Design and Prototypical Implementation of a Web Application: Network Log Correlation System

An Internship Report Submitted in fulfilment of the requirements for the award of the degree of Master in Computer Applications

Student Name:Md. Tariq Salman
ID#15165001, MCA, Fall 2016, BRAC University.

Supervisor: AmitabhaChakrabarty, Ph.D
Assistant Professor
CSE Department, BRAC University

Department of Computer Science and Engineering

BRAC University

December 2016

Abstract

Communication industry continues its growth along with innovations every day, sometimes it is treated as the world's biggest machine. However a communication network includes a large number of equipment functions consistently getting the uninterrupted services. An operating system software running on those equipment has the capability to log every event triggers during its runtime. Usually this logs are stored in a database maintained by the Network Operation Centers. One telecommunication network's operation center receives a huge number of logs or events every hour. Each event has some specific actions relevant to its resolution. Here we get a relation between events and actions, therefore every event log follows an action or a combined actions towards the resolutions. In this analysis we used historical resolution database to get the recommended course of actions for an event logged into the database from a network platform that will have an opportunity to reuse of an intellectual information base and will drive a faster way out to any problematic event of any service provider network platform.

Acknowledgements

All those persons who contributed in this work during the last few months we would like to thank them all.

This internship report topic was proposed and realized following the visionary ideas of Dr. AmitabhaChakrabarty. We are very grateful for his valuable feedbacks and the constant motivation that he has extended to us ever since we started our post graduate education at BRAC University.

We would also like to thank all our friends and class mates who helped us a lot during our studies at the university.

Beyond friends there are families, we thank them very much for their love and support for us no matter what we decided to do and also for the financial support.

Table of Contents

Chapter 1: Introduction

Every communication network has a series of devices interconnected which are running round the clock to keep the services available always, therefore a challenge for network engineers to identify any problematic events occurrence with manual interpose. Considering the context we introduce this automated system to isolate device logs base on its different category to reduce manual efforts towards the resolution.

1.1 Background

Network Log Correlation System simplify and consolidate event and impact management, improve service performance and availability, reduce operational costs, and minimize business risk.

NLCS Event and Impact Management helps to minimize service disruptions in business and users through advanced analytics that automatically correlate and prioritize events across multiple technologies, platforms, vendors, and data sources.

- Increase operational efficiencies and lower costs for event, impact, availability, and performance management for physical, virtual, and cloud environments.

- Reduce MTTR and increase service levels with high-precision root cause and service impact analytics that drive automated triage and repair.

- Drive business value by automating solution process flow, such as incident and change management, across the organization and third-party solutions.

Specifically within the context of NLCS the objectives of this paper is to-

- Identify the major challenges to develop an application from software engineering perspective.
- Explore the prospect of NLCS app to helpoperational challenges for Bangladeshi Network Service Providers.
- Develop system analysis and design of a NLCS application.
- Create a prototypical implementation of the application to demonstrate how such application can help increase service level.

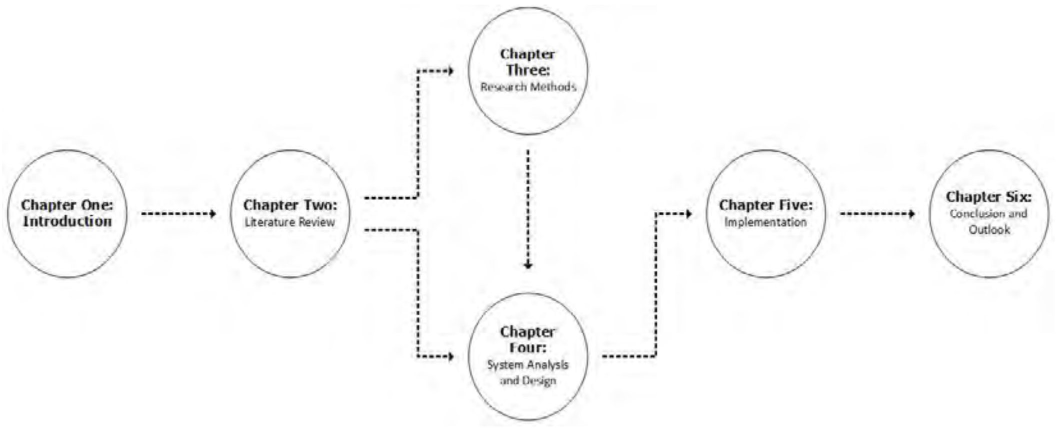

Chapter One: Introduction
Chapter Two: Literature Review
Chapter Three: Research Methods
Chapter Four: System Analysis and Design
Chapter Five: Implementation
Chapter Six: Conclusion and Outlook

Figure 1.1 Report Outline

Figure 1.1 shows a quick overview of the report structure. To meet the first two objectives, a literature review was conducted, of which the results can be found in Chapter 2. The first part of the literature review focuses on overall communication networkoperations and management areasalong with the factors drives into NLCS scenario and the second part focuses into software engineering challenges to develop an application identified by current researches. Chapter 3 states the methodological approach to this paper. The functional and non-functional requirements as well as the core use cases of the proposed tool are introduced in Chapter 4. The implementation of the application, including the data model, system design and user interface evolvement, issubject of Chapter 5. Chapter 6 concludes the report by discussing the findings and proposing future research endeavors in the area correlation.

Chapter 2: Literature Review

In this section we have searched and analyzed different engineering management processes, tool and techniques to organize our case more efficiently and effectively.

2.1 Telecommunication Service Engineering: Definition, Architecture and Tools

2.1.1 Introduction to Telecommunications Services Engineering

The demand for advanced telecommunications services has increased enormously over the last few years. This has led to situations where network operators must deploy new services at a rapid pace when satisfying customer needs. The telecommunications monopolies have disappeared, and the fight for market shares has become fiercer than ever before. Furthermore, the demand for ever more specialized end-user services keeps growing, along with the demand for having the new services deployed within shorter and shorter time frames. The structure and function of networks must change, in order to cope with these new challenges. The telecommunications industry is witnessing a changeover from being interconnection driven to being service driven. A new discipline called telecommunications services engineering is emerging. It encompasses the set of principles, architectures, and tools required to tackle activities ranging from service specification to service implementation, service deployment, and exploitation.[1]

2.1.2 Service Engineering Definition

Since the early 1980s, the major trend in the area of service provision has been toward dissociating service control from the underlying network equipment. As a result, services have been seen as sets of interactions among software pieces running on top of the network infrastructure. Consequently, the concepts, principles, and rules of service engineering were borrowed to a large extent from the software engineering area.

The telecommunications community was faced with a new challenge, which was to bring the telecommunications specific requirements together with software engineering. Interests grew to integrate results from other disciplines such as security, verification and validation, database management systems, communication management, etc.[1]

Service engineering can be defined as the set of methods, techniques, and tools to specify, design, implement, verify, and validate services that meet user needs and deploy and exploit these services over the current or future networks. Service engineering is a young discipline, but is a discipline in itself, as is protocol engineering.
Three important components are considered within the framework of service engineering (Figure 2.1):

- *Service creation environment:* A software engineering platform specialized for the development of telecommunications services.
- *Telecommunications network:* Contains the transmission and switching equipment. Each of thesepieces of equipment may be seen as one black box that offers an application programming interface(API); this may be a signaling or management interface.
- *Network architecture:* Responsible for controlling the network in such a way that a service's specificrequirements get satisfied.

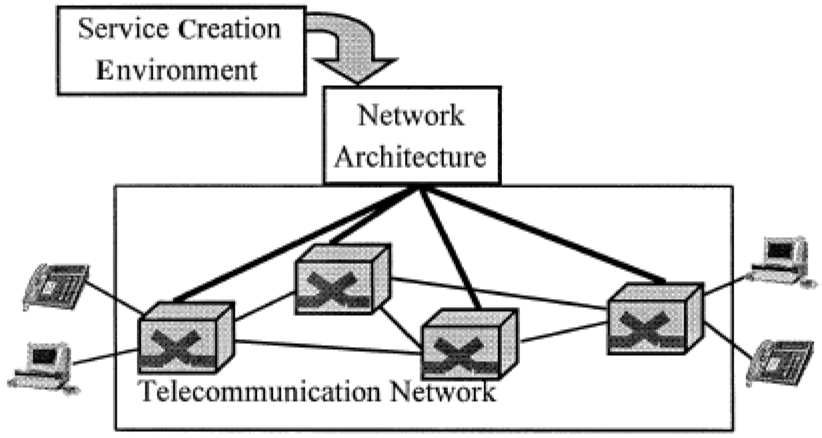


Figure 2.1: Components of telecommunications services by ITU

2.2 The Telecommunications Management Network (TMN)

ITU-T has defined the Telecommunications Management Network (TMN). TMN enables federation of the equipment that constitute the telecommunications network, produced generally by different telecommunications vendors, to enable their control in a uniform, global, and efficient way.

Management of telecommunications networks may be defined as the set of activities of *monitoring*, *analysis*, *control*, and *planning* of the operation of telecommunications network resources to provide services to customers with a certain level of quality and cost.[2]

- Monitoring is defined as the process of dynamic collection, interpretation, and presentation of information concerning objects under scrutiny.
- It is used for general management activities which have a permanent continuous nature such as the systems management functional areas (i.e., performance, configuration, fault, accounting, security).

- Analysis is applied to monitoring information to determine average or mean variance values of particular status variables. Analysis is application specific. It can range from very simple gathering of statistics to very sophisticated model-based analysis.
- Control is the process by which changes in the managed network are effected.
- Planning is defining the network topology and sizing every network element in order for the user to obtain any given service in optimal conditions with regard to quality and price.

2.2.1 **Network Operating System (NOS):** Systems software that controls the computer systems and devices on a network and allows them to communicate with each other.

2.2.2 **Network Management Software:** Enables a manager on a networked desktop to monitor the use of individual computers and shared hardware, scan for viruses, and ensure compliance with software licenses.

2.2.3 **Operations support systems (OSS):** or operational support systems in British usage, are computer systems used by telecommunications service providers to manage their networks (e.g., telephone networks). They support management functions such as network inventory, service provisioning, network configuration and fault management.[2]

Together with business support systems (BSS), they are used to support various end-to-end telecommunication services. BSS and OSS have their own data and service responsibilities. The two systems together are often abbreviated OSS/BSS, BSS/OSS or simply B/OSS.

The acronym OSS is also used in a singular form to refer to all the Operations Support Systems viewed as a whole system.

Different subdivisions of OSS have been proposed by the TM Forum, industrial research labs or OSS vendors. In general, an OSS covers at least the following five functions [2]:

- Network management systems
- Service delivery

- Service fulfillment, including the network inventory, activation and provisioning
- Service assurance
- Customer care

2.2.4 **Communication software:** Provides error checking, message formatting, communications logs, data security and privacy, and translation capabilities for networks.

2.3 Log Correlation Software Engineering

Based on the best factors to application development, we outline the following fundamental, unique challenges to the state-of practice in mobile application software engineering:

2.3.1 Creating Universal User Interfaces

User interface is the front-end application view to which user interacts in order to use the software. User can manipulate and control the software as well as hardware by means of user interface. Today, user interface is found at almost every place where digital technology exists, right from computers, mobile phones, cars, music players, airplanes, ships etc.

User interface is part of software and is designed such a way that it is expected to provide the user insight of the software. UI provides fundamental platform for human-computer interaction.

UI can be graphical, text-based, audio-video based, depending upon the underlying hardware and software combination. UI can be hardware or software or a combination of both.

The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time

- Clear to understand
- Consistent on all interfacing screens

There are a number of activities performed for designing user interface. The process of GUI design and implementation is alike SDLC. Any model can be used for GUI implementation among Waterfall, Iterative or Spiral Model [3].

A model used for GUI design and development should fulfill these GUI specific steps.

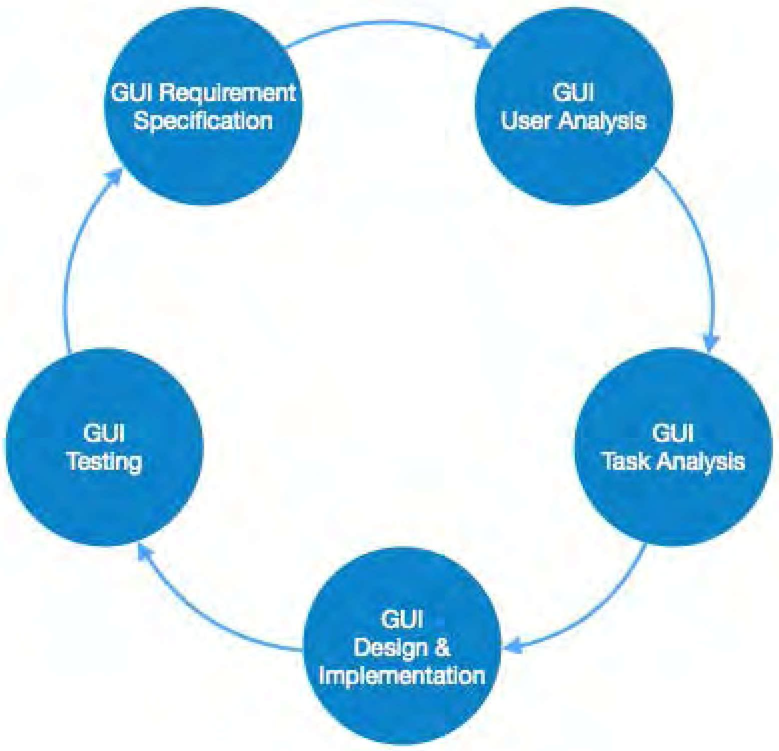


Figure 2.3.1: User Interface design Activity Model

2.3.2 Enabling information Reuse across application Platforms

Reuse is highly dependent on an established product, heritage design or experience, well documented processes, and personnel with knowledge of the product. Therefore, the successful development of a product as well as the reuse of a product may share common factors. They both rely on an existing product foundation, experience that the product functions correctly, a development process that is repeatable, and the intellectual capacity to both understand the existing product and develop the new product. Therefore, this investigation has identified successful reuse in the systems engineering domain as a function of three project factors [5]:

a) Platform

b) People

c) Processes

Platform refers product or technology that is intended to be reused. Fundamentally, for a reuse program to be successful, the product needs to capable of being reuse. Successful reuse can be defined as instances where the benefits of reusing a product outweigh the costs. While Principle #4 states anything can be reused, not everything can be reused successfully.

People are the available personnel for the reuse project and their knowledge of the heritage/new products. Personnel needs to be knowledgeable enough to avoid the pitfalls associated with reuse as well as have a familiarity with both the heritage and new products. Because of this, successful reuse may require an organization to take experienced individuals, ones who are able to see the whole picture but still focus on a particular detail, such as a true systems engineer, and ensure their knowledge is captured.

Processes encompass the organization, documentation, and production. The organization needs to have the flexibility for reuse, possibly utilizing a matrix structure. Documentation needs to be available from heritage products that are being reused and new projects should be well documented to enable reuse in the future. Lastly, production capabilities need to exist so the system can actually be developed and eventually delivered. These are three possible drivers to capture the likelihood of successful reuse and potentially indicate the expected savings of such a strategy.

In addition, reuse may have the most leverage when applied early in the development of a system, such as during the architecture design phase, one of the earliest systems engineering activities. By the time some of the other systems engineering activities are occurring, the "window of opportunity" for reuse may be limited and attempts at reuse could inadvertently affect other aspects of the system in the process.

This investigation examined the potential considerations for the successful utilization of reuse in systems engineering of this NLCS Project by reviewing how related domains, such as software and product development, handle reuse. By studying how reuse is addressed in other domains, six principles were identified as key considerations for reuse:

1) Reuse is done for the purpose of economic benefit, intending to reduce schedule, reduce cost, or increase performance
2) Reuse is not free, upfront investment is required
3) Reuse needs to be planned from the conceptualization phase of programs
4) Anything can be reused
5) Reuse is as much of a technical issue as it is an organizational one
6) The benefits of reuse are not linear

In addition to identifying these six principles, three factors appeared to contribute significantly to the successful utilization of reuse [5]:

a) Platform – appropriate product or technology, primed for reuse
b) People – adequate knowledge and understanding of both the heritage and new products
c) Processes – sufficient documentation to acquire and capture knowledge applicable to reuse as well as the capability actually deliver a system incorporating or enabling reuse

2.3.3 Designing event driven Applications

Event-driven architecture (EDA), also known as message-driven architecture, is a software architecture pattern promoting the production, detection, consumption of, and reaction to events.

An event can be defined as "a significant change in state". From a formal perspective, what is produced, published, propagated, detected or consumed is a (typically asynchronous) message called the event notification, and not the event itself, which is the state change that triggered the message emission. Events do not travel, they just occur. However, the term event is often used metonymically to denote the notification message itself, which may lead to some confusion.

This architectural pattern may be applied by the design and implementation of applications and systems which transmit events among loosely coupled software components and services. An event-driven system typically consists of event emitters (or agents), event consumers (or sinks), and event channels. Emitters have the responsibility to detect, gather, and transfer events. An Event Emitter does not know the consumers of the event, it does not even know if a consumer exists, and in case it exists, it does not know how the event is used or further processed. Sinks have the responsibility of applying a reaction as soon as an event is presented. The reaction might or might not be completely provided by the sink itself. For instance, the sink might just have the responsibility to filter, transform and forward the event to another component or it might provide a self-contained reaction to such event. Event channels are conduits in which events are transmitted from event emitters to event consumers. The knowledge of the correct distribution of events is exclusively present within the event channel. The physical implementation of event channels can be based on traditional components such as message-oriented middleware or point-to-point communication which might require a more appropriate transactional executive framework. Building applications and systems around an event-driven architecture allows these applications and systems to be constructed in a manner that facilitates more responsiveness, because event-driven systems are, by design, more normalized to unpredictable and asynchronous environments.

In this case Event-driven architecture can complement service-oriented architecture (SOA) because services can be activated by triggers fired on incoming events. This paradigm found useful whenever the sink does not provide any self-contained executive.

This new NLCS pattern triggers further autonomous human or automated processing that adds exponential value to the enterprise by injecting value-added information into the recognized pattern which could not have been achieved previously.

2.3.4 Balancing Agility and Uncertainty in Requirements

While most application developers utilize an agile approach or a nearly ad hoc approach, the growing demand for context-aware applications, competition amongst applications and low tolerance by users for unstable and/or unresponsive mobile applications (even if free) necessitates a more semi-formal approach. This should be integrated into agile engineering to specify and analyze application requirements. The dynamic, contextual nature of application content (e.g., System Management) allows for situations in which the application's behavior may not be able to fully satisfy the specified functional and non-functional requirements thereby necessitating that the application be self-adaptive. In this scenario the software will then provide less rich content satisfying less stringent requirements. For some Network applications, this may arise if, as determined in the requirements, it is better for the application to run continuously and, when necessary, to autonomously modify its behavior and provide reduced functionality rather than provide

no functionality at all. For example, in an OSS based application several factors (e.g., accuracy, reporting, SLA etc.) may affect the granularity and accuracy of its content.
Within application software engineering, the need for an application to self-adapt, depending on context, has been constructed using ad hoc approaches. Yet, as OSS applications become more context aware, self-adaptive requirements will need to be more formally integrated into agile development so that developers more rigorously consider the behavior of an application when its full requirements cannot be satisfied dynamically and how it can self-adapt to partially satisfy the requirements.[4]

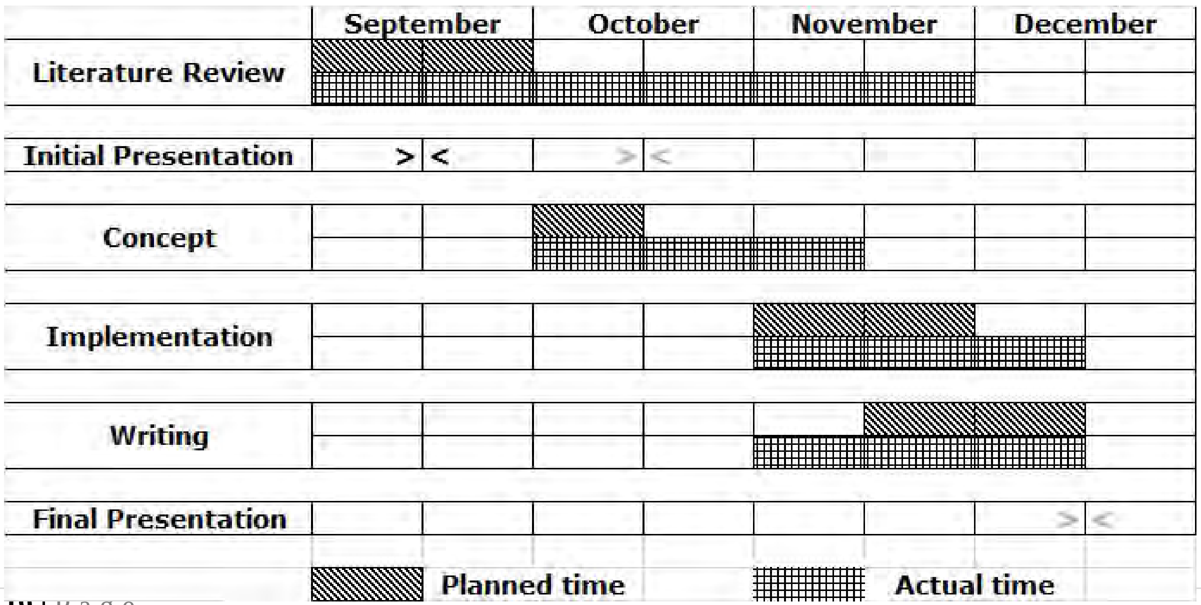
This paper briefly described few challenges that we see for application software engineering: designing correlation logic, supporting context-aware applications and balancing agility with specifying requirements uncertainty. This paper asserts that application software engineering research efforts need to focus on development approaches emphasizing UI design, proactive reuse at early software engineering phases, and attention to context-awareness and sensitivity to specifying requirements to handle requirements uncertainty within the existing agile development approaches used for development applications. In addition, software engineering research needs to emphasize education initiatives in these areas to ensure that these approaches are disseminated to those doing actual application development.

Chapter 3:Research Methods

In this section we mentioned the phases we followed to plot our requirements with different other project documentations and methods suitable for an implementation segments of our requirements.

3.1 Methods

The basic timeline of the process of writing this paper is depicted in below table. The official start date was September'16 and the date of the submission was by December'16. Originally, a waterfall-like step-by-step approach was laid out where one part of the paper would be completed before starting another part. However, mainly due to a vast amount of interesting literature, many of these time periods had to be extended and moved.



	September		October		November		December	
Literature Review	▧	▧						
	▦	▦	▦	▦	▦	▦		
Initial Presentation	>	<	>	<				
Concept			▧					
			▦	▦	▦			
Implementation					▧	▧		
					▦	▦	▦	
Writing						▧	▧	
					▦	▦	▦	
Final Presentation							>	<

▧ Planned time ▦ Actual time

Table 3.1 Gantt chart of the research process

In the above diagram an overview of the research process is shown in a Gantt chart. The following sub-sections further detail how the particular areas of this internship project were executed.

3.2 Literature Review

At the beginning of the research project, a literature search was conducted. Based on possible search terms and key word combinations using Google. In the below diagram the process of filtering out suitable research publications from a list of search results was shown. First, the title of each search result was examined in order to determine whether the paper could have a possible connection to this particular internship project and the research questions. Publications fitting into the research topic were added to the search log. Then, the abstracts of all log entries were further analyzed and not suitable papers were marked as such and excluded from any further analysis. The next step for the residual log entries was to locate the articles using the on- and offline access possibilities. However, quite a number of potentially interesting publications could not be accessed and were therefore eliminated. After skimming the content of all remaining log entries, a last few were excluded and the rest was used to construct the literature review found in section 2.

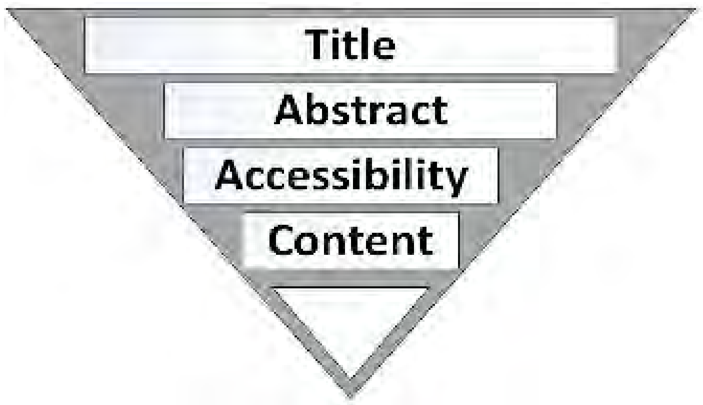


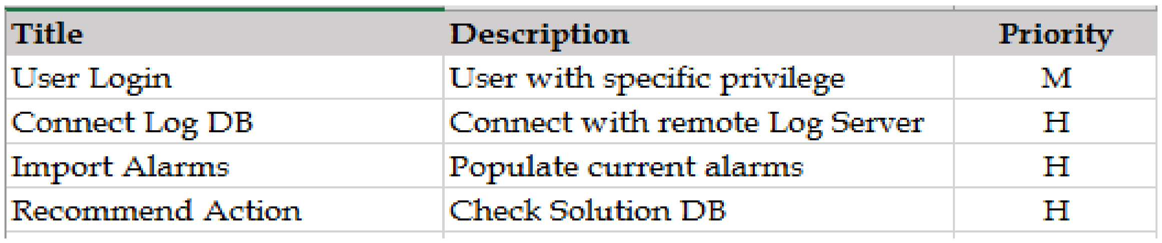
Figure 3.1 The process of filtering literature search results

3.2.1 Requirements Elicitation

The requirements that will be presented in section 4 were based upon the functionality of already existing medical healthcare applications and approaches. The Ionic Framework and various applications were analyzed with respect to their functionalities, capabilities and limitations. These tools all implement four main use cases: User login, connecting log Database, Populating Alarms, Provide Recommended Solution.

3.2.2 Concept and Implementation

Based on the requirements elicited in Chapter 4, the basic system design, data structure and user interface were created (see Chapter 5 for more details). A prioritized backlog of all needed functionalities was generated (Table below) and its elements were continuously implemented until all were fulfilled.



Title	Description	Priority
User Login	User with specific privilege	M
Connect Log DB	Connect with remote Log Server	H
Import Alarms	Populate current alarms	H
Recommend Action	Check Solution DB	H

Table 3.2 Basic backlog (priority L-M-H)

During the time of conception and implementation of the prototype, weekly meetings with the supervisor acted as small review meetings of the current development status. Drawbacks of user interface elements, ideas for the database to be used as well as re-prioritizations of certain backlog items were discussed during the meetings.

Chapter 4: System Analysis and Design

This section introduces the minimal requirements along with the internal relations to achieve expected outcomes for the prototypical implementation of a Network Log Correlation System.

4.1 Requirements

We identified functional and non-functional requirement through a study on use cases where different actors have some course of actions and defined roles to perform the actions.

4.1.1 Roles

The system should be developed to support minimum three types of roles. On one side we have general user of the application who monitors the notification. On the other side we have support engineers who interact with the application to send alarm resolution to Solution DB. Another role is for administration purpose whose primary purpose is to add, update and remove database and provide support to application users using comments.

4.1.2 Use Cases and Functional Requirements

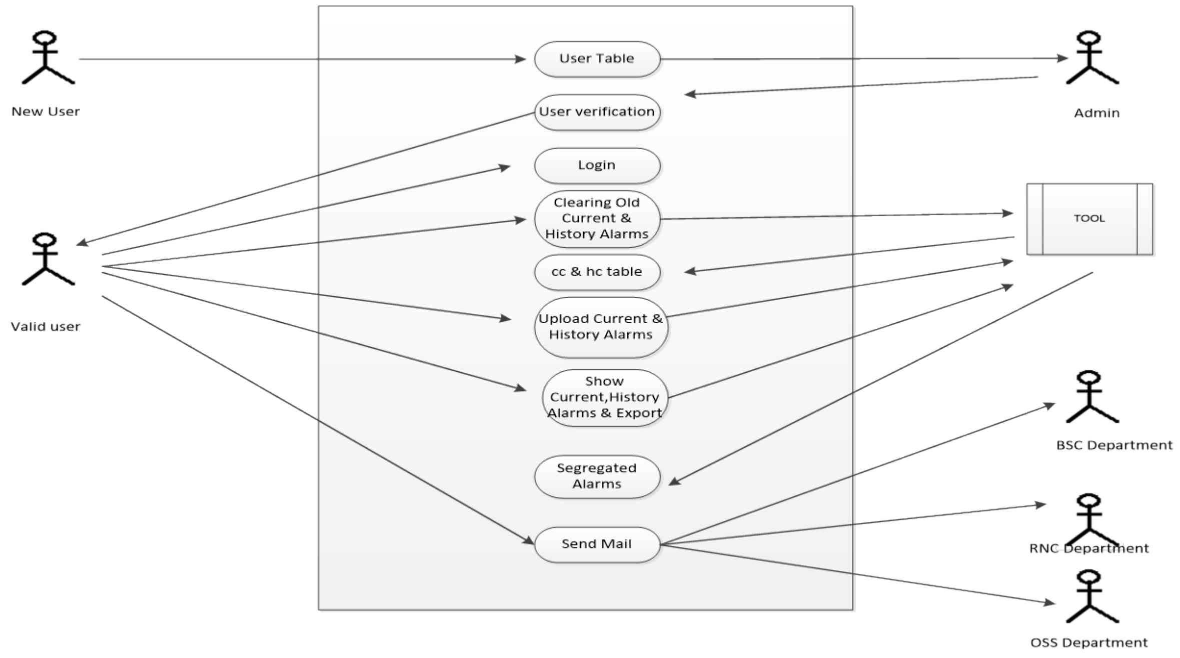


Figure 4.1 Application Use Cases

There are six main use cases that are mostly based on information seeking model from the perspective of a general user as shown in Figure 4.1 These use cases all provide a way to add, update or delete Alarm database entities, to search for alarms entities, to provide feedback for Alarm entities. Lastly a registration and log in system is necessary when storing user-specific data to differentiate among various roles described in the use case.

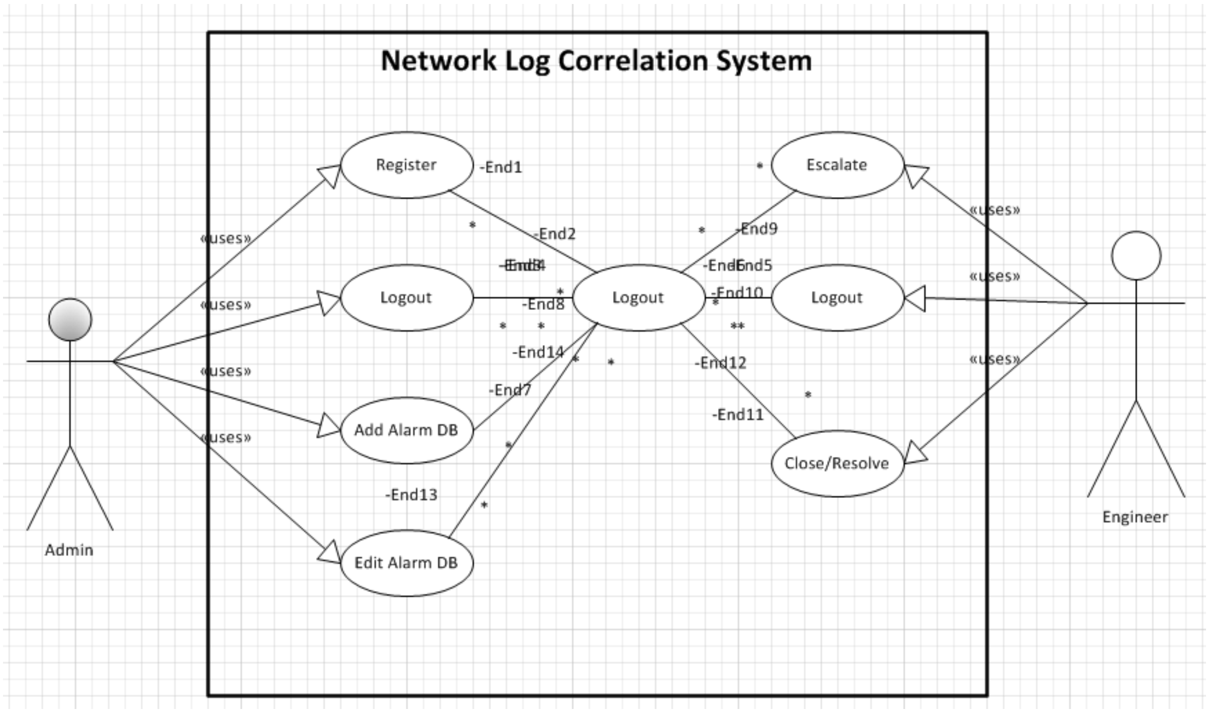


Figure 4.2 Use Cases

In above diagram use cases relevant for the Web Portal is shown there are two different actors for the web portal and they are the admin user and the Engineers. An admin user has four major use cases whereas the three primarily has three major use case.

4.1.3 Registration and Login

For any activities, the user is required to have an account and be logged in. In order to register for an account, the user must enter a valid email address, a preferred username and a password into corresponding fields or use domain accounts to sign in. After having created an account or entered a correct email-password pair, the user will be logged into the application and stay logged in.

4.1.4 Connect Alarm Databases

Users should be able to launch the application, login and immediately start looking for suitable Alarm logs in Databases. There should be a search function that allows the user to type in search keywords, searches the database of entities and displays results.

After having found or discovered a databases, the user should be able to view all detail information (see section 4.2.2).

4.1.5 Change or Modify Alarm Databases

Admin users must have the possibility to create a new DB entity. Each DB entity consists of a title describing the name of the entity, a more detailed description of the entity which includes IP address, DB details.

In addition to the ability to add a DB entity, an admin user must also be able to edit a DB entity details to reflect any changes, also the admin user should have the functionality to remove a DB entity necessary. Any newly created DB entity should be uploaded to the back end server immediately to ensure availability of data.

4.1.6 Escalate to Stakeholders

Using the application engineers should be able to escalate alarm with feedback. Once a user complete searching a log alarm he or she can see the detailed view of that particular entity and should be able to edit response for his unit in order to meet SLA.

4.1.7 Alarm Closure

The system should also offer the engineers to close alarm entities. A closing report can also be generated according to SLA.

4.1.8 Non-Functional Requirements

Apart from the functional requirements mentioned above, the system in general must adhere to certain behavioral requirements:

1. Performance: The system shall display an activity indicator when elements take time to be loaded.
2. Availability: The system must be available at all times, except when upgrading or restarting the backend database. In case of no network connection the system shall enable the admin to add, update or delete medical entities to the server once the network connection becomes available again.
3. Usability: Error messages shall only be presented when the user can (try to) perform an action that could potentially fix the erroneous situation. Therefore, error messages shall always include a proposed solution to the problem.
4. Security: The system shall insure that data is protected from unauthorized access. All authorization communication between client and server needs to be encrypted (username and password data).
5. Capacity and Scalability: The system have a modular object base to enhance and integrate with future development.

4.2 Software Development Strategy

The project's software development strategy is based on Agile Software Development disciplines. Against the traditional software development methodologies, the leaders of the software development approaches having similar opinions as Extreme Programming (XP), Scrums, Crystal Methodologies, Feature-Driven Programming formed an alliance and put out the Agile Manifesto in 2001 which is;

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

That is, while there is value in the items on the right, we value the items on the left more."

The main goal is bringing speed and effectiveness in modelling and documentation of developing software systems. Teamwork, collaboration, rapid and continuous delivery of high-quality software are the main principles laying behind agility. Agile approach does not oppose the design activities but it is against to design a software at the beginning of a software project. At the beginning of the project, system is pre-designed. The design is flourished with the evolvement of requirements, code and test results. Design is mentioned as output of the process, rather than being input.

According to the agile approach, tasks have to be broken into small increments with minimal planning. Long-term planning is opposed. All development activities have to be done iteratively. Each iteration of 3-4 weeks is composed of a full software development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of an iteration, a workingsoftware must be delivered to the customer. Stakeholder involvement takes an important place in agile methods. As stated in the principles of the Agile Manifesto, "Business people and developers must work together daily throughout the project." This method is believed to improve the requirements analysis and finding the defects earlier, which increases software quality. To gain agility to the process, tool usage is encouraged. Tools for unit and functional testing and techniques such as continuous integration, pair programming, test driven development, design patterns, domain-driven design, code refactoring are often used to improve quality.

4.2.1 Planning Phase

In this paper, only the first iteration is ensample as a cross cut of the overall system. The project plan is prepared with respect to agile project planning approach and the first iteration starts with the requirement analysis phases. The overall context of the system is analyzed and represented as the Data Flow Diagram Level 0, in the following figure-

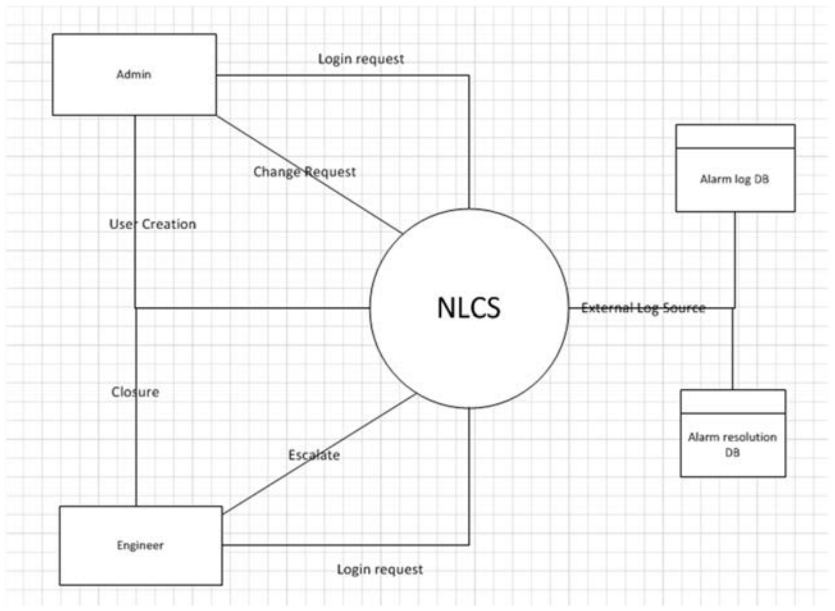


Figure 4.3 Context Diagram

After gaining an understanding of the major functionalities of the system, the requirements are stated as user stories and they are prioritized. Then the prototyping phase initiates. The user interfaces is developed and started to sketch the system. This prototype enabled us to analyze the requirements in a deeper manner. The design of the system is shaped with the development of requirements and the prototype.

4.2.2 Requirement Analysis

After the requirements are gathered, they are prioritized. The first prototype of the system is developed in the first iteration, due to the user stories. The functional requirements are;

1. User can login to the application.
2. User can connect Alarm Databases.

3. User can select an item and view details.
4. User can set an action.
5. User can view resolution.

Non-functional Requirements:

1. Performance:

An application should be fast and efficient. There is a tendency in the computing world these days to assume that Moore's Law will solve all our problems related with performance. It is important to write efficient code for independent platform. Writing fast code means keeping memory allocations to a minimum, writing tight code, and avoiding certain language and programming idioms that can cripple performance. In object-oriented terms, most of this work takes place at the method level, on the order of actual lines of code, loops, and so on.

2. Responsiveness:

The applications that feel slow, hang or freeze for significant periods, or take too long to process input are insufficiently responsive and will frequently cause the system to pop up "Application Not Responding" (ANR) message. Generally, this happens if application cannot respond to user input. For example, if applicationblocks on some I/O operation (frequently a network access), then the main application thread won't be able to process incoming user input events. After a time the system will conclude that your application has hung, and give the user the option to kill it. The fix is usually to create a child thread, and do most of your work there. This keeps the main thread running, and prevents the system from concluding code has frozen.

3. Seamless:

There are some seamless parameters, we should concern while designing and implementing system. These are:

a. Do not drop data:

If the user was editing data in your application when the other Activity appeared, your application will likely lose that data when your application is killed. A classic example of a good use of this behavior is a mail application. If the user was composing an email when another Activity started up, the application should save the in process email as a draft.
c. Got a Lot to do? Take it to a Thread:

If application needs to perform some expensive or long running computation, we should probably move it to a thread. This will prevent the dreaded "Application Not Responding" dialog from being displayed to the user, with the ultimate result being the fiery demise of the application.

d. Avoiding Huge Activities:

Any application worth using will probably have several different screens. When partitioning UI, need to be sure about to make effective use of Activities.

e. Assuming the Network is slow:

We should always code your applications to minimize network accesses and bandwidth. We can't assume the network is fast, so you should always plan for it to be slow.

4.2.3 Design

The first generation of Web applications, dedicated to e-commerce, content publication and management, focused on enabling users to perform simple operations, like searches, data uploads, and browsing of large volumes of data structured in hypertexts. More recently, the Web has become a popular implementation platform for B2B applications, whose goal is not only the navigation of content, but also the enactment of intra- and inter- organization business processes. Web-based B2B applications exhibit much more sophisticated interaction patterns than traditional Web applications: they are developed to support a well-defined process, consisting of activities and their execution constraints, serving different user roles whose joint work is coordinated. They may be distributed across different

processor nodes, due to organizational constraints, design opportunity, or existence of legacy systems to be reused. B2B Web applications demand novel methodologies for their analysis, specification and implementation, because they are more complex than either process based systems, or pure data- centric Web applications.
The application logic for a system will be expressed in programs that will be written during construction of the new system. Program design is the part of the design phase of the SDLC during which analysts determine what programs will be written, create instructions for the programmers about how the code should be written, and identify how the pieces of code will fit together to form a program.[3]
Program design techniques are still very important, however, for two reasons. First, even preexisting code needs to be understood, organized, and pieced together. Second, it is still common for the project team to have to write some (if not all) code and produce original programs that support the application logic of the system.

During design, physical process models are created to show implementation details and explain how the final system will work. These details can include references to actual technology, the format of information moving through processes, and the human interaction that is involved. In some cases, most often when packages are used, the use cases may need to be revised as well. These to-be models describe characteristics of the system that will be created, communicating the "systems view" of the new system.[3]

In this paper we argue that good Web applications should be, first of all, good hypermedia applications, i.e. they should provide easy navigational access to large information resources, preventing users from being lost in the cyberspace, and providing consistent navigation operations even when other kind of transactional behavior is involved. As navigation problems have been largely discussed in hypertext literature (see for example [3]) we should be able to reuse existing knowledge on building good Web applications.

The physical DFD contains the same components as the logical DFD (e.g., data stores, data flows), and the same rules apply (e.g., balancing, decomposition). The basic difference between the two models is that a physical DFD contains additional details that describe how the system will be built. Five steps have followed to perform making the transition to the physical DFD.

To summarize the discussion above, we can intuit that there are distinguishing features in Web applications that present new design requirements vis-à-vis traditional systems. In a broad sense, we can categorize them in three groups. The first group of design issues has to do with navigation, addressing questions such as:

- What constitutes an "information unit" with respect to navigation?
- How does one establish what are the meaningful links between information units?
- Where does the user start navigation?
- How does one organize the navigation space, i.e., establish the possible sequences of information units the user may navigate through?
- If we are adding a WWW interface to an existing system, how do we map the existing data objects onto "information units", and what relationships in the problem domain should be mapped onto links?

The second group of design issues has to do with the organization of the interface, addressing questions such as:

- What interface objects the user will perceive? How do these objects relate to the navigation objects?
- How will the interface behave, as it is exercised by the user?
- How will navigation operations be distinguished from interface operations and from "data processing" (i.e., application operations)?
- How will the user be able to perceive his location in the navigation space?

The third group of design issues has to do with implementation, addressing questions such as:

- How are information units mapped onto pages?
- How are navigation operations implemented?
- How are other interface objects implemented?
- How are existing databases integrated into the application?

We have followed the Object-Oriented Hypermedia Design Method focusing how we build navigational models as views on conceptual models; then we have designed navigational contexts as a structuring mechanism for navigation. Finally, we have reviewed some ongoing work on mining navigation patterns and present some further work on designing and implementing these kind of systems.

The OOHDM Design Framework

The Object-Oriented Hypermedia Design Method [Schwabe 98, Schwabe 96] is a model-based approach for building large hypermedia applications. It has been extensively used to design different kinds of applications such as: Web sites and information systems, interactive kiosks, multimedia presentations, etc. It should be stressed the OOHDM has been applied outside the academic environment, such as in government agencies, telecommunications companies, oil companies, IT service companies, etc.[4]

OOHDM comprises four different activities namely, Conceptual Design, Navigational Design, Abstract Interface Design and Implementation. During each activity a set of object-oriented models describing particular design concerns are built or enriched from previous iterations.

We explicitly separate conceptual from navigation design since they address different concerns in Web applications. Whereas conceptual modeling and design must reflect objects

and behaviors in the application domain, navigation design is aimed at organizing the hyperspace taking into account users' profiles and tasks. Though applications views are not new in the literature [4], the hypermedia paradigm as it appears in the Web raises additional concerns such as orientation, cognitive overhead, etc. that should be treated in a separate design activity.
Considering conceptual, navigational and interface design as separate activities allows us not only to concentrate on different concerns at a time, but mainly to obtain a framework for reasoning about the design process, encapsulating design experience specific to each activity. As we explain below, navigational design is a key activity in the implementation of Web applications and it must be explicitly separated from conceptual modeling.

OOHDM design primitives can be mapped onto non object-oriented implementation settings using some simple heuristics. We next discuss the first three activities in more detail.

Conceptual Modeling

During this step we build a model of the application domain, using well known object-oriented modeling principles and primitives similar to those in UML [3]. The product of this step is a class schema built out of Sub-Systems, Classes and Relationships.

We chose UML because it is a modeling standard whose syntax and semantic are clear and well-understood. The major differences with UML are the use of multiple valued attributes, and the use of directions explicitly in the relationships. Aggregation and generalization/specialization hierarchies are used as abstraction mechanisms.

Conceptual Modeling is aimed at capturing the domain semantics as "neutrally" as possible, with little or no concern for the types of users and tasks. When the application involves some sophisticated behavior in conceptual objects, it may evolve into an object model in the implementation environment. However it can be implemented in a straightforward way in current Web platforms combining for example a relational database with some stored procedures. The main thesis in this paper is that the conceptual model may not reflect the fact that the application will be implemented in the WWW environment, since the key application model will be built during navigational design. This view allows using the same strategy for implementing “legacy” applications in the Web, by considering their conceptual model as the product of this OOHDM activity.

Classes in the conceptual model will be mapped to nodes in the navigational model using a viewing mechanism and relationships will be used to define links among nodes. It will also be shown that there are other links that do not correspond to relationships in the conceptual model.

Using a behavioral object-oriented model for describing different aspects of Web applications allows to express a rich variety of computing activities, such as dynamic queries to an object-base, on-line object modifications, heuristics-based searches, etc. The kind of behavior required in the conceptual model depends upon the desired features of the application. For many Web applications, in particular those implementing plain browsing (i.e. read-only) functionality, class behavior beyond linking functionality is unnecessary and does not need to be specified.

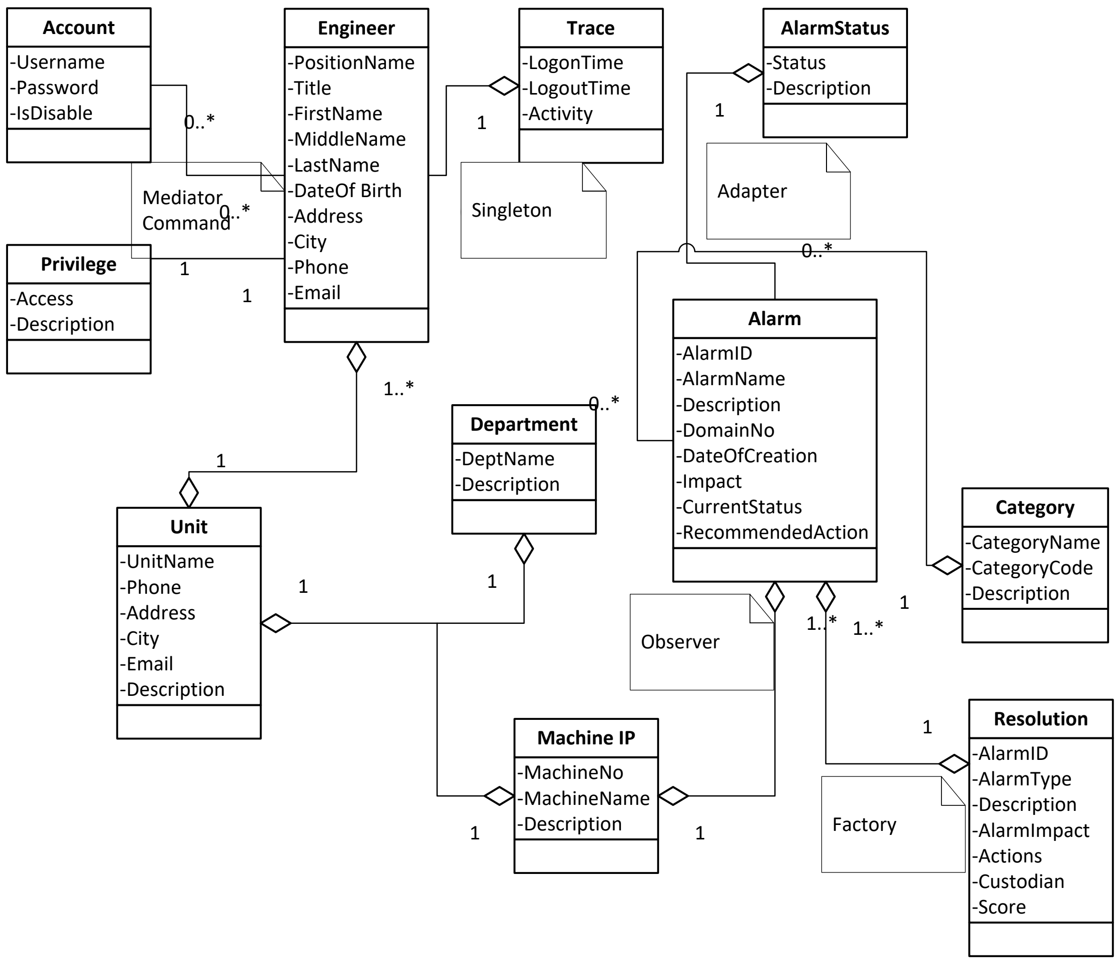


Fig 4.2: Conceptual Schema of NLCS

Navigational Design

In OOHDM, an application is seen as a navigational view over the conceptual model. This reflects a major innovation of OOHDM with respect to other methods, at it recognizes that the objects (items) the user navigates are not the conceptual objects, but other kinds of objects that are "built" (through a view mechanism) from one or more conceptual objects. Moreover, it is important to stress that the user navigates through links, many of which cannot be directly derived from conceptual relationships.

For each user profile we can define a different navigational structure that will reflect objects and relationships in the conceptual schema according to the tasks this kind of user must perform. The navigational class structure of a Web application is defined by a schema containing navigational classes. In OOHDM there is a set of predefined types of navigational classes: nodes, links, anchors and access structures. The semantics of nodes, links and anchors are the usual in hypermedia applications. Access structures, such as indexes, represent possible ways for starting navigation.

The most outstanding difference between our approach and others using object viewing mechanisms is that while the latter consider Web pages mainly as user interfaces that are built by "observing" conceptual objects, we clearly favor an explicitly representation of navigation objects (nodes and links) during design.

Abstract Interface Design

In the Abstract Interface Design activity, we specify which interface objects the user will perceive and how the interface will behave. For each attribute (either contents or anchors) we must define its appearance. By distinguishing between navigation and interface design we can build different interfaces for the same application and besides achieve implementation independence.

During this activity we define the way in which different navigational objects will look like, which interface objects will activate navigation, the way in which interface objects will be synchronized and which interface transformations will take place. In OOHDM, we use the Abstract Data View (ADV) design approach for describing the user interface of a hypermedia application. Though not completely related with the aim of this paper, it is important to stress that building a formal model of the interface of Web applications is a rewarding activity as user interfaces tend to change even faster than navigation topologies. We clearly need a precise design specification to be able to support changes smoothly. Following steps are some matrix to achieve our objective.

Step 1: Adding Implementation References The first step in creating a physical DFD is to begin with the existing logical DFD and add references to the ways in which the data stores, data flows, and processes will be implemented. Data stores on physical DFDs will refer to files and/or database tables; processes, to programs or human actions; and data flows, to the physical media for the data, such as paper reports, bar code scanning, input screens, or computer reports. The names for the various components on the physical DFD should contain references to these implementation details. By definition, external entities on the DFD are outside of the scope of the system and therefore remain unchanged in the physical diagram.

Step 2: Drawing a Human-Machine Boundary The second step is to add a human–machine boundary. Physical DFDs differentiate human and computer interaction by a human-machine boundary, a line drawn on the model to separate human action from automated processes.

Step 3: Adding System-Related Data Stores, Data Flows, and Processes In step 3 adding to the DFD additional processes, stores, or flows that are specific to the implementation of the system and have little (or nothing) to do with the business process itself. These additions can be due to technical limitations or to the need for audits, controls, or exception handling. Technical limitations occur when technology cannot support the way in which the system is modeled logically.

Step 4: Update the Data Elements in the Data Flows The fourth step was to update the elements in the data flows. The data flows will appear to be identical in both thelogical and physical DFDs, but the physical data flows may contain additionalsystem-related data elements, for reasons similar to those described in the previoussection. For example, most systems add system-related data elements to data flowsthat capture when changes were made to information (e.g., a last update dataelement) and who made the change (e.g., an updated by data element). Another physical data element is a system-generated number used to uniquely identify eachrecord in a database. During step 4, the physical data elements are added to themetadata descriptions of the data flows in the CASE repository.

Step 5: Update the Metadata in the Computer-Aided Software Engineering Repository

Finally, the project team needs to make sure that the information about the DFD components in the CASE repository is updated with implementation-specific information. This information can include when batch processes will be run and how often, names of the actual tables or files that are represented by data flows, and the sizes and projected growth rates of the data stores.

Chapter 5: Implementation

In this section we approached on a basic implementation strategy focusing a quality development within a short span of time.

5.1 Data Model

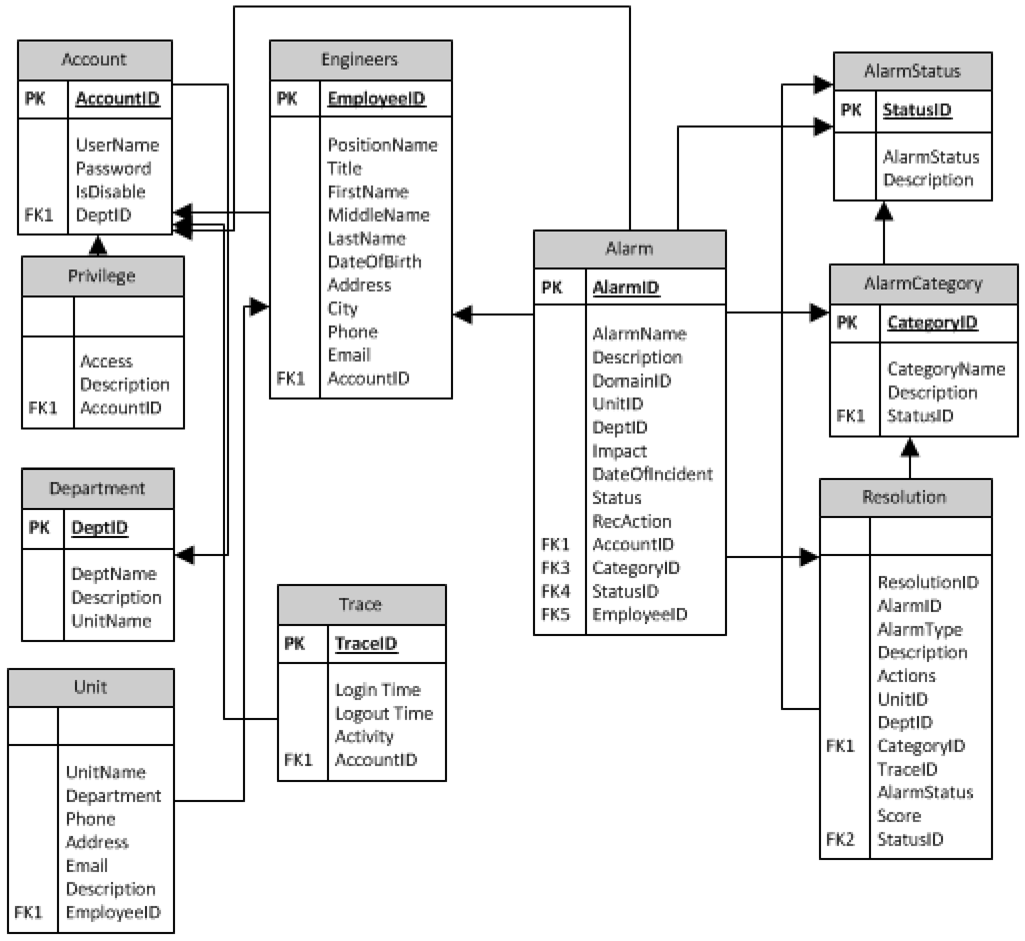


Fig 5.1: The basic data model

5.1.1 User Access & Login:

Here a User access page has been used for log in. Users are created through manual process as privacy needs to be maintainedhighly. So, only Administrator has the responsibility to add,modify,and delete users and passwords also following the specific user. No self-registry option.

You need to find what your **local network's IP** of that computer is. Then other people can access to your site by that IP.

5.1.2 User Database creation:

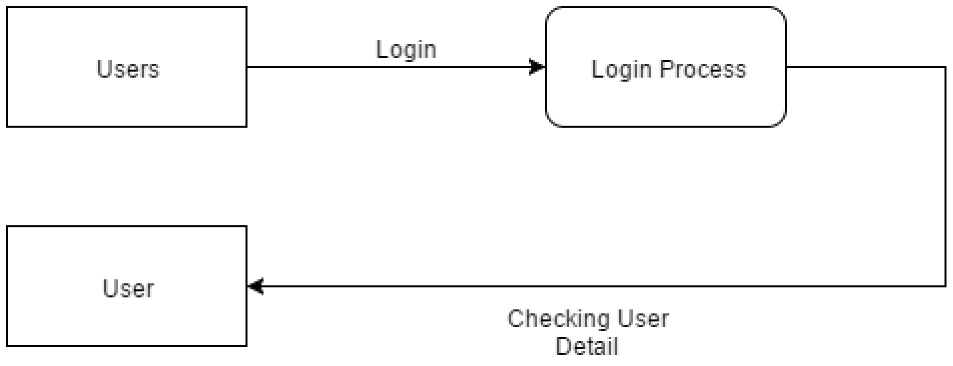
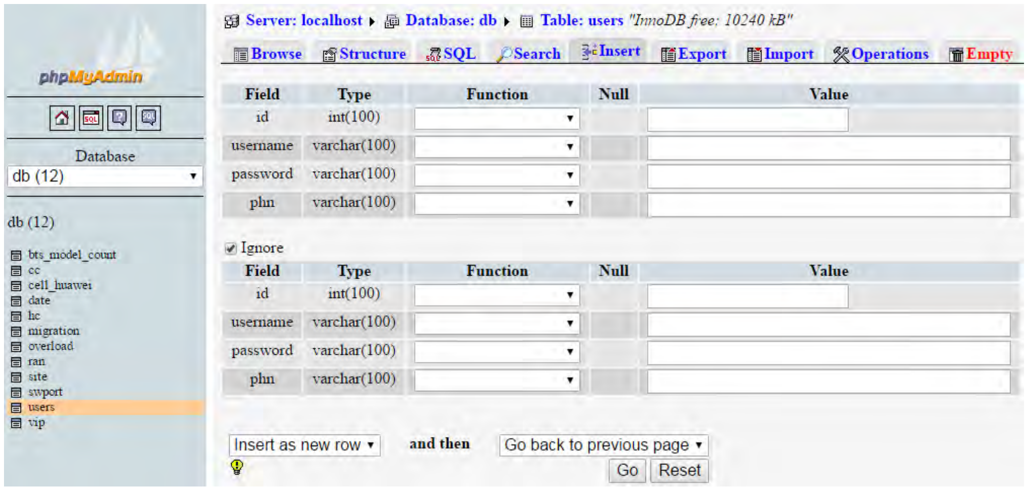


Figure: 5.1 Basic user creation process

The below is the prompt for manually creation of Users:



Server: localhost ▸ Database: db ▸ Table: users "InnoDB free: 10240 kB"

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Empty

Field	Type	Function	Null	Value
id	int(100)			
username	varchar(100)			
password	varchar(100)			
phn	varchar(100)			

☑ Ignore

Field	Type	Function	Null	Value
id	int(100)			
username	varchar(100)			
password	varchar(100)			
phn	varchar(100)			

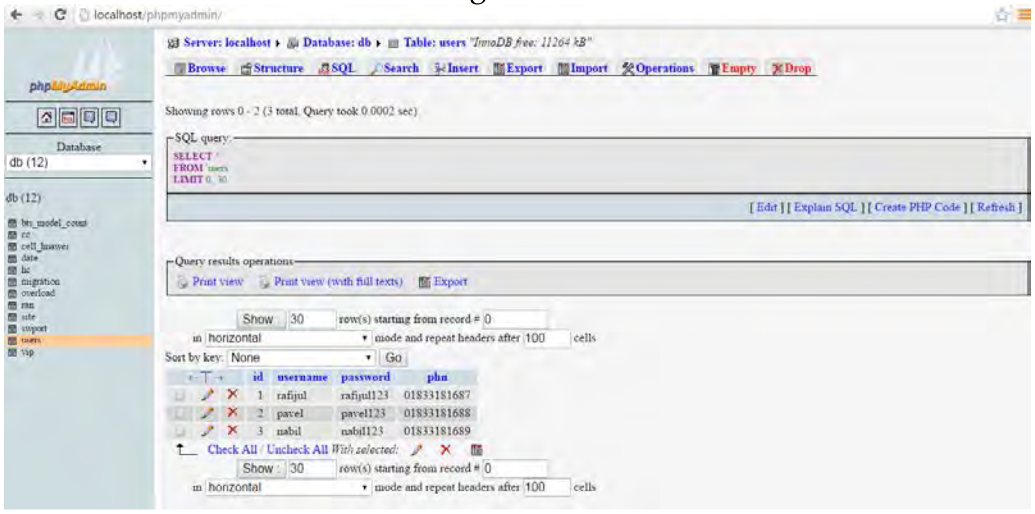
Insert as new row ▾ and then Go back to previous page ▾

Go Reset

Database: db (12)

db (12): bts_model_count, cc, cell_huawei, date, hc, migration, overload, ran, site, swport, users, vip

Figure: 5.2



localhost/phpmyadmin/

Server: localhost ▸ Database: db ▸ Table: users "InnoDB free: 11264 kB"

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Empty | Drop

Showing rows 0 - 2 (3 total, Query took 0.0002 sec)

SQL query:

```
SELECT *
FROM `users`
LIMIT 0, 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Query results operations: Print view | Print view (with full texts) | Export

Show : 30 row(s) starting from record # 0
in horizontal ▾ mode and repeat headers after 100 cells
Sort by key: None ▾ Go

id	username	password	phn
1	rafijul	rafijul123	01833181687
2	pavel	pavel123	01833181688
3	nabil	nabil123	01833181689

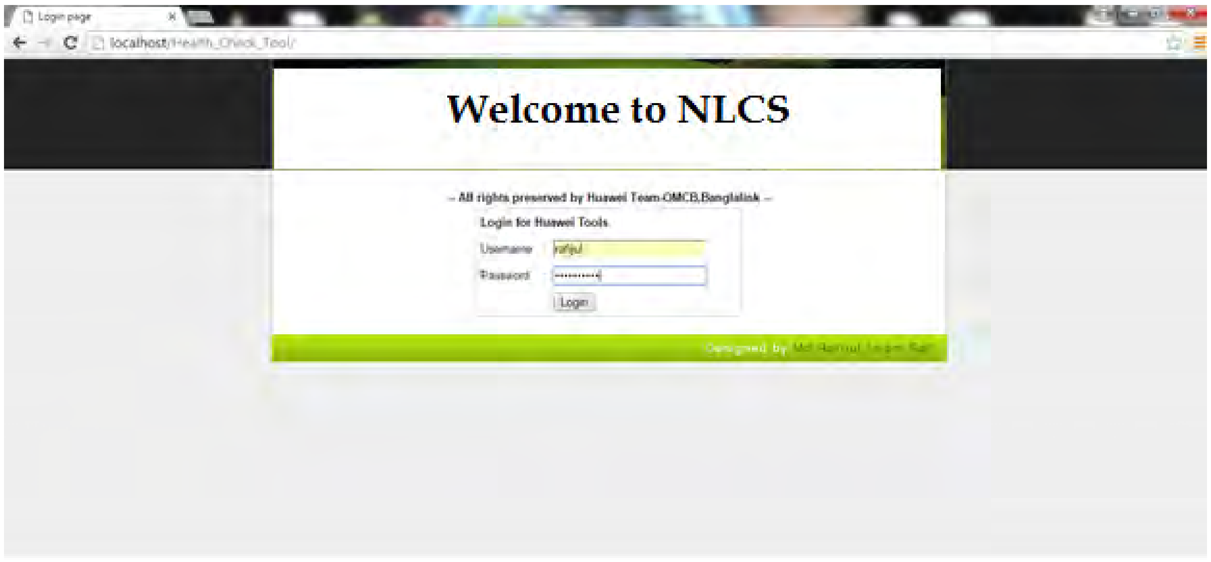
Check All / Uncheck All With selected:

Show : 30 row(s) starting from record # 0
in horizontal ▾ mode and repeat headers after 100 cells

Figure: 5.3

5.1.3 Tools Login Page

Tool has an initial prompt is look like



Welcome to NLCS

-- All rights preserved by Huawei Team-OMCB,Banglalink --

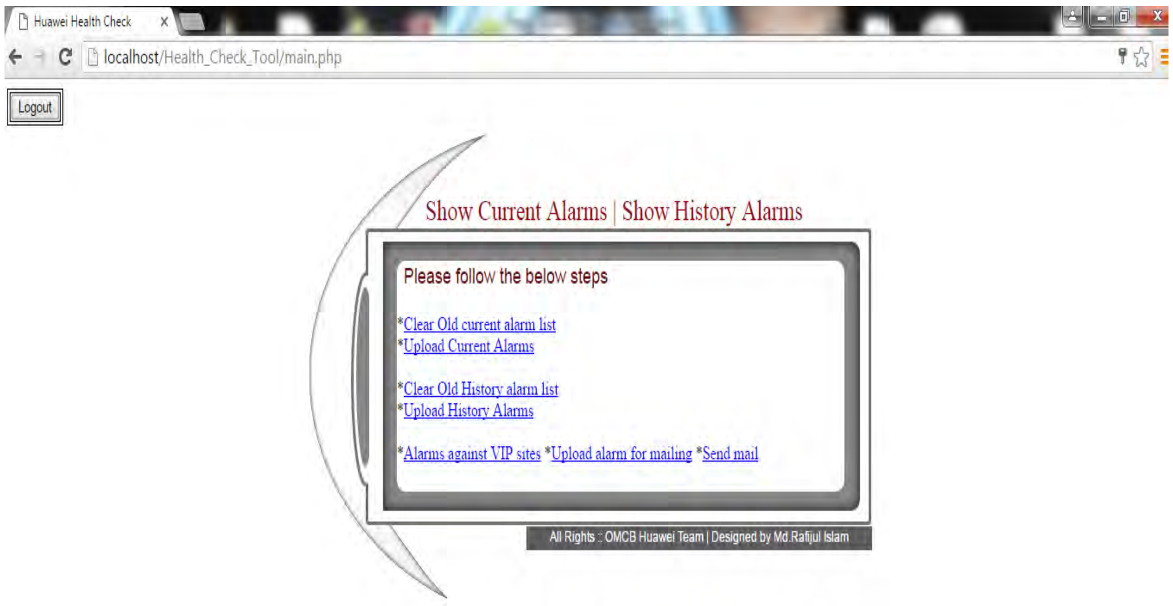
Login for Huawei Tools

Username: rafiul

Password: ••••••••••

Figure: 5.4

Logout option is also available after finishing the work as below:


Huawei Health Check
localhost/Health_Check_Tool/main.php
Logout
Show Current Alarms | Show History Alarms

Please follow the below steps

*Clear Old current alarm list
*Upload Current Alarms

*Clear Old History alarm list
*Upload History Alarms

*Alarms against VIP sites *Upload alarm for mailing *Send mail

All Rights : OMCB Huawei Team | Designed by Md.Rafijul Islam

Figure: 5.5

5.1.4 Clear Old alarm list

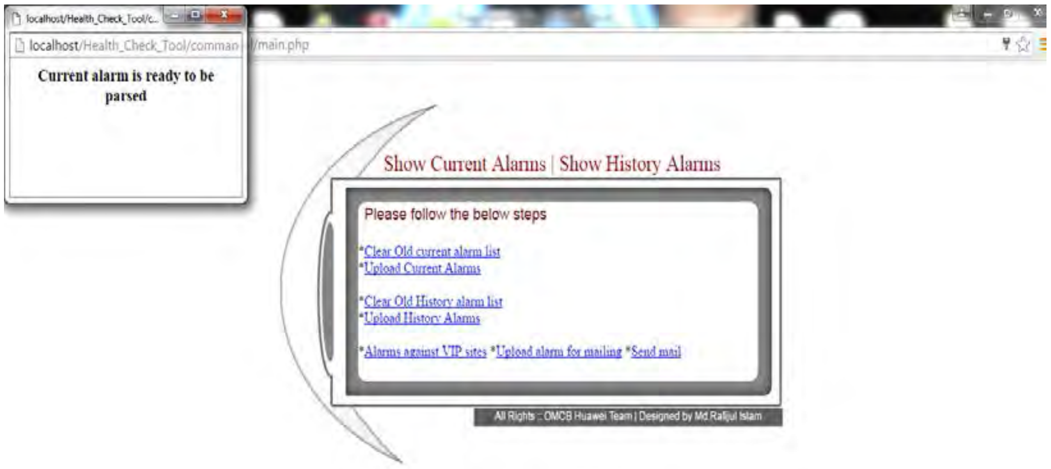
At first we need to clear the previous data from the table of database. Here truncate function is used for truncating data from table.

$sql1 = mysql_query("TRUNCATE TABLE cc");

Database is used here as "db"

Trim () function is used to remove the character from the strings.

Location of Current alarm is table cc in database db. So, when query goes to the database, it starts to truncate data following truncate function .After completion of removing data from table, it pops out a window which tells user that "Current alarm is ready to be parsed ". So, it is ready to parse the new data to the table.

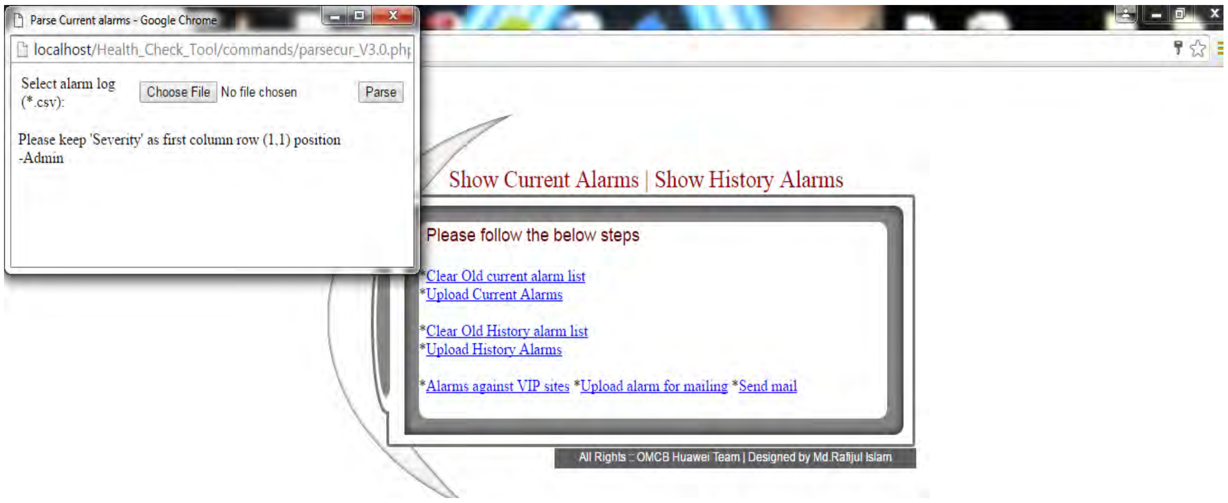

localhost/Health_Check_Tool/c...
localhost/Health_Check_Tool/comman
/main.php
Show Current Alarms | Show History Alarms
Please follow the below steps
*Clear Old current alarm list
*Upload Current Alarms
*Clear Old History alarm list
*Upload History Alarms
*Alarms against VIP sites *Upload alarm for mailing *Send mail
All Rights :: OMCB Huawei Team | Designed by Md Rafijul Islam

Current alarm is ready to be parsed

Figure: 5.6

5.1.5 Upload Current Alarm

For uploading alarms into the table of database, firstly we have to parse the alarms into this prompt. We have huge alarms generated from 3 U2000 servers. We have to parse all the alarms into this parse button in .csv format after collecting those alarms from U2000. We need to keep 'Severity' as first column row (1, 1) position for calling function in another script. For this a warning sentence will be shown as a reminder.


Parse Current alarms - Google Chrome
localhost/Health_Check_Tool/commands/parsecur_V3.0.php
Select alarm log (*.csv):
Choose File
No file chosen
Parse
Show Current Alarms | Show History Alarms
Please follow the below steps
*Clear Old current alarm list
*Upload Current Alarms
*Clear Old History alarm list
*Upload History Alarms
*Alarms against VIP sites *Upload alarm for mailing *Send mail
All Rights :: OMCB Huawei Team | Designed by Md.Rafijul Islam

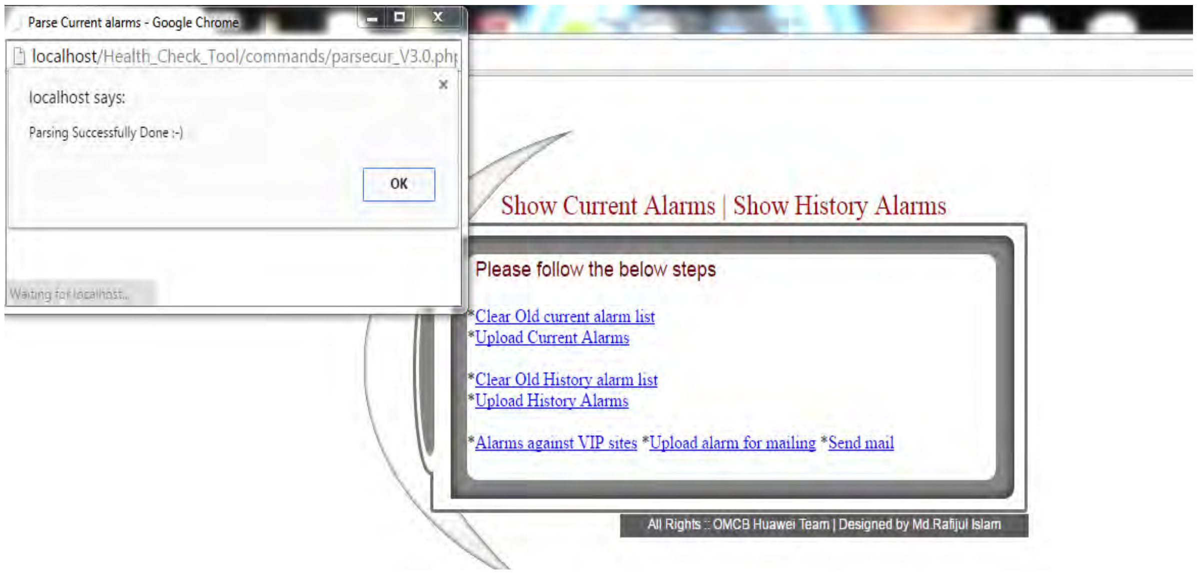
Please keep 'Severity' as first column row (1,1) position
-Admin

Figure: 5.7

After parsing successfully, “Parsing Successfully Done” will be pop out. Firstly data are transferred to a file named “fl” from there.

```
move_uploaded_file($_FILES["file"]["tmp_name"],
    "fl/" . $_FILES["file"]["name"]);
```

There some trimming occurred through which unnecessary data are being trimmed. After that it enters data into the database again by following some logic.


Parse Current alarms - Google Chrome

localhost/Health_Check_Tool/commands/parsecur_V3.0.ph

localhost says:

Parsing Successfully Done :-)

Waiting for localhost...

Show Current Alarms | Show History Alarms

Please follow the below steps

*Clear Old current alarm list
*Upload Current Alarms

*Clear Old History alarm list
*Upload History Alarms

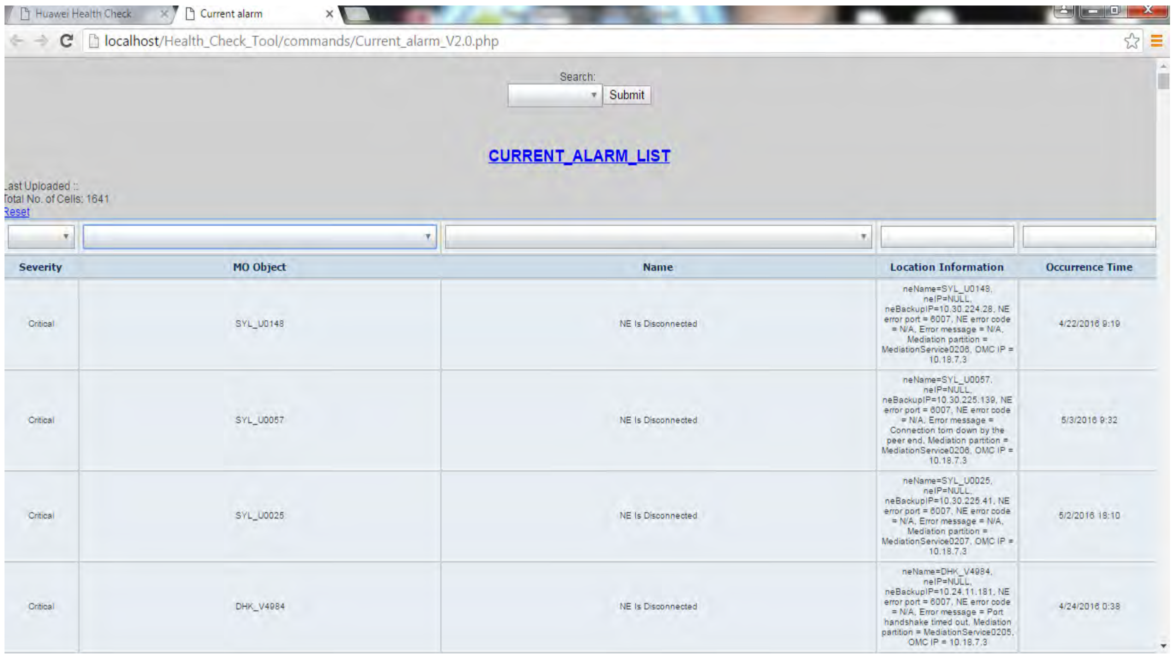
*Alarms against VIP sites *Upload alarm for mailing *Send mail

All Rights :: OMCB Huawei Team | Designed by Md.Rafijul Islam

Figure: 5.8

5.1.6 Show Correlated Alarm list

Show current Alarms button leads to Current alarm list page. Here we can get sorted out form of alarms. One can get required department’s current alarm by using drop down menu.



Severity	MO Object	Name	Location Information	Occurrence Time
Critical	SYL_U0148	NE Is Disconnected	neName=SYL_U0149, neIP=NULL, neBackupIP=10.30.224.28, NE error port = 6007, NE error code = N/A, Error message = N/A, Mediation partition = MediationService0206, OMC IP = 10.18.7.3	4/22/2016 9:19
Critical	SYL_U0057	NE Is Disconnected	neName=SYL_U0057, neIP=NULL, neBackupIP=10.30.225.139, NE error port = 6007, NE error code = N/A, Error message = Connection torn down by the peer end, Mediation partition = MediationService0206, OMC IP = 10.18.7.3	5/3/2016 9:32
Critical	SYL_U0025	NE Is Disconnected	neName=SYL_U0025, neIP=NULL, neBackupIP=10.30.225.41, NE error port = 6007, NE error code = N/A, Error message = N/A, Mediation partition = MediationService0207, OMC IP = 10.18.7.3	5/2/2016 18:10
Critical	DHK_V4984	NE Is Disconnected	neName=DHK_V4984, neIP=NULL, neBackupIP=10.24.11.181, NE error port = 6007, NE error code = N/A, Error message = Port handshake timed out, Mediation partition = MediationService0205, OMC IP = 10.18.7.3	4/24/2016 0:38

Figure: 5.9

Sorted out alarms according to the Department:

Here is drop down menu. There we can find the department's name –

- **BSC Alarms**
- **BTS Alarms**
- **RNC Alarms**
- **Node B alarms**
- **OSS Alarms**

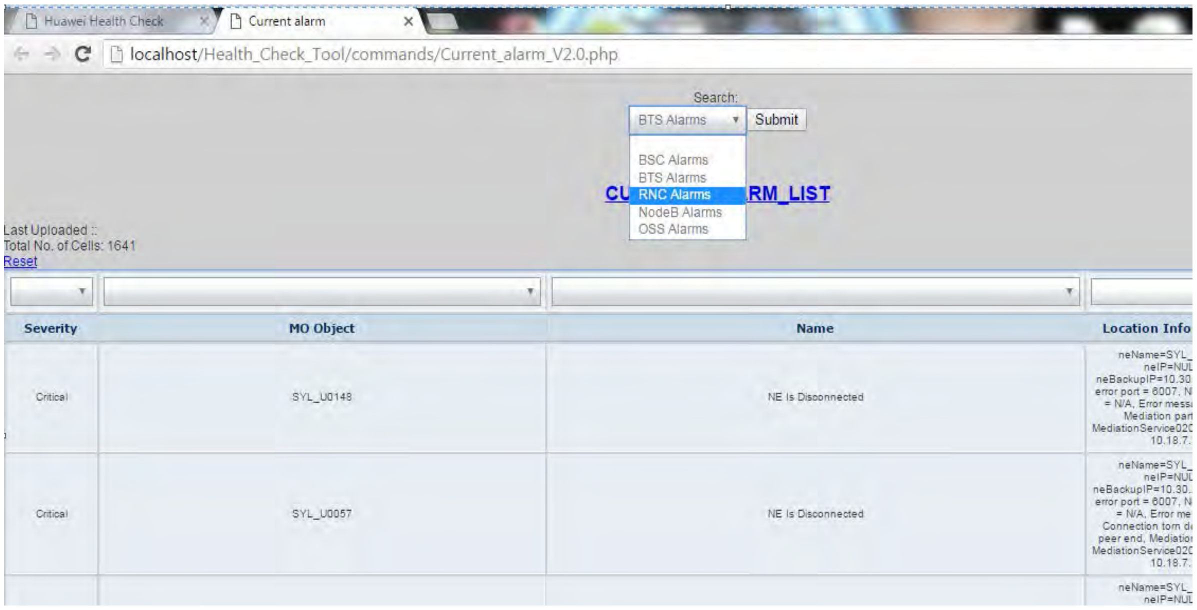
We can query the database to send back the alarms only for those departments. Then we can export those alarms in a excel file format.

One of the sorting logic is as below:

if ($search==2)

{$sql = mysql_query("SELECT * FROM cc where ((Alarm_ID BETWEEN 25600 AND 26529) OR (Alarm_ID BETWEEN 26532 AND 26540) OR (Alarm_ID BETWEEN 26542 AND 26750) OR (Alarm_ID BETWEEN 26754 AND 28052) OR (Alarm_ID BETWEEN 2114 AND 2622)

OR (Alarm_ID BETWEEN 3570 AND 4016) OR (Alarm_ID BETWEEN 10246 AND 13330) OR (Alarm_ID BETWEEN 4100 AND 9770) OR (Alarm_ID BETWEEN 21801 AND 21807))");



Last Uploaded ::
Total No. of Cells: 1641

Severity	MO Object	Name	Location Info
Critical	SYL_U0148	NE Is Disconnected	neName=SYL_ neIP=NUL neBackupIP=10.30 error port = 6007, N = N/A, Error mess Mediation par MediationService02C 10.18.7.
Critical	SYL_U0057	NE Is Disconnected	neName=SYL_ neIP=NUL neBackupIP=10.30. error port = 6007, N = N/A, Error me Connection torn d peer end, Mediatio MediationService02C 10.18.7.
			neName=SYL_ neIP=NUL

Figure: 5.10

We can export those alarms in a excel file format when we click CURRENT_ALARM_LIST .An user can get his specific department's alarm, if any one try to get the whole current alarm list, he can get it by pressing the same button.

Main logic behind the correlation will be based on the resolution table of database. Problem resolutions will be stored in a table along with a scoring system based on Engineers feedback. Existing alarmID will be cross-matched with the resolution of historical data.

5.1.7 Alarms against VIP sites

This is a button by pressing we will get some alarms of important VIP sites. Only VIP sites alarms will be chosen here.

5.1.8 Historical Alarms

To get the Historical alarms, same procedure has to be followed by user. This alarms are needed to analysis the previous day's Critical situations.

5.1.9 Uploading alarm for mail & Send mail

The below is a sample of mail that generates by this button. We need to parse the segregated data in "Uploading alarm for mail". Then after pressing send mail, it will go to that location where mail server connected and mail to proper department.

Chapter 6: Conclusion

In conclusion, the work and contribution of this paper is summarized below. A literature review was conducted that revealed a need for applying modern technologies in correlation technique. Current technology trends were examined to determine how this could be used to solve this problem. Requirements for a web application that helps to resolve this issue were elicited and a prototypical application was implemented. Furthermore a literature review was also conducted to identify what are the major software engineering challenges to develop web application according our case. Four major issues that were identified are Creating Universal User Interfaces, Enabling Software Reuse across DB Platforms, Event driven Applications, Balancing Agility and Uncertainty in Requirements.

6.1 Further Scope

While the potential of the application and its supportive structure for telecom engineers, there is a need for a review of the user interface of the application in order to decrease its complexity. Apart from a user interface re-design, there are also many capabilities and functionalities that could potentially increase the usefulness of the prototypical application. New elements such as possibility to correlate through internet and international solution bases. Another useful addition would be sharing possibilities with different platforms which could increase solution information base. Due to a lack of time, the features mentioned above could not be implemented however in future these areas of improvement should be explored. Also it was not possible to conduct an evaluation of the application by distributing it among a selected group of users. It is very much necessary that in future an evaluation is conducted to find out potential shortcomings of the application both from software and hardware perspectives.

References

[1] The Telecommunications Handbook. Ed. KornelTerplan, Patricia Morreale

[2] International telecommunication Union, record/T-REC-M.3010

[3] Book- Systems analysis and design /Alan Dennis, Barbara Haley Wixom, Roberta M. Roth.–5thed.p. cm.

[4] Boehm, B., "When Models Collide: Lessons from Software Systems Analysis," IT Professional, Vol. 1, No. 1, 1999, pp. 49-56.

[5] Stephens, R., "Measuring Enterprise Reuse in Large Corporate Environment," Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory, 2004, pp. 565-569.

[6] Book-DCN Introduction Dr. John Sum Institute of Electronic Commerce

[7] http://atlas.tk.informatik.tu-darmstadt.de/Publications/1999/web.pdf

[8] http://www.slideshare.net/secomps/telecommunicationppt?next_slideshow=1

[9] National Chung Hsing University/NetworkManagement

[10] http://www.tutorialspoint.com/gsm/gsm_specification.htm

[11] https://en.wikipedia.org/wiki/Operations_support_system

[12] https://en.wikipedia.org/wiki/Operations_support_system

[13] http://e.huawei.com/en/products/wireless/gsm-r/oss/m2000

[14] http://www.ossline.com/what-is-oss

[15] http://www.webml.org/webml/upload/ent5/2/BrambillaEtAl.pdf