**Chapter 1**

**INTRODUCTION**

**1.1 COMPANY PROFILE**

NeoApp develops custom software solutions for companies in a variety of industries. Since it’s beginning in August 2008, NeoApp has offered efficient, reliable and cost-effective solutions with good quality by implementing CMMI practices from it’s development facility located at Hyderabad.

NeoApp has expertise in latest technologies and caters to your exacting requirements. NeoApp helps you from concept to completion of a project with full range of service offerings. Most importantly, NeoApp combines the right strategy with the right products and the right people, ensuring technical superiority, high quality deliverables and timely implementations. NeoApp supports different delivery and billing models to fit your requirements. By having NeoApp involved with your software development projects, you benefit with reduced costs and faster development cycles. To reduce the development costs NeoApp strictly adhere on reusable component model with plug and play architecture.Offshore outsourcing model became easily adoptable and has increased benefits beyond cost reductions. The offshore outsourcing with NeoApp includes full spectrum services and multi fold benefits.

NeoApp, with its experience in executing offshore projects ranging from large enterprise solutions to small plug-in applications, helps customers achieve the offshore outsourcing goals.

NeoApp establishes suitable project execution methodologies for each project and accomplishes offshore execution on time and on budget. NeoApp pays high importance to quality of deliverables and has mandatory quality doors in place for each project ensuring success of the overall project.

NeoApp works with you from conceptualization to completion and has the required expertise to pick up the project at any stage in its life cycle.

1. Business concept and system study
2. Requirement Study
3. Design Architecture and develop specifications
4. Design the framework of the solution
5. Develop the solution
6. QA the solution against requirements
7. Continuous support for the solution
8. Develop prototypes for proof of concept
9. Engineer the solution

10. Release as per plan

The team and project approach of NeoApp has resulted in above expected deliveries of projects. NeoApp works with you in refining the project at every stage and with its vast and experienced talent pool, NeoApp brings value with innovation to the project.

NeoApp offers complete solutions to application maintenance requirements helping organizations to cut costs and optimize resource utilization. NeoApp performs the following tasks on a variety of technology platforms beginning with Legacy to Client Server to Browser based internet application.

1. Application Development

2. Application Maintenance

3. Application Support

NeoApp with its experience in wide range technologies and ability to learn quickly help you ensuring availability of your systems to your customers. NeoApp performs systems monitoring and undertakes evolutionary development of these applications as required and deemed fit.

**1.2 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. 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.

**Chapter 2**

2. Literature Survey

Preventing Information Leakage between Collaborating

Organisations

Imad M. Abbadi Group Technology Services

Information sharing and protection against leakage is a critical problem especially for organisations having sensitive information.Sharing content between individuals in the same organisation extends to exchanging and sharing content between collaborating organisations. In this paper we

Propose a novel solution for preventing shared information between collaborating organizationsn from getting leaked to unauthorized users inside the destination organisation or out side it.In addition, once the content is in the hands of authorised users our solution prevents unethical authorised users from leaking such content to other users in the same organisationor third parties.

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

In the proposed solution we used trusted computingtechnology to provide a hardware-based root of trust for the master controller and organisation 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) [GunterLM], 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 Protectionin Relational Database Systems

-Won Byun, Ninghui Li

In this article, we present a comprehensive approachfor privacy preserving access control based on thenotion of purpose. In our model, purpose information associatedwith a given data element specifies the intended useof the data element. A key feature of our model is that it allowsmultiple purposes to be associated with each data elementand also supports explicit prohibitions, thus allowingprivacy officers to specify that some data should not be usedfor certain purposes. An important issue addressed in this articleis the granularity of data labeling, that is, the units ofdata with which purposes can be associated. We address thisissue in the context of relational databases and propose fourdifferent labeling schemes, each providing a different granularity.

We also propose an approach to representing purposeinformation, which results in low storage overhead, and weexploit query modification techniques to support access controlbased on purpose information. Another contribution ofour work is that we address the problem of how to determinethe purpose for which certain data are accessed by agiven user. Our proposed solution relies on Role-Based AccessControl (RBAC) models as well as the notion of conditionalrole which is based on the notions

Collaborative information seeking:

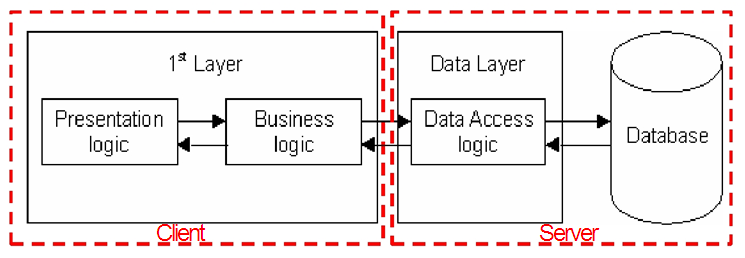
**Collaborative information seeking (CIS)** is a field of research that involves studying situations, motivations, and methods for people working in collaborative groups for information seeking projects, as well as building systems for supporting such activities. Such projects often involve information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

**Chapter 3**

**3. SYSTEM ANALYSIS**

**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.



**Fig 3.1: client - server architecture**

Database runs on Server

 Separated from client

 Easy to switch to a different database

 Presentation and logic layers still tightly connected

 Heavy load on server

 Potential congestion on network

 Presentation still tied to business logic

**ANALYSIS MODEL**

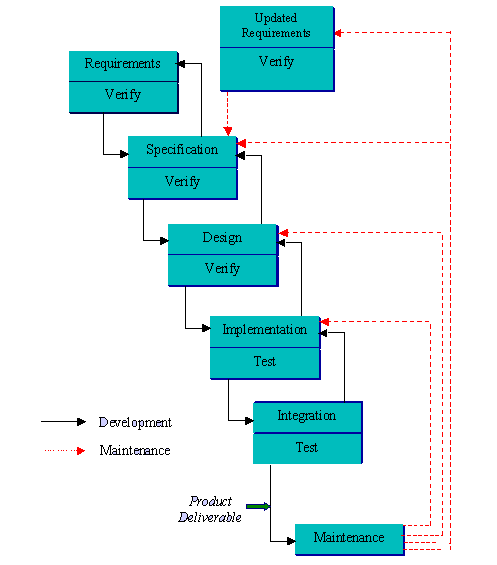
The model that is basically being followed is the WATER FALL MODEL, which states that the phases are organized in a linear order. First of all the feasibility study is done. Once that part is over the requirement analysis and project planning begins. If system exists one and modification and addition 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 3.2: Water Fall Model**

* 1. **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

* 1. **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

1. HVU Acheives More Comparable Results When Compared To Supervised Models
2. MetaCADS dominates when the mix rate is low, but CADS dominates when the mix rate is high
   1. **PROCESS DIAGRAM**



**Fig 3.3: 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.

* 1. **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 browses 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
* Economical 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 carryout 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 FEASBILITY**

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. Economical 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

**Chapter 4**

1. **SYSTEM REQUIREMENTS SPECIFICATION**

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 Requirements**

Operating System : Windows XP or 7

Language : C#

Tool : Visual Studio 2005 or latest

Data Base : SQL Server 2005

**4.2 Hardware Requirements**

System : Pentinum IV and above

Hard Disk : 50 GB and above

Monitor : VGA Colour

Ram : 1GB and above

**4.3 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**

**4.4 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.

**Chapter 5**

1. **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 requirement 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 High Level Design**

**System Design:-**

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

User Interface

Detecting Anomalous Insiders in Collaborative Information Systems

**Fig 5.1: System Design**

**Sub system design:-**

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

Databaseeeee System

Web User Interface

Detecting Anomalous Insiders in Collaborative Information Systems

**Fig 5.2 : Sub System Design**

**Blocks Design:**

UI Manager

DB Storage

Login Manager

Doctor Details

DB Manager

Patient Details

Registration

Error Handler

DB Storage

Agent details

DB Storage

**Fig 5.3:Block Design**

**Chapter 6**

1. **PROJECT DESCRIPTION**

**6.1 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: a) construction of the user-subject access network and b) 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 e) discovering a user’s nearest neighbors and f) 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 nonanomalous user class labels, whereas the final model is an unsupervised heuristic. For each of these models, we treat real and simulated users as nonanomalous 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.

1. **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

****

**Fig 6.1: Block Diagram of Project process**

* 1. **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 users 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 is specifically constructed through two different domains they are
* 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.

# **INTRODUCTION TO THE UNIFIED MODIFIED LANGUAGE:-**

Building a model for a software system prior to its construction is as essential as having a blueprint for building a large building. Good models are essential for communication among project teams. As the complexity of the systems increases, so does the importance of good modeling techniques.

A modeling language must include:

Model elements- fundamentally modeling concepts and semantics. Notation-visual rendering of model elements Guidelines-expression of usage within trade. The use of visual notation to represent or model a problem can provide us several benefits relating to clarity, familiarity, maintenance, and simplification. The main reason for modeling is the reduction of complexity. The Unified Modeling Language (UML) is a set of notations and conventions used to describe and model an application. The UML is intended to be a universal language for modeling systems, meaning that it can express models of many different kinds and purposes, just as a programming language or a natural language can be used in different ways. A model” is an abstract representation of a system , constructed to understand the system prior to building or modifying it. The term “system” is used here in a broad sense to include any process or structure. For example, the organizational structure of a corporation , health services, computer software, instruction of any sort (including computers) , the national economy, and so forth all would be termed “systems”. The unified modeling language is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English, in a form known as “object constraint language”(OCL).OCL is a specification language that uses simple logic for specifying the prop-erties of a system.The UML is not intended to be a visual programming language in the sense of having all the necessary visual and semantic support to replace programming languages. However, the UML does have a tight mapping to a family of object-oriented languages, so that you can get the best of both worlds.

**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.

**Architectural views and diagrams of the UML**

The UML Meta model elements are organized into diagrams. Different diagrams are used for different purposes depending on the angle from which you are viewing the system. The different views are called “architectural views”. Architectural views facilitate the organization of knowledge, and diagrams enable the communication of knowledge. Then knowledge itself is within the model or set of models that focuses on the problem and solution. The architectural views and their diagrams are summarized below:

* The “user model view” encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of “use case diagrams”. These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships.
* *“*Class diagrams*”* describe the static structure of a system, or how it is declared rather than how it behaves. These diagrams contain classes and associations.
* *“*object diagrams*”* describe the static structure of a system at a particular time during its life. These diagrams contain objects and links.
* “Sequence diagrams*”* render the specification of behavior. These diagrams describes the behavior provided by a system to interactions. These diagrams contain classes that exchange messages with in an interaction arranged in time sequence. In generic form, These diagrams describe a set of message exchange sequences among a set of classes. In instance form(scenarios), these diagrams describe one actual message exchange sequence among objects of those classes.
* “Collaboration diagrams*”*  render how behavior is realized by components with in a system. These diagrams contain classes, associations, and their message exchanges with in a collaboration to accomplish a purpose. In generic form, these diagrams describe a set of classes and associations involved in message exchange sequences. In instance form(scenarios), these diagrams describe a set of objects of those classes links confirming to the associations, and one actual message exchange sequence that inconsistent with the generic form and uses those objects and links.
* “State chart diagrams*”* render the states and responses of a class participating in behavior, and the life cycle of an object. These diagrams describe the behavior of a class in response to external stimuli.
* “Activity diagrams” render the activities of a class participating in behavior. These diagrams describe the behavior of a class in response to internal processing rather than external events. Activity diagrams describe the processing activities with in a class.

**6.2.1UML 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

**DATA FLOW DIAGRAMS:**

A data flow diagram is graphical tool used to deMHribe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be deMHribed logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full deMHription of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a deMHriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD’s is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is deMHribed for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**DFD SYMBOLS:-**

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow.

Source or Destination of data Data flow

Data

**CONSTRUCTING A DFD:**

Several rules of thumb are used in drawing DFD’s:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data Traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

**SILENT FEATURES OF DFD’s:**

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflows take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

**TYPES OF DATA FLOW DIAGRAMS:**

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

**CURRENT PHYSICAL:**

In Current Physical DFD proecess label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

**CURRENT LOGICAL:**

The physical aspects at the system are removed as mush as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

**NEW LOGICAL:**

This is exactly like a current logical model if the user were completely happy with he user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

**NEW PHYSICAL:**

The new physical represents only the physical implementation of the new system

**RULES GOVERNING THE DFD’S:**

PROCESS:

1. No process can have only outputs.
2. No process can have only inputs. If an object has only inputs than it must be a sink.
3. A process has a verb phrase labe+l.

DATA STORE:

1. Data cannot move directly from one data store to another data store, a process must move data.
2. Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3. A data store has a noun phrase label.

SOURCE OR SINK:

The origin and /or destination of data.

1. Data cannot move direly from a source to sink it must be moved by a process
2. A source and /or sink has a noun phrase land

DATA FLOW:

1. A Data Flow has only one direction of flow between symbol. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3. A data flow cannot go directly back to the same process it leads. There must be atleast one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4. A Data flow to a data store means update ( delete or change).
5. A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

**LEVELS OF DFD:**

The complexity of the business system means that it is a responsible to represent the operations of any system of single data flow diagram. At the top level, an Overview of the different systems in an organization is shown by the way of context analysis diagram. When exploded into DFD

They are represented by:

* LEVEL-1:SUBSYSTEM LEVEL DATAFLOW FUNCTIONAL
* LEVEL-2 : FILE LEVEL DETAIL DATA FLOW.

The input and output data shown should be consistent from one level to the next.

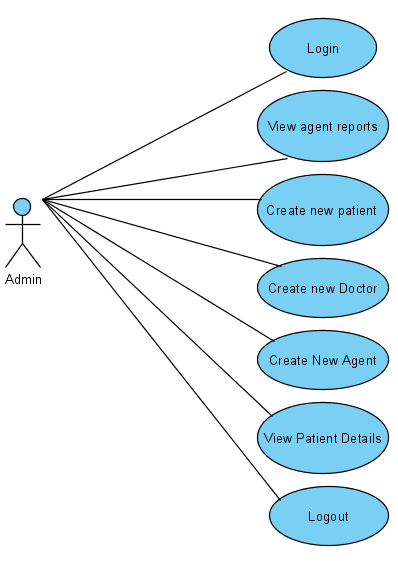
**LEVEL-1: SUBSYSTEM LEVEL DATA FLOW**

A level-1 DFD deMHribes the next level of details within the system, detailing the data flows between subsystems, which makeup the whole.

**LEVEL-2: FILE LEVEL DETAIL DATA FLOW**

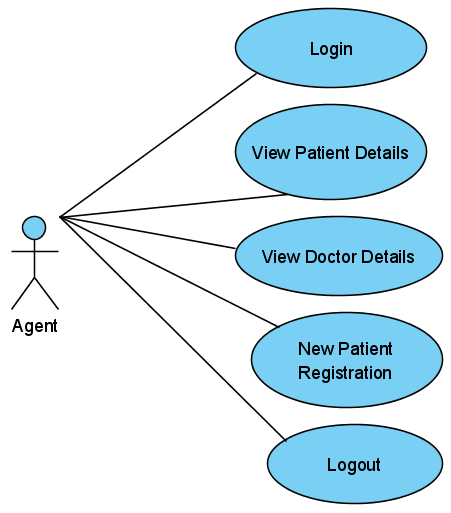
All the projects are feasible given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. Feasibility and the risk analysis are pertained in many ways. If project risk is great.

**Use Case Diagram**:



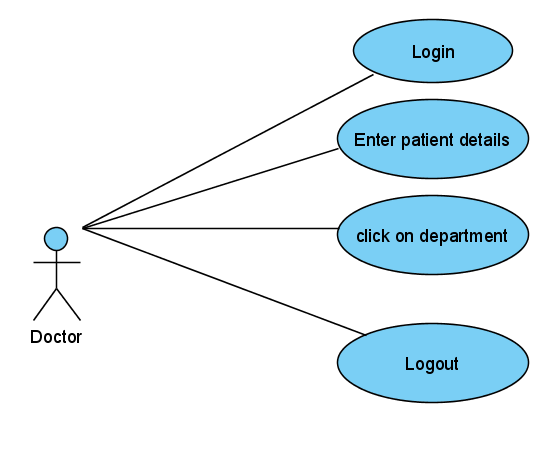
**Fig 6.2: use case diagram of Admin**

**Agent:**



**Fig 6.3 : use case diagram of agent**

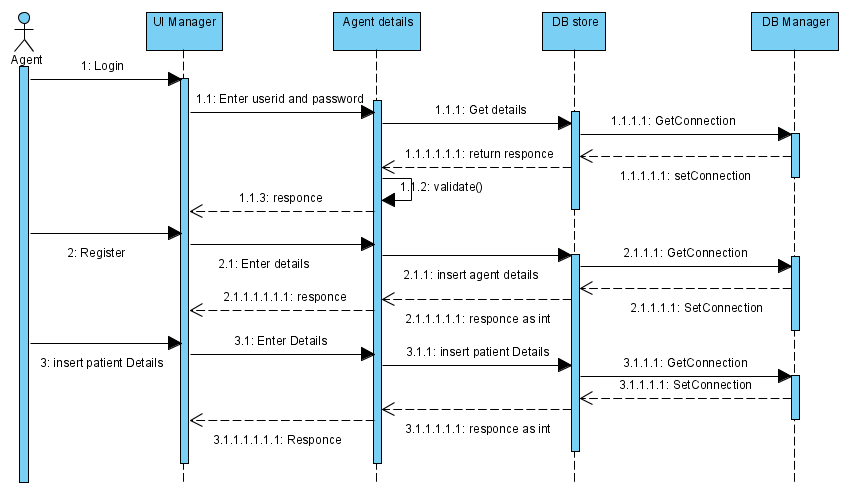
**Doctor:**



**Fig 6.4 usecase diagram of doctor**

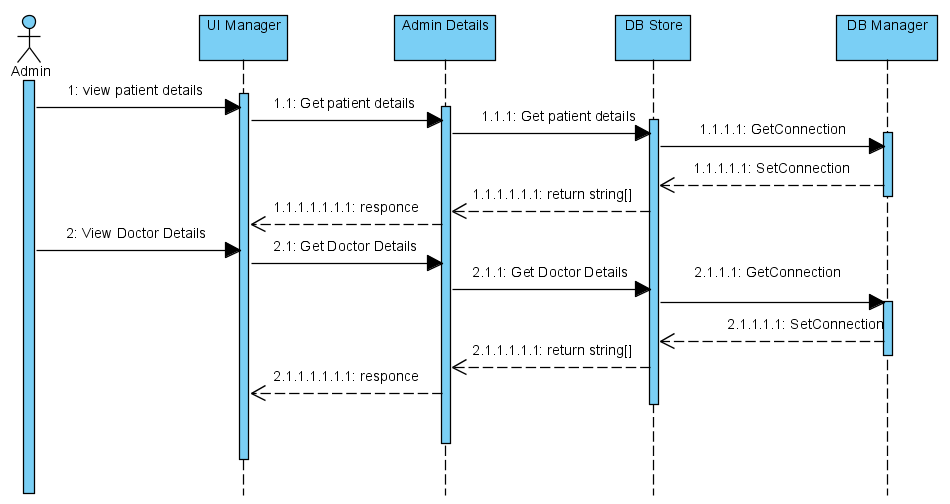
**Sequence Diagrams:**

Agent:



**Fig 6.5: sequence diagram of agent**

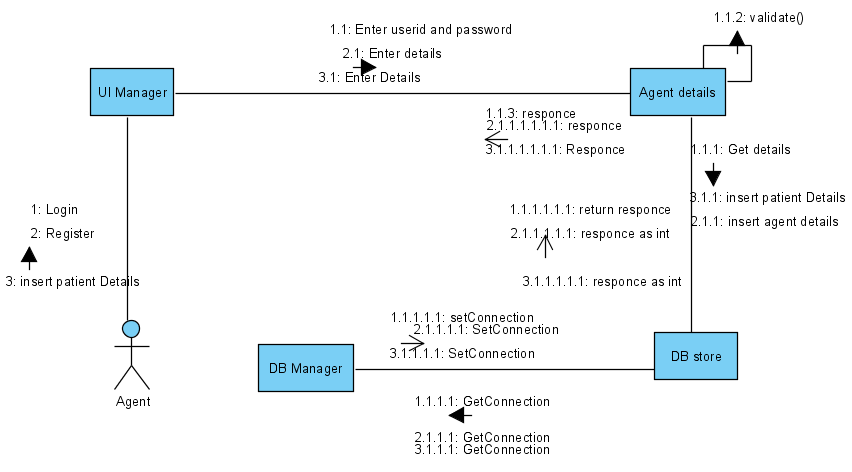
**Admin:**



**Fig 6.6: sequence diagram of admin**

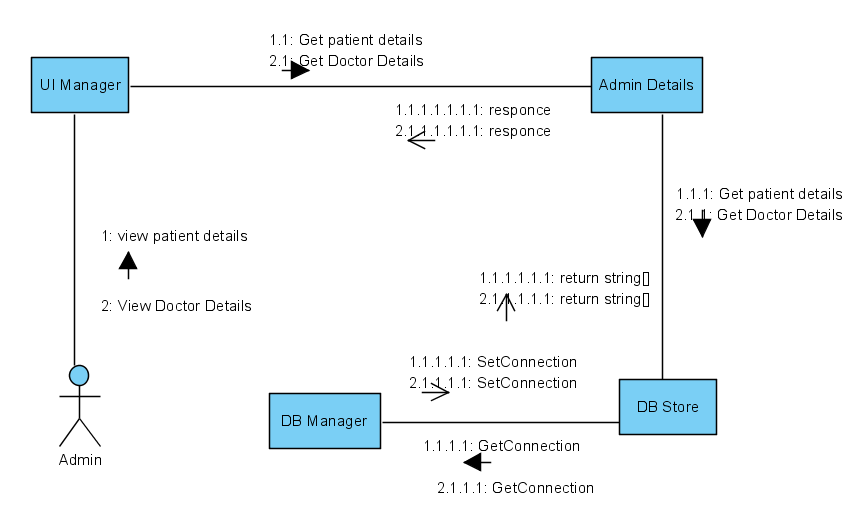
**Collaboration Diagrams:**

**Agent:**



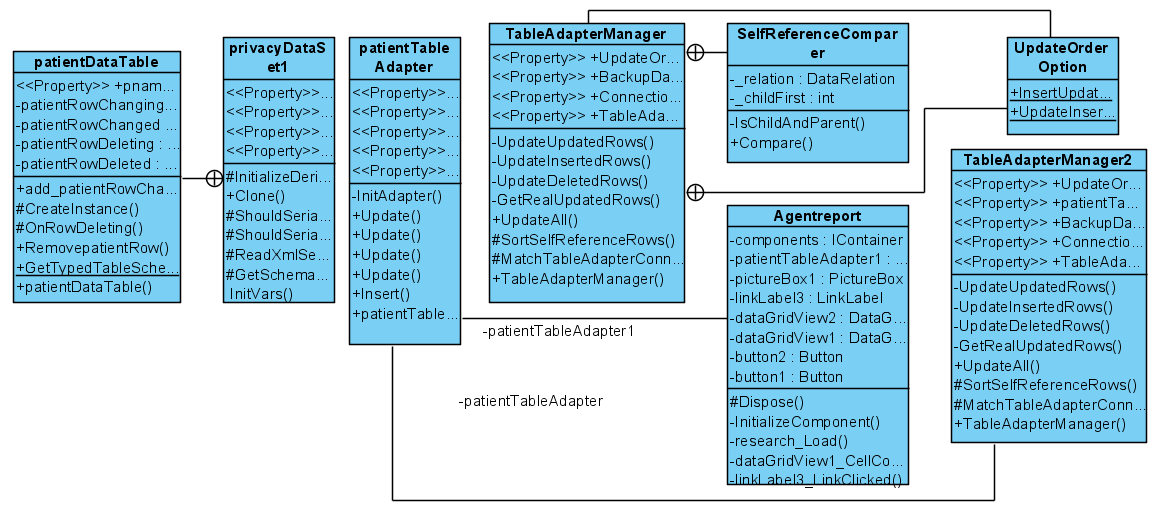
**Fig 6.7 : collaboration diagram of agent**

**Admin:**



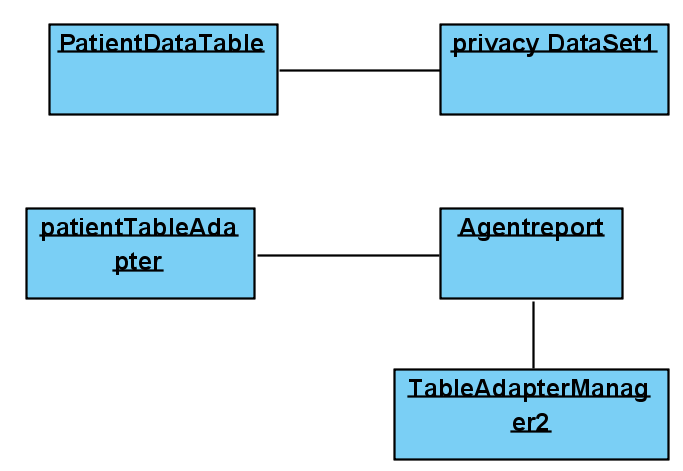
**Fig 6.8 : collaboration diagram of admin**

**Class Diagram:**



**Fig 6.9 :Class Diagram**

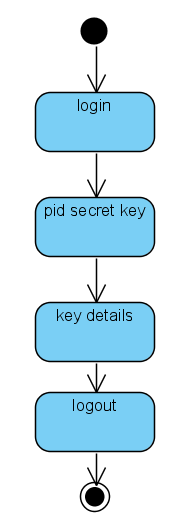
**Object Diagram:**



**Fig 6.10: object diagram**

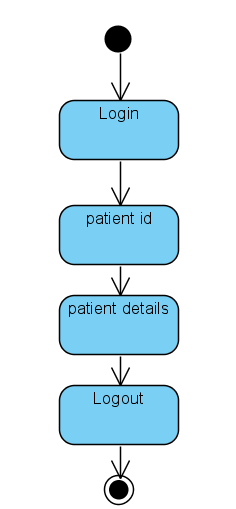
**Activity Diagram:**

**Patient:**



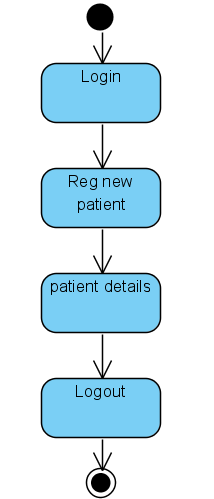
**Fig 6.11: activity diagram of patient**

**Doctor:**



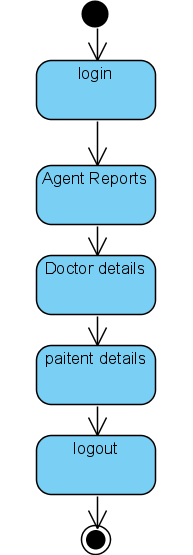
**Fig 6.12: activity diagram of doctor**

**Agent:**



**Fig 6.13: activity diagram of agent**

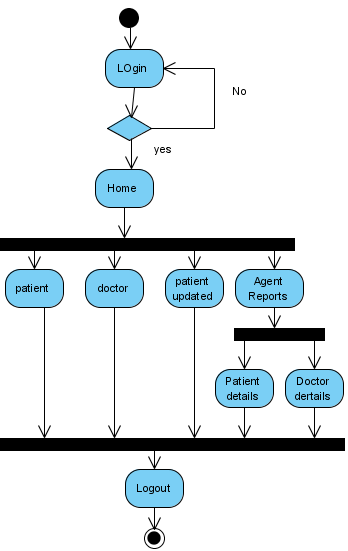
**Admin:**



**Fig 6.14: activity diagram of admin**

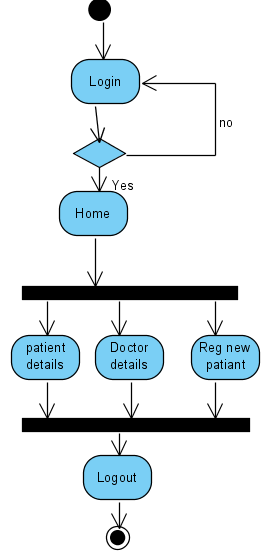
**State chart diagram:**

**Admin:**



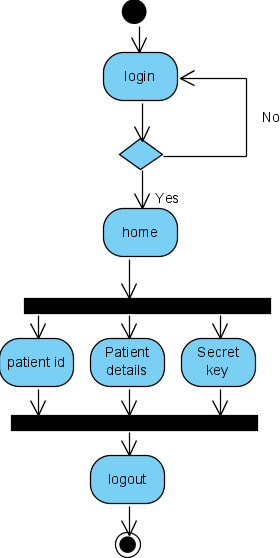
**Fig 6.15: state chart diagram of admin**

**Agent**



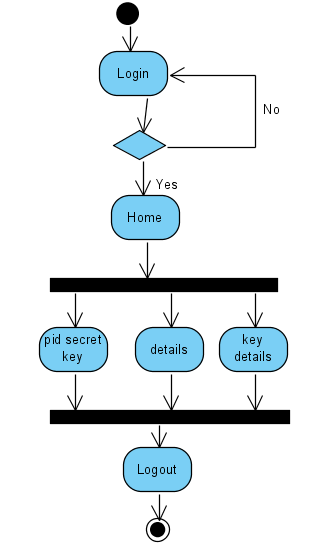
**Fig 6.16: state chart diagram of agent**

**Doctor**



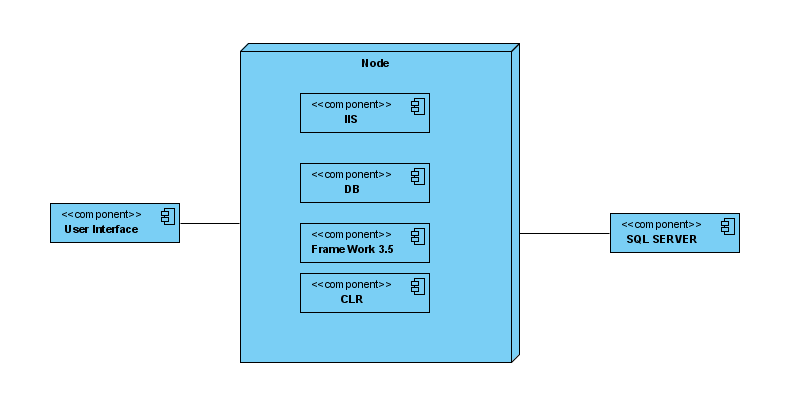
**Fig 6.17 : state chart diagram of doctor**

**Patient**



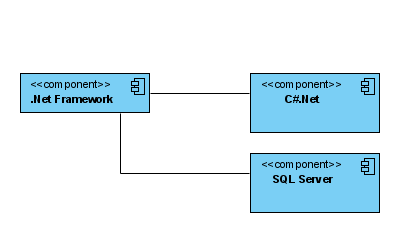
**Fig 6.18: state chart diagram of patient**

**Deployment Diagram:**



**Fig 6.19: deployment diagram**

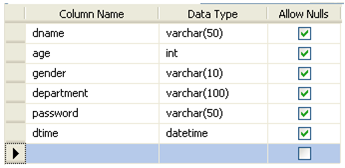
**COMPONENT:**



**Fig 6.20: component diagram**

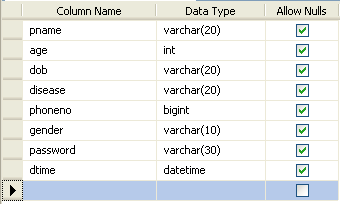
**Database Tables:**

**Doctor Table:**



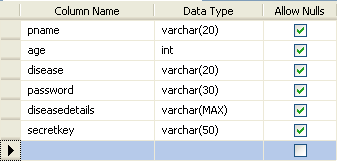
**Table 6.1 :doctor table**

**Patient Table:**



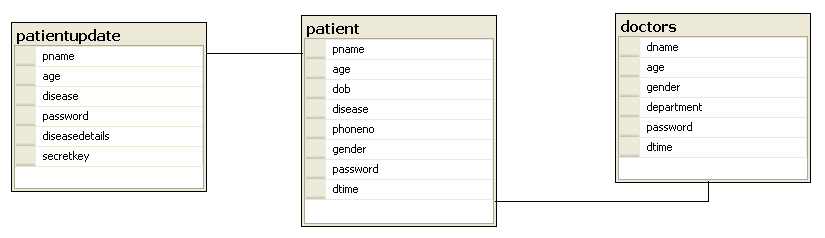
**Table 6.2: patient table**

**Patient Update:**



**Table 6.3: patient update table**

**E-R Diagram:**



**Fig 6.21: E-R diagram**

**Chapter 7**

**7.SOFTWARE DESCRIPTION**

**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 remoting, 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.

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

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

## **Features of the Common Language Runtime**

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally feature rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

## **.NET Framework Class Library**

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

* Console applications.
* Scripted or hosted applications.
* Windows GUI applications (Windows Forms).
* ASP.NET applications.
* XML Web services.
* Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

## **Client Application Development**

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlike ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can now be safely deployed through the Web. Your applications can implement the features of a local application while being deployed like a Web page.

**C#.NET**

**ACTIVE X DATA OBJECTS.NET**

#### **ADO.NET Overview**

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind. ADO.NET uses some ADO objects, such as the Connection and Command objects, and also introduces new objects. Key new ADO.NET objects include the DataSet, DataReader, and DataAdapter.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the DataSet -- that is separate and distinct from any data stores. Because of that, the DataSet functions as a standalone entity. You can think of the DataSet as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a DataSet, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A DataAdapter is the object that connects to the database to fill the DataSet. Then, it connects back to the database to update the data there, based on operations performed while the DataSet held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the DataAdapter, which provides a bridge to retrieve and save data between a DataSet and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based DataSet object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the DataSet is, it is manipulated through the same set of standard APIs exposed through the DataSet and its subordinate objects. While the DataSet has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the DataSet to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the Command, Connection, DataReader and DataAdapter. In the remaining sections of this document, we'll walk through each part of the DataSet and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

* Connections: For connection to and managing transactions against a database.
* Commands: For issuing SQL commands against a database.
* Data Readers: For reading a forward-only stream of data records from a SQL Server data source.
* Datasets: For storing, remoting and programming against flat data, XML data and relational data.
* DataAdapters. For pushing data into a DataSet, and reconciling data against a database.

When dealing with connections to a database, there are two different options: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

Connections:

Connections are used to 'talk to' databases, and are respresented by provider-specific classes such as SQLConnection. Commands travel over connections and resultsets are returned in the form of streams which can be read by a DataReader object, or pushed into a DataSet object.

Commands:

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as SQLCommand. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the Northwind database.

Data Readers:

The Data Reader object is somewhat synonymous with a read-only/forward-only cursor over data. The DataReader API supports flat as well as hierarchical data. A DataReader object is returned after executing a command against a database. The format of the returned DataReader object is different from a recordset. For example, you might use the DataReader to show the results of a search list in a web page.

DataSets and DataAdapters :

DataSets:  
The DataSet object is similar to the ADO Recordset object, but more powerful, and with one other important distinction: the DataSet is always disconnected. The DataSet object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a DataSet can and does behave much like a database, it is important to remember that DataSet objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into DataSet objects. Then, as changes are made to the DataSet they can be tracked and verified before updating the source data. The GetChanges method of the DataSet object actually creates a second DatSet that contains only the changes to the data. This DataSet is then used by a DataAdapter (or other objects) to update the original data source.

The DataSet has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a DataSet with a schema can actually be compiled for type safety and statement completion.

DataAdapters (OLEDB/SQL):

The DataAdapter object works as a bridge between the DataSet and the source data. Using the provider-specific SqlDataAdapter (along with its associated SqlCommand and SqlConnection) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the OleDbDataAdapter object and its associated OleDbCommand and OleDbConnection objects.

The DataAdapter object uses commands to update the data source after changes have been made to the DataSet. Using the Fill method of the DataAdapter calls the SELECT command; using the Update method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a CommandBuilder object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the DataSet and DataAdapter, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a DataSet in order to insert, update, or delete it.
6. Also, you can use a DataSet to bind to the data, move through the data, and navigate data relationships.

**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 raw 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.

**SQL Server Tables:**

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

**Primary Key**

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

**Relational Database:**

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

**Foreign Key:**

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

**Referential Integrity:**

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

**Data Abstraction:**

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

**Physical level:**

This is the lowest level of abstraction at which one describes how the data are actually stored.

**Conceptual Level:**

At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

**View level:**

This is the highest level of abstraction at which one describes only part of the database.

**Advantages of RDBMS:**

1. Redundancy can be avoided
2. Inconsistency can be eliminated
3. Data can be Shared
4. Standards can be enforced
5. Security restrictions ca be applied
6. Integrity can be maintained
7. Conflicting requirements can be balanced
8. Data independence can be achieved.

**Disadvantages of DBMS:**

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

**FEATURES OF SQL SERVER (RDBMS):**

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today’s most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

SQL SERVER is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.

SQL SERVER RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application.

SQL SERVER with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

1. The row level lock manager

**Enterprise wide Data Sharing:**

The unrivaled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

**Portability:**

SQL SERVER is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database sever platform that meets the system requirements.

**Open Systems:**

SQL SERVER offers a leading implementation of industry –standard SQL. SQL Server’s open architecture integrates SQL SERVER and non –SQL SERVER DBMS with industries most comprehensive collection of tools, application, and third party software products SQL Server’s Open architecture provides transparent access to data from other relational database and even non-relational database.

**Distributed Data Sharing:**

SQL Server’s networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

**Unmatched Performance:**

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.

**Sophisticated Concurrency Control:**

Real World applications demand access to critical data. With most database Systems application becomes “contention bound” – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access . SQL Server employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

**No I/O Bottlenecks**

SQL Server’s fast commit groups commit and deferred write technologies dramatically reduce disk I/O bottlenecks. While some database write whole data block to disk at commit time, SQL Server commits transactions with at most sequential log file on disk at commit time, On high throughput systems, one sequential writes typically group commit multiple transactions. Data read by the transaction remains as shared memory so that other transactions may access that data without reading it again from disk. Since fast commits write all data necessary to the recovery to the log file, modified blocks are written back to the database independently of the transaction commit, when written from memory to disk.

**NORMALIZATION**

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

Insertion anomaly: Inability to add data to the database due to absence of other data.

Deletion anomaly: Unintended loss of data due to deletion of other data.

Update anomaly: Data inconsistency resulting from data redundancy and partial update

Normal Forms: These are the rules for structuring relations that eliminate anomalies.

**First Normal Form:**

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

**Second Normal Form:**

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1. Primary key is a not a composite primary key
2. No non key attributes are present
3. Every non key attribute is fully functionally dependent on full set of primary key.

**Third Normal Form:**

A relation is said to be in third normal form if their exits no transitive dependencies.

Transitive Dependency: If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

**Chapter 8**

**8.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.

8.1 TESTING APPROACHES

**8.1.1 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.

**8.1.2 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.

8.2 TEST CASES

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases.

A representative sample to tests that will exercise all software functions.

Additional tests that focus on software functions that are likely to be affected by the change.

**8.2.1 Validation Testing:**

Validation testing demonstrates the traces the requirements of the software. This can be achieved through a series of black box tests.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Id | Test | Inputs | Actual Output | Obtained Output | Description |
| 1 | patient  Valid Login | Uid & pwd | Success | Success | Test passed. Passes the control to the Other Module Menus. |
| 2 | patient  inValid Login | Uid&pwd | Failed | Failed | Test Passed.  Passes the control to the Error Page with appropriate message |
| 3 | Doctor  Valid Login | Uid & pwd | Success | Success | Test passed. Passes the control to the Other Module Menus. |  |
| 4 | patient  inValid Login | Uid&pwd | Failed | Failed | Test Passed.  Passes the control to the Error Page with appropriate message |
| 5 | Agent  Valid Login | Uid & pwd | Success | Success | Test passed. Passes the control to the Other Module Menus. |
| 6 | Agent  inValid Login | Uid&pwd | Failed | Failed | Test Passed. |

**Table8.1:validation testing**

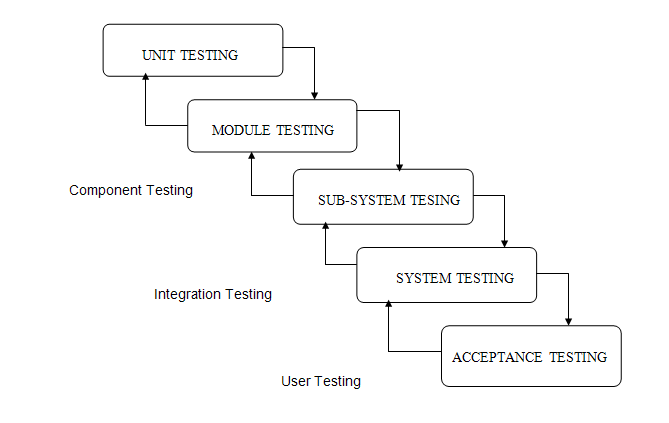
**8.3 TESTING STRATEGIES:**

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.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing 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. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be preformed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.



**Fig 8.1: testing strategies**

**8.3.1 UNIT TESTING:**

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

**8.3.2 SYSTEM TESTING:**

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths through out the system to make sure that no path is making its way into chaos.

**8.3.3 INTEGRATION TESTING:**

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an end less loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.

Different incremental integration strategies are top – down integration, bottom – up integration, regression testing

**Chapter 9**

**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 incorpate 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.

**Chapter 10**

**10.APPENDIX**

**10.1 SOURCE CODE**

**Agentreport.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.IO;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApplication4

{

public partial class Agentreport : Form

{

public Agentreport()

{

InitializeComponent();

}

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

{

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void linkLabel3\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

NewUser patinew = new NewUser();

patinew.Show();

linkLabel3.Visible = true;

this.Hide();

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Alladmin al = new Alladmin();

al.Show();

this.Hide();

}

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;

}

private void button1\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection(@"server=NEOAPP21\SQLEXPRESS;database=privacy;user id=sa;password=neoapp123");

con.Open();

string cmd = "select pname as 'Patient Name',age as 'Age',phoneno as 'Contact Number',gender as 'Sex', dtime as 'Registertime' from patient";

DataSet ds = new DataSet();

SqlDataAdapter sda = new SqlDataAdapter(cmd, con);

sda.Fill(ds);

dataGridView1.DataSource = ds.Tables[0].DefaultView;

}

}

}

**DoctorRegister.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;

using System.IO;

using System.Data.SqlClient;

namespace WindowsFormsApplication4

{

public partial class Doctorregisters : Form

{

public static string randamkeys;

SqlConnection conn = new SqlConnection(@"server=NEOAPP21\SQLEXPRESS;database=privacy;user id=sa;password=neoapp123");

public Doctorregisters()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

randamkeys = Path.GetRandomFileName();

string name = textBox1.Text;

string age = textBox2.Text;

string gender = comboBox1.SelectedItem.ToString();

string department = textBox3.Text; ;

if (conn.State == ConnectionState.Closed)

conn.Open();

SqlCommand cmd = new SqlCommand("insert into doctors (dname,age,gender,department,password,dtime) values('" + name+ "','" + age+ "','" + gender+ "','"+department+"','" + randamkeys + "','"+DateTime.Now+"')", conn);

cmd.ExecuteNonQuery();

//cmd.ExecuteNonQuery();

MessageBox.Show("Your ID is" + randamkeys);

textBox6.Text = randamkeys;

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Doclogin hosd = new Doclogin();

hosd.Show();

linkLabel1.Visible = true;

this.Hide();

}

private void linkLabel2\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Alladmin al = new Alladmin();

al.Show();

this.Hide();

}

}

}

**Pdoctordetails.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;

using System.IO;

using System.Data.SqlClient;

namespace WindowsFormsApplication4

{

public partial class pdoctordetails : Form

{

SqlConnection conn = new SqlConnection(@"server=NEOAPP21\SQLEXPRESS;database=privacy;user id=sa;password=neoapp123");

public pdoctordetails()

{

InitializeComponent();

}

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

{

//label2.Text = Exploitinglogin.name;

label8.Text = patient.password;

}

private void docpid\_txt\_TextChanged(object sender, EventArgs e)

{

}

private void SubmitPId\_btn\_Click(object sender, EventArgs e)

{

conn.Open();

SqlCommand cmd = new SqlCommand("Select pname,age,disease,secretkey From patientupdate Where password='" + textBox1.Text + "'", conn);

SqlDataReader dr;

dr = cmd.ExecuteReader();

try

{

if (dr.HasRows==false)

{

MessageBox.Show("Enter correct ID");

}

if (dr.Read())

{

textBox2.Text = dr["pname"].ToString();

textBox3.Text = dr["age"].ToString();

textBox4.Text = dr["disease"].ToString();

textBox5.Text = dr["secretkey"].ToString();

}

else

{

//label1.Text = "Password is incorrect";

//MessageBox.Show("password is incorrect");

}

}

finally

{

//label1.Text = "user name is incorrect";

}

conn.Close();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

conn.Open();

SqlCommand cmd = new SqlCommand("Select diseasedetails,secretkey From patientupdate Where password='" + Pidkey\_txt.Text + "'", conn);

SqlDataReader dr;

dr = cmd.ExecuteReader();

try

{

if (dr.HasRows)

{

while (dr.Read())

if (docpid\_txt.Text == dr["secretkey"].ToString())

{

textBox6.Text=dr["diseasedetails"].ToString();

}

else

{

//MessageBox.Show("password is incorrect");

//txtkey.Text = "";

}

}

}

finally

{

//label1.Text = "user name is incorrect";

}

conn.Close();

}

private void linkLabel2\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

patient p = new patient();

p.Show();

this.Hide();

}

}

}

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;

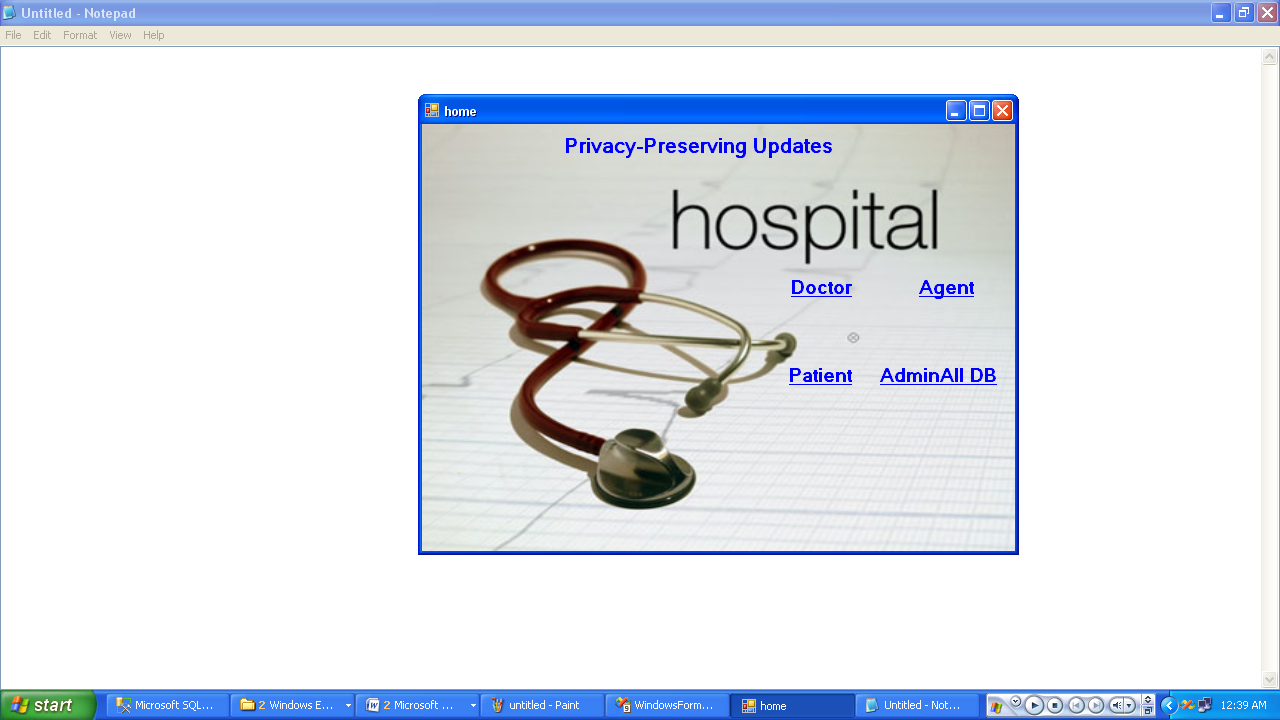
}

}

}

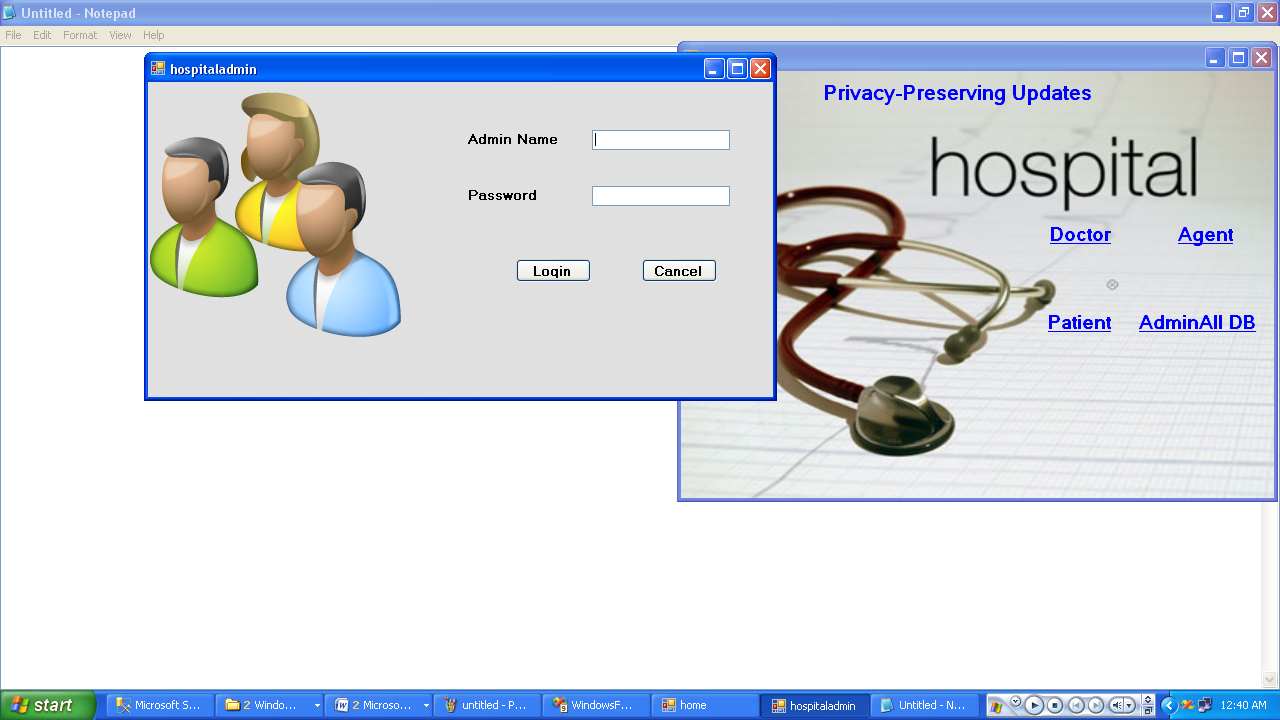
**10.2 SCREEN SHORTS**

**Home Page:**



**Fig 10.1:Home Page**

**Admin login:**



**Fig 10.2:Admin Login Page**

**Admin Home Page:**



**Fig 10.3:Admin Home Page**

**Agent Reports:**



**Fig 10.4:Agent Reporting Page**

**Doctor Login:**



**Fig 10.5:Doctor Login Page**

**Doctor Home Page:**



**Fig 10.6:Doctor Home Page**

**Chapter 11**

**11.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 Organisations,” 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 PrivacyProtection 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 SecurityInformatics 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 inCollaborative Environments via Relational Analysis of AccessLogs,” Proc. First ACM Conf. Data and Application Security 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 AnomalyDetection 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, “Behavioral

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

**Chapter 12**

**12.BIBILOGRAPHY**

* FOR .NET INSTALLATION

[www.support.mircosoft.com](http://www.support.mircosoft.com)

* FOR DEPLOYMENT AND PACKING ON SERVER

[www.developer.com](http://www.developer.com)

[www.16seconds.com](http://www.15seconds.com)

* FOR SQL

[www.msdn.microsoft.com](http://www.msdn.microsoft.com)

[www.asptoday.com](http://www.asptoday.com)

[www.aspfree.com](http://www.aspfree.com)