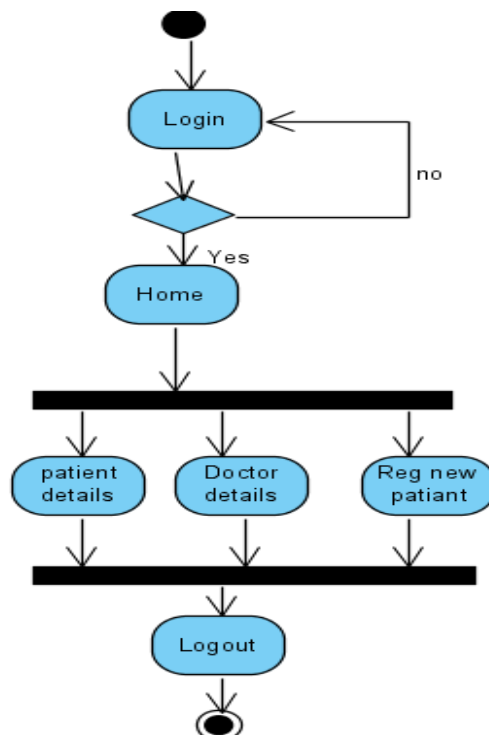


Fig 5.18: state chart diagram of agent

Doctor

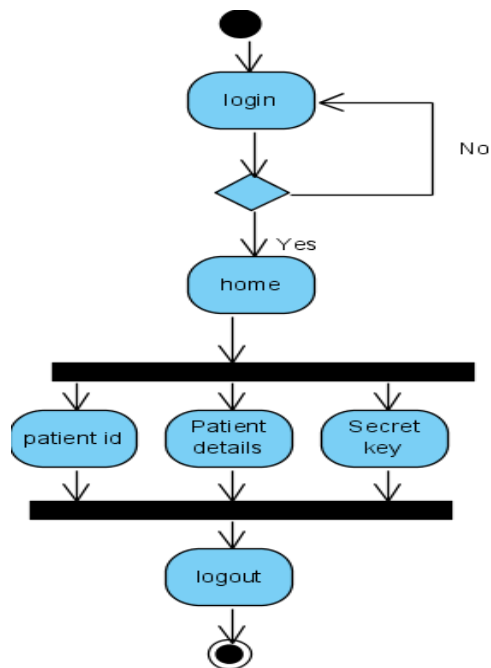


Fig 5.19: state chart diagram of doctor

Patient:

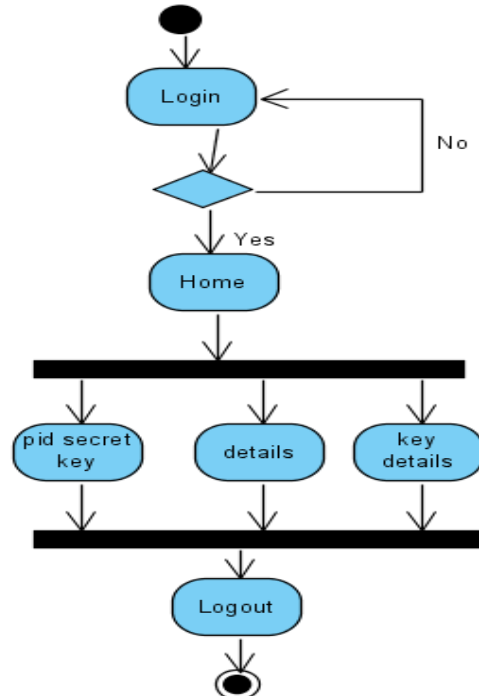


Fig 5.20: state chart diagram of patient

Deployment Diagram:

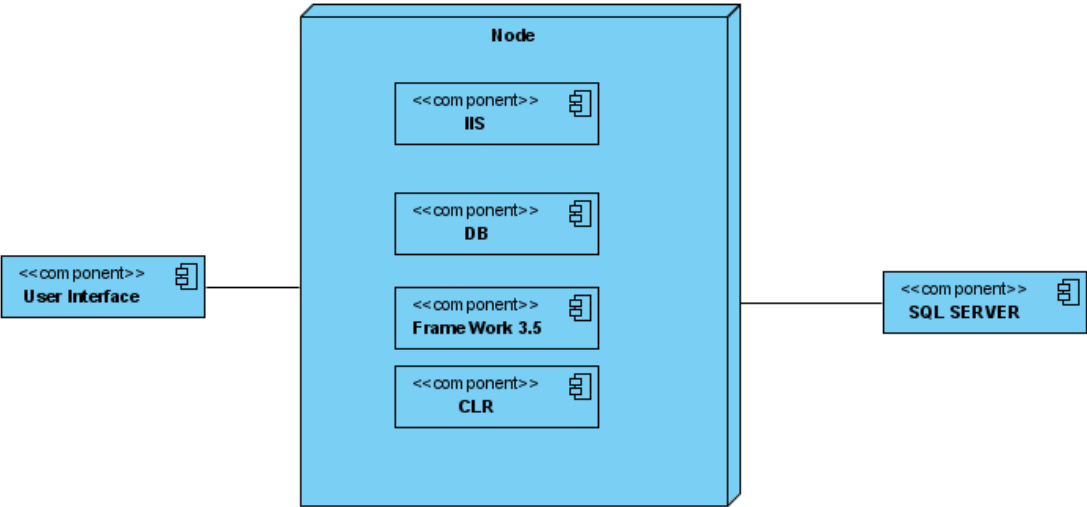


Fig 5.21: deployment diagram

COMPONENT:

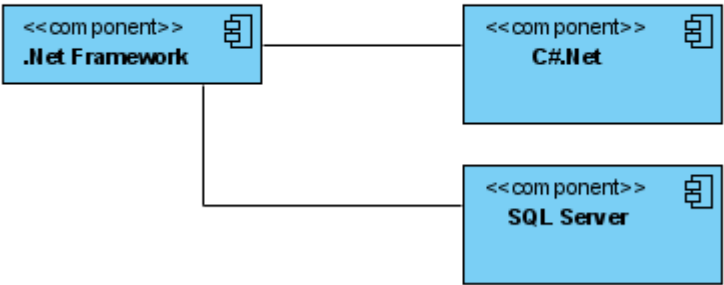


Fig 5.22: component diagram

Database Tables:

Doctor Table:


	Column Name	Data Type	Allow Nulls
	dname	varchar(50)	<input checked="" type="checkbox"/>
	age	int	<input checked="" type="checkbox"/>
	gender	varchar(10)	<input checked="" type="checkbox"/>
	department	varchar(100)	<input checked="" type="checkbox"/>
	password	varchar(50)	<input checked="" type="checkbox"/>
	dtime	datetime	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Table 5.1: doctor table

Patient Table:


	Column Name	Data Type	Allow Nulls
	pname	varchar(20)	<input checked="" type="checkbox"/>
	age	int	<input checked="" type="checkbox"/>
	dob	varchar(20)	<input checked="" type="checkbox"/>
	disease	varchar(20)	<input checked="" type="checkbox"/>
	phoneno	bigint	<input checked="" type="checkbox"/>
	gender	varchar(10)	<input checked="" type="checkbox"/>
	password	varchar(30)	<input checked="" type="checkbox"/>
	dtime	datetime	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Table 5.2: patient table

Patient Update:


	Column Name	Data Type	Allow Nulls
	pname	varchar(20)	<input checked="" type="checkbox"/>
	age	int	<input checked="" type="checkbox"/>
	disease	varchar(20)	<input checked="" type="checkbox"/>
	password	varchar(30)	<input checked="" type="checkbox"/>
	diseasedetails	varchar(MAX)	<input checked="" type="checkbox"/>
	secretkey	varchar(50)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Table 5.3: patient update table

E-R Diagram:

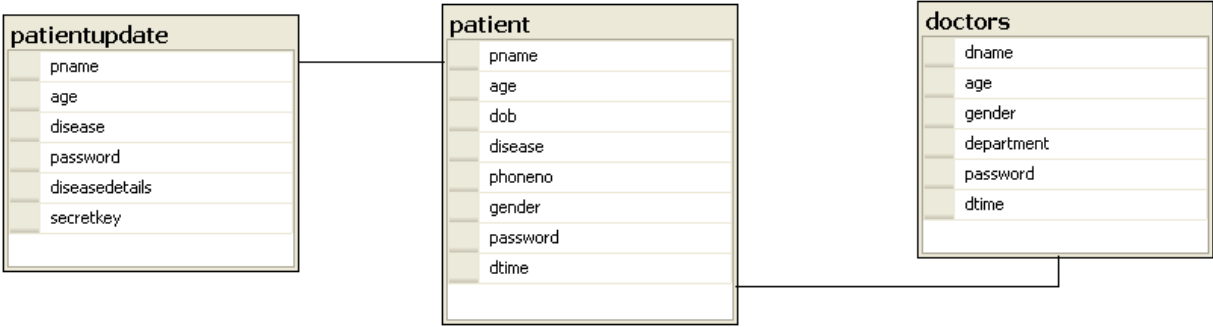


Fig 5.23: E-R diagram

6. SYSTEM IMPLEMENTATION

Microsoft.NET Framework

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

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

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

SQL SERVER

DATABASE:

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as row or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

6.1 CODE FREGMENTS

Homepage.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication4
{
```

```
public partial class homepage : Form
{
    public homepage()
    {
        InitializeComponent();
    }

    private void Patient_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        patient pa = new patient();
        pa.Show();
    }

    private void Doctor_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        Doclogin r = new Doclogin();
        r.Show();
    }

    private void linkLabel3_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        receplog d = new receplog();
        d.Show();
    }

    private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        hospitaladmin adminss = new hospitaladmin();
        adminss.Show();
        linkLabel1.Visible = true;
    }
}
```

-Database Connectivity code

```
Private void button2_Click (object sender, EventArgs e)
{
    SqlConnection con = new
SqlConnection(@"server=NEOAPP21\SQLEXPRESS;database=privacy;user
id=sa;password=neoapp123");
    con.Open();
    string cmd = "select dname as 'Doctorname',age as 'Age',gender as 'Sex', department
as 'Department',dtime as 'RegisterTime', password as 'ID' from doctors";
    DataSet ds = new DataSet();
    SqlDataAdapter sda = new SqlDataAdapter(cmd, con);
    sda.Fill(ds);
    dataGridView2.DataSource = ds.Tables[0].DefaultView;
}
```

7. SYSTEM TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

There of basically two types of testing approaches.

One is *Black-Box testing* – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated

The other is *White-Box testing* – knowing the internal workings of the product ,tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package. The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

7.1 TESTING APPROACHES

7.1.1 TOP-DOWN INTEGRATION TEST:

Modules are integrated by moving downwards through the control hierarchy beginning with main program. The subordinate modules are incorporated into structure in either a breadth first manner or depth first manner. This process is done in five steps:

- Main control module is used as a test driver and steps are substituted or all modules directly to main program.
- Depending on the integration approach selected subordinate is replaced at a time with actual modules.
- Tests are conducted.
- On completion of each set of tests another stub is replaced with the real module
- Regression testing may be conducted to ensure trha6t new errors have not been introduced.

This process continuous from step 2 until entire program structure is reached. In top down integration strategy decision making occurs at upper levels in the hierarchy and is encountered first. If major control problems do exists early recognitions is essential.

If depth first integration is selected a complete function of the software may be implemented and demonstrated.

Some problems occur when processing at low levels in hierarchy is required to adequately test upper level steps to replace low-level modules at the beginning of the top down testing. So no data flows upward in the program structure.

7.1.2 BOTTOM-UP INTEGRATION TEST:

Begins construction and testing with atomic modules. As modules are integrated from the bottom up, processing requirement for modules subordinate to a given level is always available and need for stubs is eliminated. The following steps implements this strategy.

- Low-level modules are combined in to clusters that perform a specific software sub function.
- A driver is written to coordinate test case input and output.
- Cluster is tested.
- Drivers are removed and moving upward in program structure combines clusters.

Integration moves upward, the need for separate test driver's lesions

If the top levels of program structures are integrated top down, the number of drivers can be reduced substantially and integration of clusters is greatly simplified.

7.2 TESTING STRATEGIES:

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

The main objective of software testing is to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases.

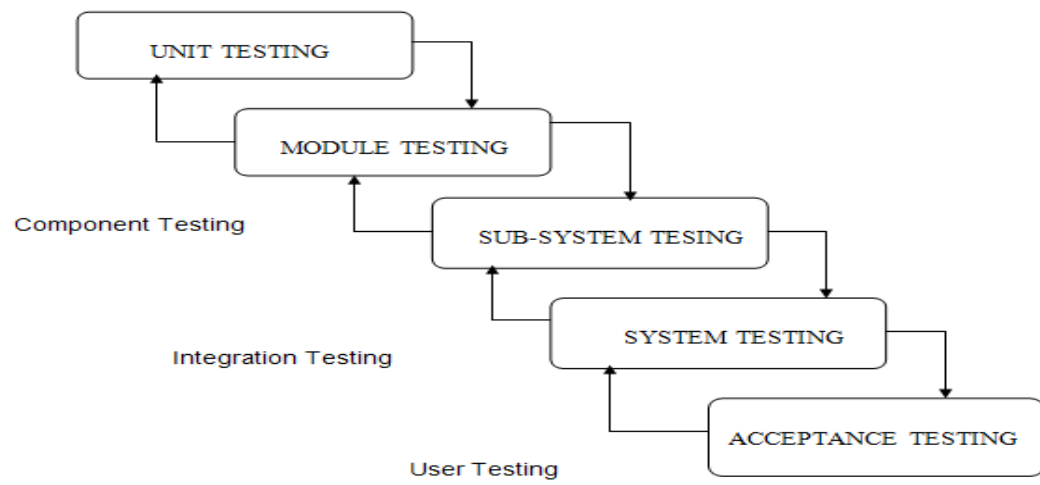


Fig 7.1: testing strategies

8. RESULT ANALYSIS

SCREEN SHORTS

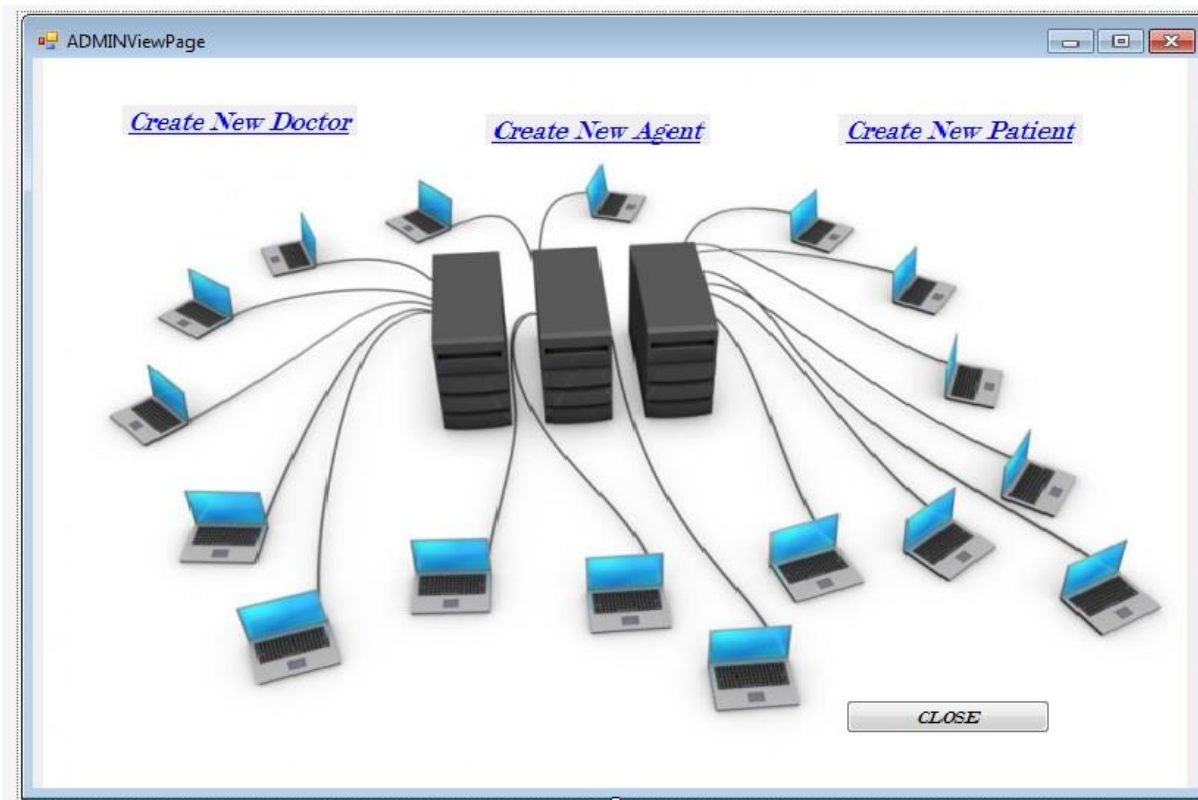
Home Page:



Admin login:




Admin Home Page:



Agent Reports:



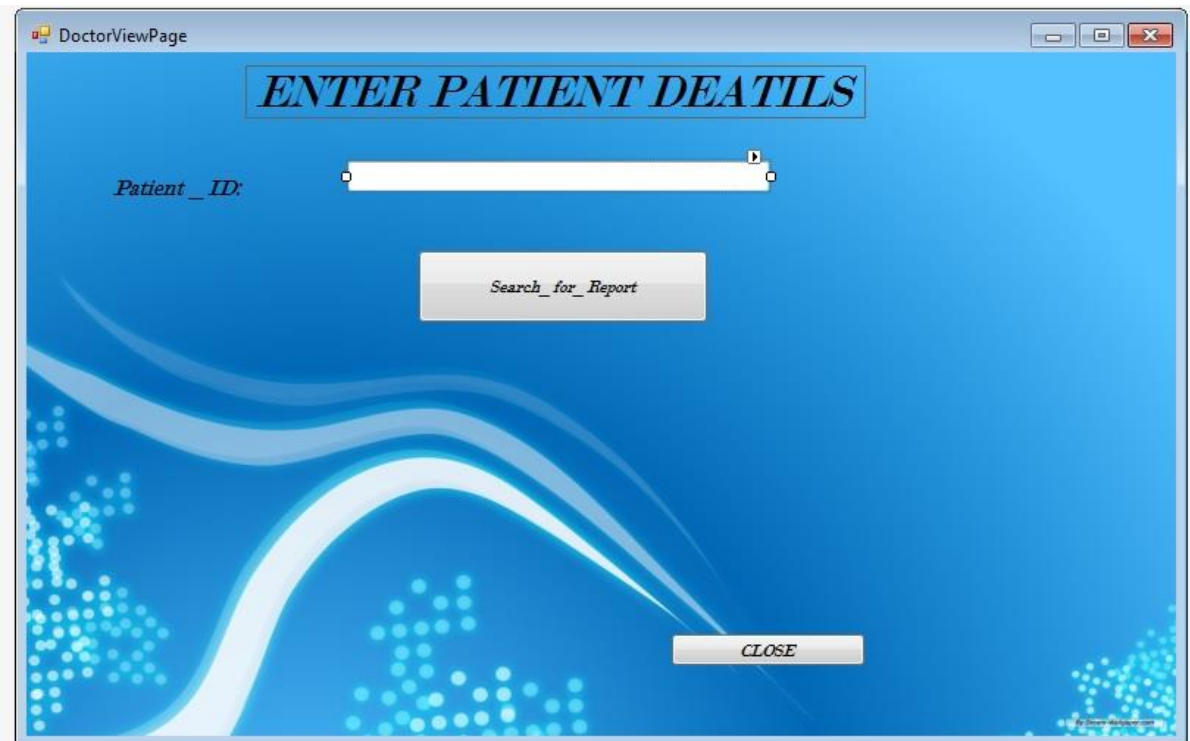
Doctor Login:



The **Doctorlogin** window displays a login form overlaid on a background image of five medical professionals (three doctors in white coats and two nurses in blue scrubs) standing in a hospital hallway. The form includes the following elements:

- Header:** *LOGIN TO YOUR ACCOUNT*
- Fields:**
 - Doctor_Name:* [Text input field]
 - Password:* [Text input field]
- Buttons:**
 - Login*
 - Cancel*

Doctor Home Page:



The **DoctorViewPage** window features a blue background with a white wavy line and a cluster of blue dots in the bottom left corner. The interface includes the following elements:

- Header:** *ENTER PATIENT DEATILS* (Note the spelling error in the image)
- Form:**
 - Patient_ID:* [Text input field]
- Buttons:**
 - Search_for_Report*
 - CLOSE*

9. CONCLUSION

To detect anomalous insiders in a CIS, we proposed CADS, a community anomaly detection system that utilizes a relational framework. To predict which users are anomalous, CADS calculates the deviation of users based on their nearest neighbor networks. We further extended CADS into MetaCADS to incarnate the semantics of the subjects accessed by the users. Our model is based on the observation that “normal” users tend to form communities, unlike illicit insiders. To evaluate the performance of our model, we conducted a series of experiments that compared our framework with the state-of-the-art anomaly detection methods for CIS systems. In the experiments, we mixed simulated users with the real users of a real electronic health record system. Our results illustrated that the community-based models exhibited better performance at detecting simulated insider threats. The evidence further suggested that MetaCADS is the best model when the number of intruders is relatively small, but that CADS dominates when the number of intruders increases. Since the framework is an unsupervised system, we believe it may be implemented in real time environments with offline training. There are limitations of the system; however, and in particular, we intend to validate and improve our system with adjudication through real human experts.

10. REFERENCES

- [1] L.A. Adamic and E. Adar, "Friends and Neighbors on the Web," *Social Networks*, vol. 25, no. 3, pp. 211-230, 2003.
- [2] M. Alawneh and I. Abbadi, "Preventing Information Leakage between Collaborating Organizations," *Proc. 10th Int'l Conf. Electronic Commerce*, pp. 185-194, 2008.
- [3] V. Bellotti and S. Bly, "Walking Away from the Desktop Computer: Distributed Collaboration and Mobility in a Product Design Team," *Proc. ACM Conf. Computer Supported Cooperative Work*, pp. 209-218, 1996.
- [4] F. Benaben, J. Touzi, V. Rajsiri, and H. Pingaud, "Collaborative Information System Design," *Proc. Int'l Conf. Assoc. Information and Management*, pp. 281-296, 2006.
- [5] A.A. Boxwala, J. Kim, J.M. Grillo, and L.O. Machado, "Using Statistical and Machine Learning to Help Institutions Detect Suspicious Access to Electronic Health Records," *J. Am. Medical Informatics Assoc.*, vol. 18, pp. 498-505, 2011.
- [6] J. Byun and N. Li, "Purpose Based Access Control for Privacy Protection in Relational Database Systems," *Int'l J. Very Large Databases*, vol. 17, pp. 603-619, 2008.
- [7] S. Chakraborty and I. Ray, "TrustBac: Integrating Trust Relationships Into the RBAC Model for Access Control in Open Systems," *Proc. 11th ACM Symp. Access Control Models and Technologies*, pp. 49-58, 2006.
- [8] H. Chen, F. Wang, and D. Zeng, "Intelligence and Security Informatics for Homeland Security: Information, Communication, and Transportation," *IEEE Trans. Intelligent Transportation Systems*, vol. 5, no. 4, pp. 329-341, Dec. 2004.
- [9] H. Chen, D. Zeng, H. Atabakhsh, W. Wyzga, and J. Schroeder, "COPLINK: Managing Law Enforcement Data and Knowledge," *Comm. ACM*, vol. 46, no. 1, pp. 28-34, 2003.
- [10] Y. Chen and B. Malin, "Detection of Anomalous Insiders in Collaborative Environments via Relational Analysis of Access Logs," *Proc. First ACM Conf. Data and Application Security and Privacy*, pp. 63-74, 2011.
- [11] Y. Chen, S. Nyemba, W. Zhang, and B. Malin, "Leveraging Social Networks to Detect Anomalous Insider Actions in Collaborative Environments," *Proc. IEEE Ninth Intelligence and Security Informatics*, pp. 119-124, 2011.
- [12] P. Cheng, P. Rohatgi, C. Keser, P.A. Karger, and G.M. Wagner, "Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control," *Research Report RC24190 (W0702-085)*, IBM, 2007.
- [13] J. Crampton and M. Huth, *Towards an Access-Control Framework for Countering Insider Threats*. Springer, 2010.

[14] W. Eberle and L. Holder, "Applying Graph-Based Anomaly Detection Approaches to the Discovery of Insider Threats," Proc. IEEE Int'l Conf. Intelligence and Security Informatics, pp. 206-208, 2009.

[15] L. Eldenburg, N. Soderstrom, V. Willis, and A. Wu, "Behavioural Changes Following the Collaborative Development of an Accounting Information System," Accounting, Organizations and Soc., vol. 35, no. 2, pp. 222-237, 2010.

[16] J. George, G. Easton, J. Nunamaker, and G. Northcraft, "A Study Of Collaborative Group Work with and without Computer-Based Support," Information Systems Research, vol. 1, no. 4, pp. 394-415

- FOR .NET INSTALLATION

www.support.mircosoft.com

- FOR DEPLOYMENT AND PACKING ON SERVER

www.developer.com

www.16seconds.com

- FOR SQL

www.msdn.microsoft.com

www.asptoday.com

www.aspfree.com