

DISASTER MANAGEMENT SUPPORT SYSTEM



By

Ahsan Sadeque
Nofel Elahi
Shakeeb Ahmad

Submitted to the Faculty of Computer Software Engineering, National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering
JULY 2011

ABSTRACT

DISASTER MANAGEMENT SUPPORT SYSTEM

The main purpose of the project is to set up an IT infrastructure to respond to different disasters/emergencies. The IT infrastructure would constitute various organizations providing their services (resources) to perform various post management tasks.

The system is accessible through a Web interface. User authentication and access management is controlled by a User Management Service. A role-based access policy is used to provide access to different features of the system. Disaster Management Service is the key component of the system which consists of various subsystems including Workflow System. This system appoints tasks to different organizations and also sets priorities and dependencies between different tasks. A Capability Assessment Engine suggests resources of different organization based on disaster specific parameters. Resource Management Service has the responsibility to update status of resources. Communication Service allows different organizations and officials involved in a task to communicate with one another. Disaster Support Repository is a normalized relational database managing the records of the entire system.

The above tasks have been accomplished by establishing Web Services in Java language using JAX-WS API. JAX-WS makes use of annotations to simplify the development and deployment process of Web service and its clients that communicate using XML. JAX-WS uses SOAP as message protocol. Specification of SOAP describes the envelope composition, its encoding rules and the ways of expressing Web service request and response. Request and response are used as SOAP messages over HTTP. The interface consists of JSP pages, for dynamic data handling without reloading the page we have also employed JQUERY, AJAX, JSON. Java Servlet enables us to separate business logic from presentation. Methodologies have been used to prevent SQL Injection attacks.

A comprehensive testing methodology including Prototype Testing, unit testing, and integration testing is followed. Usability analysis is also carried out by following different usability heuristics.

CERTIFICATE OF CORRECTNESS AND APPROVAL

Certified that work contained in this thesis “Disaster Management Support System” carried out by Ahsan Sadeque, Shakeeb Ahmad and Nofel Elahi under the supervision of Asst. Prof. Dr. Awais Majeed for partial fulfillment of Degree of Bachelor of Software Engineering is correct and approved.

Approved by

(Supervisor Name)

_____ Department

MCS

Dated: _____

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, The Most Beneficent

To our teachers and parents, without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible.

ACKNOWLEDGEMENT

We are eternally grateful to Almighty Allah for bestowing us with the strength and resolve to undertake and complete the project.

We gratefully recognize the continuous supervision and motivation provided to us by our supervisor, Assistant Professor Dr Awais Majeed and Madam Nausheen for their continuous and valuable suggestions, guidance, and commitment towards provision for the undue support throughout this project.

We will also like to thank other faculty members of the department of Computer Software Engineering for their continuous support and help. It was their help and guidance which helped us complete project in due time.

We would also like to thank our class mates and last but not the least our family members for their support throughout the project in every way possible.

TABLE OF CONTENTS

Chapter 1 Introduction	1
1.1 Introduction	1
1.2 Background	1
1.3 Problem Statement	2
1.4 Goals and Objective	2
1.5 Deliverables	3
1.6 Document Organization	3
1.7 Summary	3
Chapter 2 Literature Review	3
2.1 Introduction	4
2.2 Disaster Management in Pakistan	4
2.3 IT based tools for Disaster Management.....	5
2.3.1 SAHANA FOSS Disaster Management System.....	5
2.3.2 Disaster Management Information System.....	5
2.3.3 Limitations	6
2.4 Enabling Technologies	6
2.4.1 Service Oriented Architecture	6
2.4.2 Web Services.....	7
2.4.3 Application Programmer Interface (API's).....	8
2.5 Summary.....	9
Chapter 3 System Requirements	9
3.1 Introduction	9
3.2 Project Scope	10
3.3 Product Features	10
3.3.1 Competency/Resource repository of organizations.....	10
3.3.2 Competency Analysis Module	11
3.3.3 Team formation and Task management (Workflow Engine).....	11
3.3.4 Peer Evaluation (Trust/Reputation Engine)	11
3.4 Assumptions and Dependencies.....	11

3.4.1	Basic Assumptions.....	12
3.4.2	Operating System	12
3.4.3	Web Server	12
3.5	System Features	12
3.5.1	Team Selection	12
3.5.2	Workflow System	12
3.5.3	Appoint Tasks	13
3.5.4	Notification/Communication Channel	13
3.5.5	Status Check.....	13
3.5.6	Record Maintenance.....	13
3.5.7	Login/Access Rights	13
3.5.8	Establishing Services.....	14
3.6	External Requirements.....	14
3.6.1	User Interface	14
3.6.2	Hardware Requirements.....	14
3.6.3	Software Requirements	14
3.7	Other Non-Functional Requirements	14
3.8	Software Quality Attributes	15
3.8.1	Runtime System Qualities	15
3.8.2	Non-Runtime System Qualities	15
3.9	Other Requirements.....	16
3.9	Summary.....	16
	Chapter 4 System Design.....	17
4.1	Introduction	17
4.2	System Overview	17
4.3	Assumptions and Dependencies.....	18
4.4	System Requirements	18
4.5	General constraints.....	18
4.6	Architectural Strategies	18
4.7	System Architecture.....	20
4.7.1	User Support Service.....	22
4.7.2	Disaster Manager Service	22

4.7.3	Disaster Alert Service	22
4.7.4	Workflow Manager	22
4.7.5	Capability Assessment Engine.....	23
4.7.6	Resource Manager	23
4.7.7	Communication Manager	23
4.7.8	Grading Service	23
4.7.9	External Agency System	23
4.7.10	Support Repository	23
4.8	Use Case Diagram.....	24
4.8.1	Basic Flow of DMSS	24
4.8.2	Post Conditions.....	25
4.8.3	Alternate Scenarios	25
4.9	Class Diagram.....	25
4.10	Entity Relationship Diagram	29
4.11	Detailed Design.....	31
4.11.1	Activity Diagram: Task creation.....	31
4.11.2	Sequence Diagram: Adding resource to task.....	32
4.11.3	State Machine Diagram: Organization.....	33
4.12	Web Modeling	34
4.12.1	Presentation Model.....	34
4.12.2	Hypertext Structure Model.....	35
4.13	Summary:.....	36
	Chapter 5 Implementation.....	37
5.1	Introduction	37
5.2	Tools and Technologies	37
5.2.1	Web Services	39
5.2.2	WSDL	39
5.2.3	XML BINDING.....	40
5.2.4	SOAP MESSAGE	41

5.2.5	JAX-WS.....	43
5.2.6	JAX-B.....	44
5.2.7	SAAJ.....	44
5.2.8	jQuery.....	45
5.2.9	JSON.....	45
5.2.10	AJAX.....	45
5.3	User Interface.....	46
5.4	User Management Module.....	47
5.5	Disaster Management Module.....	48
5.6	Disaster Support Repository.....	50
5.7	Grading Service.....	51
5.8	Disaster Alert Service.....	51
5.9	Summary.....	51
	Chapter 6 Testing.....	52
6.1	Introduction.....	52
6.2	Testing Levels.....	52
6.2.1	Unit Testing.....	52
6.2.2	Integration Testing.....	53
6.2.3	System Testing.....	53
6.3	Box Approach.....	53
6.4.1	Test Case 1.....	54
6.4.2	Test Case 2.....	54
6.4.3	Test Case 3.....	55
6.4.4	Test Case 4.....	56
6.4.5	Test Case 5.....	56
6.5	Summary.....	56
	Chapter 7 Results and Analysis.....	56
7.1	Introduction.....	57
7.2	Results.....	57
7.3	Analysis.....	57
7.4	Summary.....	58
	Chapter 8 Conclusion and Future Work.....	58

8.1	Introduction	58
8.2	Concept	59
8.3	Future Work	59
8.4	Summary	60
	Appendix A.....	60
	Appendix B.....	70
	References	80

LIST OF FIGURES

Figure	Figure Name	Page Number
4.1	Client Server Architecture	19
4.2	Component Diagram	20
4.3	Use Case Diagram	24
4.4	Class Diagram	27
4.5	ER Diagram	30
4.6	Activity Diagram	32
4.7	Sequence Diagram	33
4.8	State Machine Diagram	34
4.9	Navigation Model	36
4.10	Hypertext Model	37
5.1	Overall System Diagram	38
5.2	XML Binding Disaster Support Service	40
5.3	Soap Message	43
5.4	JAX WS API Architecture	44
5.5	WSDL Disaster Support Service	47
5.6	WSDL Disaster Management Service	49
5.7	Prepared Statement	51
6.1	Session Cookie	56

LIST OF TABLES

Table	Table Name	Page No
5-1	Java Servlets	46

Chapter 1

Introduction to Disaster Management Support System

1.1 Introduction

Natural and manmade disasters bring destruction, chaos and misery to human life and society. Under such circumstance, need for effective and timely communication, collaboration of different organizations and people becomes very much important to carry our various activities of rescue, relief and rehabilitation. Disaster Management Support System is a resource management and collaboration platform to support such activities specifically rescue and relief operations after a disaster. Disaster Management Support System has been developed using open standards like XML, SOAP and Web services for interoperability and cross platform support to facilitate organizations in integrating their existing IT solutions.

1.2 Background

Disasters are mostly naturally occurring events that have a negative effect on human life. Disasters can be of two types, due to geological activities on surface of earth or human errors. Geological activities are occurring at all times its only when it causes a significant damage to human life it's considered a disaster. Disasters that occur in low vulnerability areas such as uninhabited areas are usually not called disasters. Natural disasters can include floods, tsunamis, tornadoes, volcanic eruptions and so on. While disasters due to human error can be like fire, oil spills, nuclear waste and so forth. In the last decade hundreds of thousands of people have died in these disasters and countless other have suffered their impacts and are still suffering. Tsunami in 2004 and 2011, earthquake in Pakistan in 2005 and floods in 2010 have made a huge impact on human life. Many people died while still many became homeless. Disaster Management is vast and critical subject in this scenario because if a disaster occurs and post disaster activities are not managed properly it can not only expand the impact of disaster but cause further disasters. However the impact of

disasters can be minimized by using disaster mitigation strategies. These include implementation of disaster early warning systems, preparing development plans to provide resilience for disasters, mobilizing resources and expediting rehabilitation and post disaster activities.

Unfortunately after suffering two of the biggest disasters in last decade Pakistan still don't have an IT infrastructure for management of post disaster activities. To manage post disaster activities Pakistan needs an effective IT solution and that is the reason of development of DMSS. Since a large number of organizations are active in Pakistan, it is necessary that the said solution should be easy to integrate and communicate with existing and upcoming solution. That is why DMSS has been built using web services.

1.3 Problem Statement

Natural calamities cannot be avoided. However efforts can be made to minimize effects of disasters. Mismanagement can lead to more damage especially more human casualties. In the recent years a number of disasters have occurred in the world. Pakistan itself has faced two of the biggest disasters in recent history in shape of earthquake in 2005 and floods in 2010 which has triggered the need of developing a system to gain control of disaster quickly and to effectively plan rescue and relief operations and execute them.

DMSS addresses this problem by efficiently managing post disaster activities to provide timely and effective help to the effected people.

1.4 Goals and Objective

DMSS has been developed to facilitate in post disaster activities. The objectives of DMSS is locating and allocating resources and helping organization to collaborate with each other in post disaster activities of rescue and relief. The mandatory functions of DMSS are locating and allocating resource, assigning tasks to different organizations and developing a workflow for different rescue and relief activities. The goal of the project is to create a new project once a disaster has occurred,

develop workflow(s) for different post disaster activities, assign activities and resources to different organizations, monitor their performance and finally grade how efficiently they performed a task. All of this happens in real time once a disaster has occurred. Another goal of DMSS is easy integration with I.T infrastructures of different organizations.

1.5 Deliverables

The deliverables for this project is software system/application that will provide functionalities to help in post disaster activities.

1.6 Document Organization

This document provides basic knowledge about DMSS. Initially the description of project has been given, the next chapter discusses related software products and their functionalities and how DMSS is different from them. Requirement specifications have been covered in chapter 3, Design specifications are discussed in chapter 4. Implementation is described in chapter 5 and further chapters explains analysis of the software , testing techniques employed and suggestion about further work in enhancing the functionalities and capabilities of the system.

1.7 Summary

DMSS is a resource management and collaboration tool to assist in post disaster rescue and relief operations and comes under the umbrella of web engineering. This chapter is an introduction to DMSS, goals and objectives which have been defined in order to develop this software. A brief introduction to document contents is also included.

Chapter 2

Literature Review

2.1 Introduction

Main objective of this project is to help and support the Pakistani government in post-disaster activities. For this purpose, the disaster National Disaster Management Authority (NDMA) has developed a comprehensive disaster management framework. Salient features of this framework and working of NDMA is discussed in this chapter. This framework is the foundation of the proposed Disaster Management Support System. Different existing systems to support the disaster management are also discussed in this chapter. The limitations of these systems are also discussed.

2.2 Disaster Management in Pakistan

Pakistan's inability to deal with large disasters was highlighted during the earthquake of 2005. It was after observing this deficiency National Disaster Management Authority was formed. The framework for dealing with disasters laid down by NDMA lays down SOPs for dealing with different disasters. It also calls for establishment of a forum to increase collaboration between armed forces and other governmental and non-governmental organizations. The said framework also calls for strengthening PDMA's and DDMA's as they are currently very weak. Under this framework NDMA has also established Emergency Operations Centers at national, provincial and district level.

Emergency Operations Centers at regional and district level are responsible for continuously monitoring different risks, hazards and vulnerable conditions. They also coordinate complete spectrum of disasters in their regions and are responsible for spreading awareness and education on disaster reduction and response.

NDMA has laid down the foundations for efficient post and pre disaster activities in its framework. However, the support infrastructure has not yet been provided by the concerned agencies. The core issues under such conditions are of communications, information sharing, access to right and correct information and collaboration between different agencies. An IT infrastructure along with the relevant software systems can greatly help

these operations. However, in Pakistan, we still don't have the central repository that contains information about the capabilities and resources available that can be used under such conditions. Therefore, the proposed DMSS can help the NDMA and the concerned organizations in this regards.

In the next sections we will discuss the available disaster management systems that have been used by different relief and welfare organizations worldwide for the post-disaster relief and rescue operations.

2.3 IT based tools for Disaster Management

The following section provides a brief description of tools already developed for disaster management and the last section highlights the limitation of these systems.

2.3.1 SAHANA FOSS Disaster Management System

SAHANA is a free and open source system built for disaster management after 2004 Sri Lanka tsunami. The system was initially deployed by Sri Lanka's CNO (Center of National Operations) . The project has since grown with deployment in Pakistan Earth Quake , Philippine Mudslide (2006) and Indonesian Earthquake (2006) as examples. The system was initially developed for missing person registry while as an open source project rest of the modules have been developed by different communities all over the world. While the issue of scalability has been recently identified and people are working on exposing SAHANA FOSS functions as web services.

Sahana provides the functionalities of missing person registry, shelter registry, request management system, inventory management, and volunteer coordination.

2.3.2 Disaster Management Information System

This is a web based tool developed by International Federation of Red Cross and Red Crescent societies. The biggest disadvantage is that this tool can only be used by members of Red Cross and Red Crescent societies. The tool basically provides information to federation about disaster trends, resource both internal and external and different tools for information management [3].

2.3.3 Limitations

SAHANA FOSS's limitation is that it was developed as a database for missing person so the focus from the start wasn't on disaster management activities. Secondly SAHANA didn't support scalability as such because it is not based on web services . Though now a project has been started to shift SAHANA FOSS to web services.

Disaster Management Information System can be only used by volunteers of International Red Cross and Red Crescent Federation.

2.4 Enabling Technologies

Accessibility, interoperability and open standards can help the organizations to develop systems that can be integrated together. In the case of disaster management, the main monitoring agency has to work with different organizations including health services, armed forces, civil organizations, met office and NGOs. These organizations may have their own systems which include the information related to their available resources, held stock levels of goods, competencies and team/project management system. Thus, the integration of these systems at National level is a major challenge. Moreover, accessibility to a centralized system is also a major concern. Therefore, appropriate tools and technologies are required to develop an integrated solution that can help in various post-disaster activities.

The following sections give a brief description of various tools and technologies that can be helpful in developing an open, interoperable and extendible disaster management support system.

2.4.1 Service Oriented Architecture

DMSS has been built on service oriented architecture, publishing functionalities as web services.

Service oriented architecture is in essence design principles which are used during computer system development and integration. A system developed on SOA will have different functionalities offered as interoperable services which can be used within multiple, separate systems from several business domains [5].

SOA also generally provides a way for consumers of services, such as web-based applications, to be aware of available SOA-based services. For example, several disparate departments within a company may develop and deploy SOA services in different implementation languages; their respective clients will benefit from a well understood, well defined interface to access them. XML is the most common method to access services. [5].

A service has three fundamental attributes: a description of service, this is the interface of the service; a method to access the service by invoking its interface and an implementation of the service.

2.4.2 Web Services

Web services provide a mean of interoperating between different software applications that might or might not run on the same platform or web service.[6] To access a web service a client has to subscribe to a web service first and once authorized only then it can access its functionalities.

Web services can be implemented and used in three ways.

2.4.2.1 RPC Web Services

RPC (acronym for Remote Procedure Call) based web services are basically accessed synchronously accessed by RPC through a specific port and protocol. The simplest unit in a RPC based web service is an operation. The web services are directly mapped to these operations written in a specific programming language. This shows that RPC web services are not loosely couple and even though the very first web services tools were based on RPC and vendors are trying hard that Web Services Interoperability Industry Consortium disallow RPC Web Services.

2.4.2.2 SOA Web Services

Service Oriented Architecture means that architecture based on web services. This strategy means that web services can be used to implement architecture, unlike RPC web services where an operation is the building block, in SOA a message is the basic unit.

In SOA the biggest advantage is the use of protocols that work over HTTP for example SOAP message. These web services are also known as message oriented web services because a message is the method of communication in a distributed environment. SOA web services are focused on WSDL contract and how client and server communicate instead of implementation details like in case of RPC hence more and more vendors and software developers are shifting towards SOA web services for web application development.

2.4.2.3 RESTful Web Services

REST stands for representational state. REST web services combine principles of REST architecture and HTTP protocol , it uses standard HTTP operations for example GET, POST , DELETE and so on. The interaction is focused on stateful resources.

An architecture based on REST can use WSDL to describe SOAP messaging over HTTP, can be implemented as an abstraction purely on top of SOAP (e.g., WS-Transfer), or can be created without using SOAP at all. [16]

WSDL version 2.0 offers support for binding to all the HTTP request methods so it enables a better implementation of RESTful Web services. However, support for this specification is still poor in software development kits, which often offer tools only for WSDL 1.1 [16]

2.4.3 Application Programmer Interface (API's)

Different programming languages provide APIs to support the development of Web services. Java provides the following APIs to develop Web Services.

2.4.3.1 JAX-WS

JAX-WS stands for Java API for XML based Web Services. JAX-WS API is part of Java EE platform and it uses annotations for simplification in development and deployment of web services.

2.4.3.2 AXIS 2

Apache Axis2 is a core engine for Web services. Though developed specifically for web services it can also function as a standalone server.

Apache Axis2 not only supports SOAP 1.1 and SOAP 1.2, but it also has integrated support for the widely popular REST style of Web services.

2.4.3.3 JAX-RPC

JAX-RPC stands for Java API for XML based RPC allow an application to invoke Java based web services. It is now replaced by JAX-WS which support SOAP message format and W3C (World Wide Web Consortium) standards. JAX-RPC was the most used style of implementation of web services but now it's an obsolete practice.

2.5 Summary

This chapter introduces us to existing system for disaster management; their features and limitations. The section of enabling technologies provides an introduction to various technologies that can be used to develop DMSS so that overcomes limitations of existing systems. The comparison of different technologies also helps in choosing the technology that is not obsolete and in future would be helpful in expanding DMSS.

Chapter 3

System Requirements

3.1 Introduction

System requirements for the proposed Disaster Management Support System (DMSS) are gathered based on the framework proposed by the National Disaster Management System (DMSS) and the general project management principles. Moreover, these requirements are specified

based on the principles of Computer Supported Collaborative Work (CSCW), collaborative networks and Virtual Organizations. The proposed system helps in post disaster activities of rescue and relief. When a disaster occurs the system helps in developing workflows for different post disaster activities and assigns tasks based on the workflow to different teams thus enabling teams to collaborate and share resources to perform different rescue and relief tasks effectively and efficiently. This chapter provides specifications that DMSS has to fulfill so it can perform in the best possible manner.

3.2 Project Scope

The main objective of this project is to develop a Web based decision support system to support various activities associated with rescue and relief processes during a disaster/emergency situation. This system will particularly help disaster management authorities and concerned agencies to collaborate and cooperate with each other. Moreover, it will also help authorities to efficiently locate and allocate required resources to a particular rescue or relief operation. The scope of the project is restricted to rescue and relief operations only.

3.3 Product Features

DMSS has specific features which have been designed by focusing on how the rescue and relief operations take place. These features are the main functionalities the system will provide and form the basis for the system design and implementation. The required features are elaborated in more detail in the following sections.

3.3.1 Competency/Resource repository of organizations

DMSS should allow the maintenance a complete database of resources/competencies of different organizations. For this purpose DMSS should allow the organization registration process. Moreover, the system should support the features that allow the automatic updating of the central repositories whenever a particular organization updates the status of its resources in their own system. For this purpose, Web services can be

used to integrate the existing IT solutions of the registered organizations and the central repository maintained by DMSS.

3.3.2 Competency Analysis Module

DMSS should support the competency analysis of the organizations. This feature will be used whenever a new project is created, and a set of required competencies or required resources need to be generated. Based on the tasks involved in a particular operation, the system should help in identifying the concerned organizations with relevant skills/competencies as well as the available resources.

3.3.3 Team formation and Task management (Workflow Engine)

DMSS should help in developing a workflow to support the tasks associated with a particular type of disaster. Initially, the system should present a workflow based on some best practices in the form of a template. This workflow should be customizable by the Manager Operations (MO). Each task presented in the workflow should then be allocated to the most suitable organization. This information will be retrieved by the Competency Analysis Module. MO can give some special instruction and request for the required resources from the chosen organizations. The system should provide means for communication as well as progress tracking for the tasks that will be performed in a particular project (operation).

3.3.4 Peer Evaluation (Trust/Reputation Engine)

DMSS should support the evaluation of various participating organizations for future reference and the overall performance measurement of a particular operation. Time of completion, number of resources available/provided can be used as criteria for the evaluation. MO or the member organizations in a particular task can evaluate their peers.

3.4 Assumptions and Dependencies

Some basic assumptions have been made in the development of DMSS. This section describes assumptions and dependencies based on which DMSS has been developed.

3.4.1 Basic Assumptions

The system is robust and available 24/7 for the user. Because a disaster can happen anytime the system should be always available. In case of disaster the number of users using the system can increase drastically so software should use computer resources optimally. Also it should be able to serve a large number of requests. The user of the system should have basic knowledge of web applications.

3.4.2 Operating System

DMSS has to be built using Java and JAX-WS (Java API for web services). So the system is operating system independent and existing IT infrastructures can integrate with it because it's based on SOA.

3.4.3 Web Server

DMSS should be able to run on a J2EE compliant Web/Application Server. However, GlassFish Server is recommended for this purpose.

3.5 System Features

System features based on the functional requirements extracted from NDMA framework and Virtual Organization (VO) concepts are as follows:

3.5.1 Team Selection

This feature would allow user to select a team to undertake a task of rescue or relief operation once a disaster has struck. The user would create a new project once a disaster has occurred and for a given task would select a team based on available resources, abilities and past performance. The user will enter the nature and location of disaster based on which system will display a workflow or plan to combat disaster. The user would then assign the given list of tasks to appropriate teams.

3.5.2 Workflow System

After a disaster has struck, the user would create a new project giving parameters about the type of disaster. The system would then generate a workflow or a plan of rescue and relief activities. The system would also

set priority levels of different activities listed in the plan as well as the interdependency between tasks if required.

3.5.3 Appoint Tasks

This feature would allow appointing tasks to different teams or team leads. Once a workflow is generated based on disaster type, the different tasks enlisted in the workflow are appointed to teams based on their area of expertise.

3.5.4 Notification/Communication Channel

DMSS would have an effective communication channel so different organizations can collaborate with each other as well as update central command on the status of work they are performing/have performed. Also it will help in effective resource utilization on different tasks and projects.

3.5.5 Status Check

Each organization that has subscribed to DMSS will be able to publish its status on DMSS, so that a track of current activities can be kept. This feature would also help in planning and executing future tasks.

3.5.6 Record Maintenance

This feature would allow user to maintain records of organizations registered with the system. DMSS would have complete record of resources available and tasks and services performed by different teams during rescue and relief operations. DMSS will have its own record separate from organizations. The records will also be useful in future incidents as well as in studies of disaster management. Different users will have different access level to database repository. The database would be designed in such a way to protect data integrity and confidentiality.

3.5.7 Login/Access Rights

DMSS would allow users to login based on their hierarchy in the organization. Different users will have different features visible to them based on his role in the organization.

3.5.8 Establishing Services

This is a high priority feature and DMSS would export its features using web services. This will help in cross platform support and allow existing IT infrastructures to register with DMSS.

3.6 External Requirements

DMSS is a web application so it has some specific interface requirements elaborated in this section.

3.6.1 User Interface

DMSS is a web application and will perform actions based on user input. This requires that the interface of DMSS should have an easy learning curve for the user. Most of the important features should be visible to the user and no functionality should be hidden. It should be easy for user to perform basic actions and data available to the user should be displayed in a meaningful way.

3.6.2 Hardware Requirements

The hardware requirements for DMSS should be high enough so that the system can perform optimally without any compromise in performance under different levels of loads and stress.

3.6.3 Software Requirements

DMSS is operating system independent though the DMSS server on open source Glass Fish server. So Glass Fish web server needs to be installed on required machine. The database is developed using MySQL 5.0. Though the CREATE database script can be used to create repository on a different RDBMS.

3.7 Other Non-Functional Requirements

Certain other functionalities are required based on performance and response of DMSS. DMSS has to be efficient in its response and operation. The product domain requires that the software is optimized in terms of performance. The data flow should happen in the most efficient

way. The system performance shouldn't be affected when a new disaster happens and a new project is created.

3.8 Software Quality Attributes

Quality attributes of DMSS are described in this section. By following these attributes quality of DMSS has to be improved.

3.8.1 Runtime System Qualities

At run time DMSS has to provide its users functionalities so that they can perform post disaster activities of rescue and relief efficiently and effectively. Some of the runtime qualities that should be considered in development of DMSS are described here.

3.8.1.1 Functionality

DMSS must provide functions to create project and track project status. It should also provide functionalities to develop a work flow. DMSS must provide functions of task and resource appointment, status of tasks and resources.

3.8.1.2 Performance

DMSS performance shouldn't be affected in case a new project is created and different organizations login at the same time.

3.8.1.3 Availability

DMSS should be available 24x7 since a disaster can happen at any time.

3.8.1.4 Usability

Usability is an important criterion in development of DMSS. The system should present all functionalities in such a way that nothing is missed by user. DMSS will also provide a lot of data inputs and outputs to the end user so the interface present data in a proper standard format.

3.8.2 Non-Runtime System Qualities

These are qualities of DMSS which are required to make this software useful for further enhancements and future development as well as extending system to different environments.

3.8.2.1 Modifiability

DMSS must support modifiability so any further improvements or features are easy to incorporate

3.8.2.2 Portability

DMSS should be able to run in different computer environments. The DMSS server should be platform independent and support interoperability.

3.8.2.3 Reusability

The different functionalities DMSS is providing should also be available as a stand-alone project. So if another system is being developed which needs functionalities of DMSS, the system (DMSS) should be easy to understand to reuse it.

3.8.2.4 Integrate-ability

Different components of DMSS should work together in correct manner so that DMSS can be used in the most useful manner. Also different IT infrastructures should be able to integrate with DMSS.

3.8.2.5 Testability

Different quality tests should be performed so that DMSS is free from faults and perform according to requirements specified in this chapter.

3.9 Other Requirements

DMSS should be robust enough so that it can perform in the best manner when the system is under different loads and stress. If during a disaster the system fails, a backup server must be present. The database backup timings should also be defined. System should also validate any data user enters. DMSS database repository should also maintain data integrity and confidentiality.

3.9 Summary

This chapter describes the requirements of the system as described by Project Supervisor. It includes interface, functional and nonfunctional

requirements along with the main features system would provide to the end user. These requirements have been set after checking the feasibility of the system. These requirements have been considered as the fundamental principles for testing and standardization of the product.

Chapter 4

System Design

4.1 Introduction

This chapter describes the design specifications of DMSS. The design specifications have been developed using requirements described in chapter 3. This chapter provides details about system structure and architecture as well as database architecture.

4.2 System Overview

DMSS is a software system to facilitate in post disaster activities of rescue and relief. DMSS provides certain functionalities to help organizations carry out rescue and relief activities efficiently and effectively. Also it helps to locate and allocate resources effectively during these activities and develop an effective work plan to carry out these activities.

4.3 Assumptions and Dependencies

Basic assumption for development of DMSS is that system should be available 24x7 since a disaster can happen at any time. The server should be able to handle a large number of requests especially when a disaster occurs.

4.4 System Requirements

The software is platform independent since programming language used to build system is Java. But since it's a web application browser version greater than Internet Explorer 6 or Mozilla Firefox 3 or equivalent are recommended.

4.5 General constraints

DMSS has to provide functionalities described in chapter 3 but in order to enhance software usability few constraints are applied which are described below.

The server machine must use J2EE based application server preferably glass fish server. The web application has been designed for browsers supporting CSS 3.0. The user must have basic knowledge of Web applications and database repository is developed on MySQL 5.0 RDBMS.

4.6 Architectural Strategies

DMSS is a distributed application and is based on client server architecture. Figure 4-1 shows an overall picture of DMSS server, database server and different clients. The clients will interact with web services through HTTP's defined protocols. DMSS has two servers one for the core system which provides services to client and the second is the database repository which the core system communicates with using JDBC driver.

Comment [AM1]: Mention about Client-Server architecture. Then mention that Service Oriented Architecture will be used and various functionalities will be exposed as Web Services.

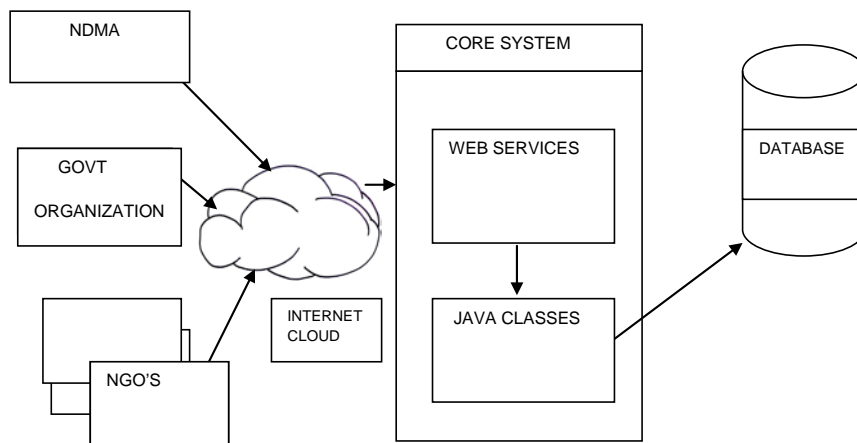


Figure 4-1 Client Server Architecture

DMSS has also been built using service oriented architecture, publishing functionalities as web services. Service oriented architecture is in essence

design principles which are used during computer system development and integration. A system developed on SOA will have different functionalities offered as interoperable services which can be used within multiple, separate systems from several business domains [5].

The major modules of DMSS are designed as Web Services providing an interface to the client to subscribe and then invoke the necessary service and use the functionalities it provides. SOA being platform independent and interoperable allows the clients to be language independent.

4.7 System Architecture

To manage a disaster through DMSS, a new project will be created. There are certain details the system needs as an input to create a new project and then develop a work flow for different rescue and relief activities. DMSS functionality is exported to the interface by web services. These web services use java packages to access java classes or classes. Which further access database to insert or retrieve data.

The component diagram of DMSS is shown in figure 4-2.

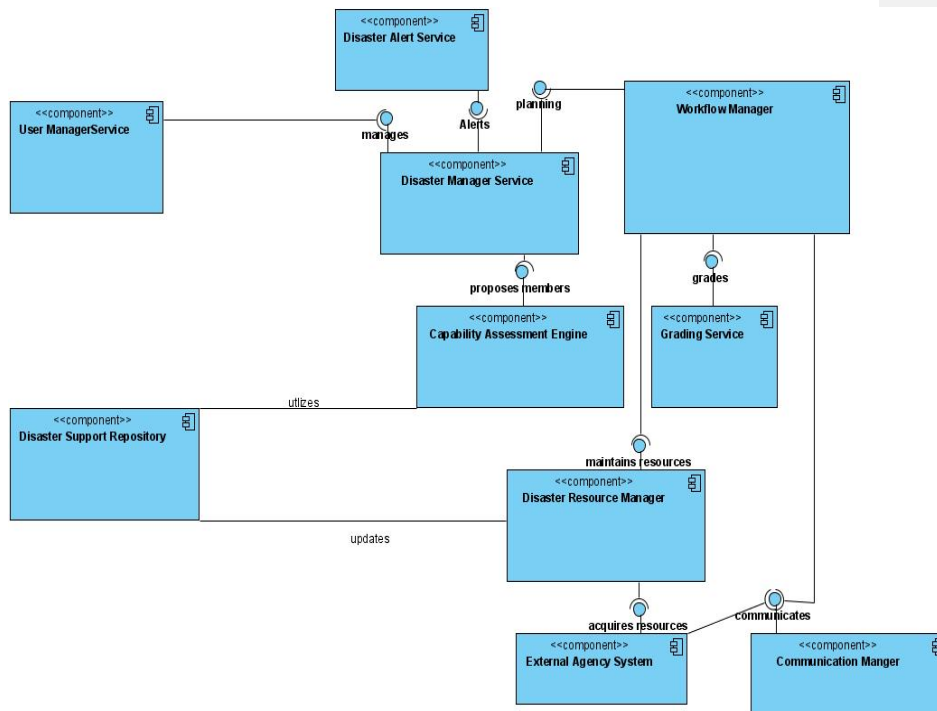


Figure 4-2 Component Diagram

Once a disaster occurs the user enters parameters for the disaster to setup a work flow/plan for rescue and relief efforts. The project is open till the time the tasks given to various teams are not completed. When a project is open it is constantly monitored and status of different teams and tasks are updated regularly once all tasks are performed and are reported as complete the project is closed. Then the tasks are reviewed whether a team performed the task successfully or not or whether the results of the tasks were satisfying. Based on these results the teams are graded. This is helpful in future activities where task allotment could be made easy by allotment of tasks on basis of team's past record and task's complexity level.

Once a rescue or relief operation is completed Grading service is used to grade organization on that particular task. This will not only help in evaluation of tasks performed and performance of different organizations but also help in future projects by giving complex tasks to more

competitive organizations. It will also help organization to learn and improve their operations.

The classes within the components are further connected to database repository through a single Java bean.

4.7.1 User Support Service

User Support Service aka Disaster Support Service is a web service provides functionality for system login/access as well as for user registration. When a user enters his name and password the web service sends message to the respective class in `core.disaster` package which checks the credentials with the database's table. If the credentials are correct user is logged in successfully and can access system functions according to his/her roles.

4.7.2 Disaster Manager Service

This web service is the gateway between the classes that maintain the disaster related information and the front end. It provides web methods for all information on different disaster projects like creating a new project, viewing existing projects, creating new tasks and viewing existing tasks, resource management, updating status on different tasks and inbox for viewing messages as well as creating messages.

4.7.3 Disaster Alert Service

This component is developed as a web service and it contains operations related to providing alerts and notifications to clients which have subscribed to DMSS.

4.7.4 Workflow Manager

This component contains classes which generate a workflow for a new disaster management project. The workflow tasks are then assigned resources and forwarded to respective teams.

4.7.5 Capability Assessment Engine

Capability Assessment Engine calculates capability of different teams based on their performance on different rescue and relief activities and how effectively they performed those activities.

4.7.6 Resource Manager

This component contains classes which are collectively used to manage resource. This component's classes are accessed by web methods defined in Disaster Manager Service. The different methods allow DMSS to manage resources it has acquired of different organizations and also functions for organizations to update their resources and their status.

4.7.7 Communication Manager

This component contains classes which handles communication between different organizations which are working on the same project. It provides functions to send messages to different organizations as well as to post information and updates on a message board.

4.7.8 Grading Service

Grading service is a web service consisting of functions defined in capability assessment engine. It's the interface between client and capability assessment engine, exposing all function which are related to grading an organization based on its performance in various activities.

4.7.9 External Agency System

This package contains classes which handles project related information. This package's classes are accessed by web methods defined in Disaster Support Service. The different web methods accessing this package are described in section 4.8.3.

4.7.10 Support Repository

This is the database of the whole software system. All the user, organization and project related data is maintained. It has been designed by keeping data integrity and confidentiality principles in mind. Also

database normalization principles were applied during database design. The database design is further explained in an E-R Model in section.

4.8 Use Case Diagram

The use case diagram of DMSS has been given as Figure 4-3. This diagram describes the interaction of user and the system.

4.8.1 Basic Flow of DMSS

The software has three types of users, admin, manager operations (MO) and organizations. All the three types of users have different access level to the system and its data and can perform functions assigned to their respective roles.

The admin user can create new user or organizations or register organization. It can also deem a user as active or inactive based on some policy. The MO user is basically the project manager; he/she can also manage more than one project at a time. It can perform all actions related to a project like project creation, appoint tasks, check task status, send receive messages and grade users/organizations based on their performance on a given task.

The organization user can view project and tasks it has been appointed to, it can also update the organization resources whether new resources are added or existing resources are occupied or busy. It can update MO on task status and can send and receive messages to communicate with others on the same project.

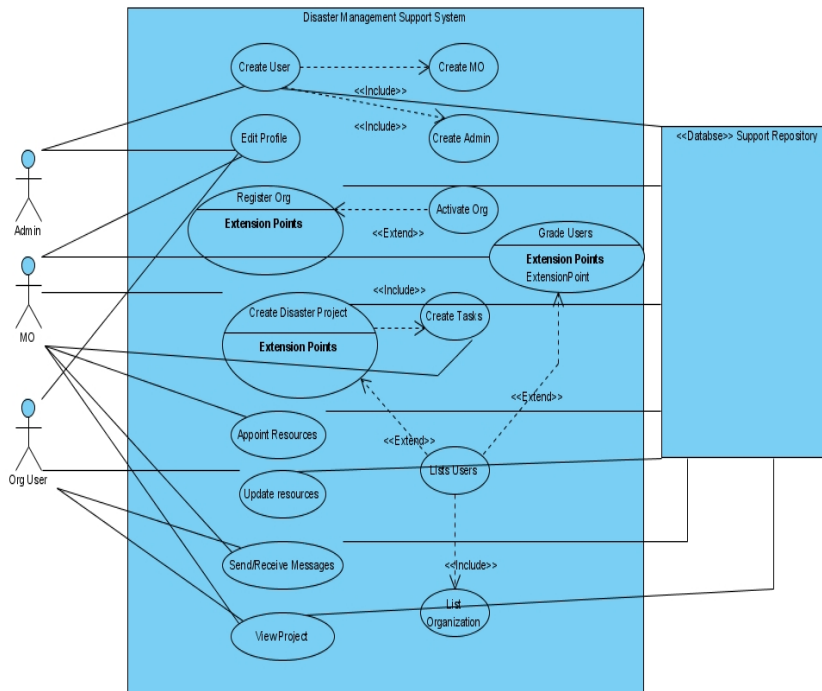


Figure 4-3 Use Case Diagram

4.8.2 Post Conditions

System has performed the action which was requested by user through web interface.

4.8.3 Alternate Scenarios

No alternate scenarios are defined because the objective of the system is described in chapter 1 and the software is being developed by following the given requirements.

4.9 Class Diagram

Figure 4-4 shows the class diagram of DMSS. All the classes shown in the class diagram are further described in the sub sections.

DisasterSupportService is the main service to provide system access to users and allow them to access different system functions based on their

roles. This service acts as an interface between web application's front end and back end Java classes.

DisasterDBAccess is the main class to access database objects. All the data operations are performed through this class. All other classes send or retrieve data from database through this class. It acts as an interface between all software classes and database system.

Login System class contains data and functions for users and organizations to login. This class simply authenticates user information and checks which role is assigned to user based on which user is given access to different areas and functions of the software.

OrganizationSystem class shown in Figure 4-4 provides data related to an organization. It also provides functions for MO to see list of organizations and narrowing search by different parameter inputs. Also this class has the important function for MO to register organizations. This class is inherited from User base class.

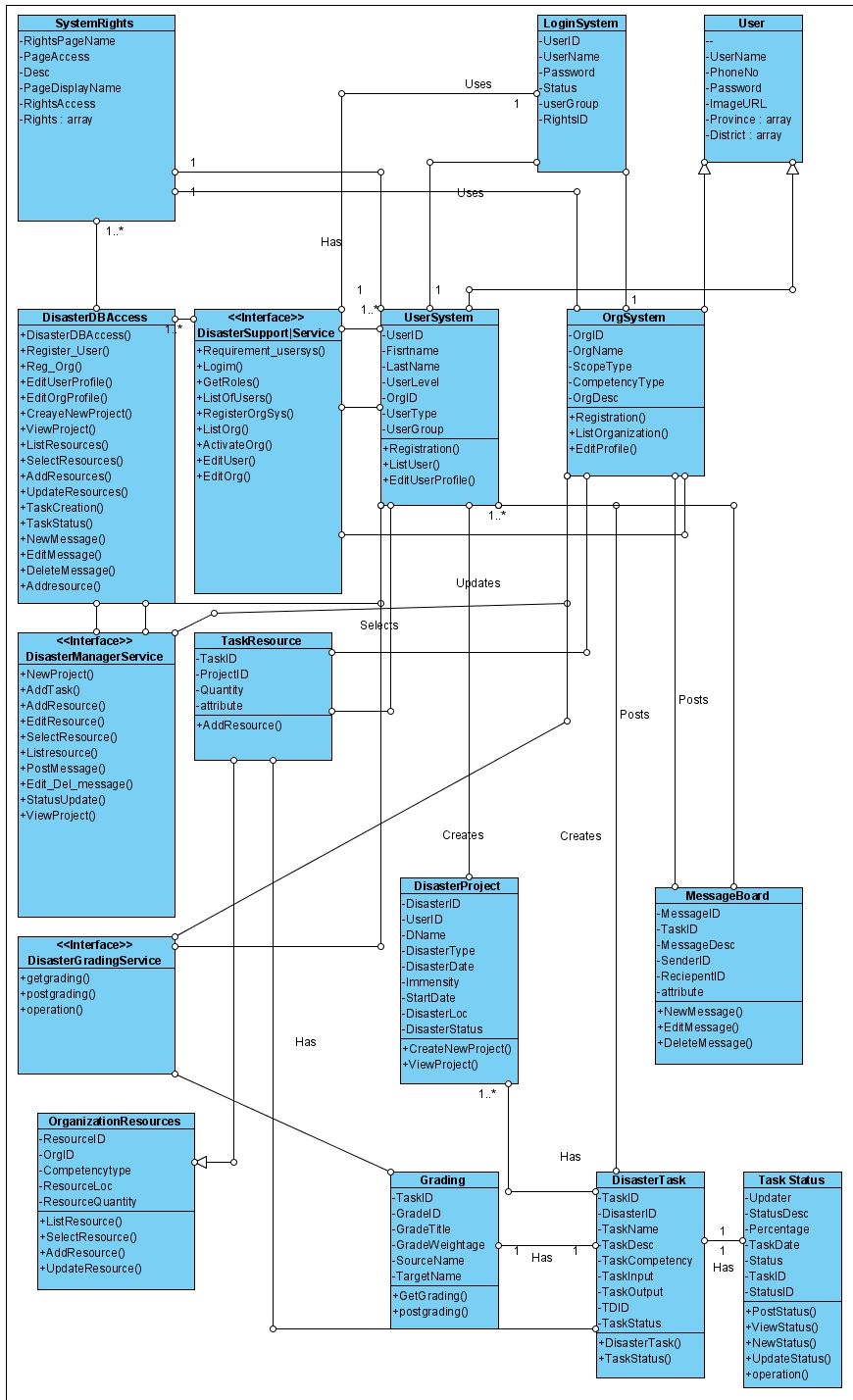


Figure 4-4 Class Diagram of Disaster Management Support System

The *System Rights* class has information roles and responsibilities for each user type. So all the roles and their details and functionalities offered for different roles are in this class.

The *User* class has basic information about user and organization. This is the base class which is extended by user system and organization system classes which have details specific to a user or organization.

User System class is extended from user and adds additional detail which is more specific to a user which in this case can be the Web application administrator or manager operations.

Disaster Manager Service is the second web service providing various Web methods which are used to provide functionality regarding a disaster. All functionalities of the system related to disaster management are exported as web methods by this service.

DisasterProject class contains data about a disaster project and functions to view existing project or create a new one.

Task class has all the data about a task which is related to some disaster project. It has functions to create new tasks and set up priorities or set up a workflow of different tasks which may or may not be interrelated.

Message board class keeps track of all messages, it provides functions to create new message, post message to desired authority and view messages.

Organization Resource class keeps a record of organization's resources. Whether a new resource is added, a resource is busy or free all information is maintained through this class. This class has a high importance in project because efficient resource management will lead to better performance in different tasks or activities on a project.

Task resource class keeps a track of resources which are currently allocated to a task. This class is extended from organization resource class to add the extra functionality if a task has acquired a resource.

Task Status class monitors status of an individual task and provides functions to set status, update status, and view status to concerned users.

Grading class is used for grading different tasks so to track performance level of different organizations and help shaping future decisions like task appointment and resource appointment to improve overall disaster management strategies.

Disaster Grading Service is the service which exposes grading service functions to authorized users

4.10 Entity Relationship Diagram

DMSS has a database repository to maintain all the records. ER Diagram shows how data is modeled and relationship between different data are identified and stored in a database management system. ER Diagram is shown in Figure4-5.

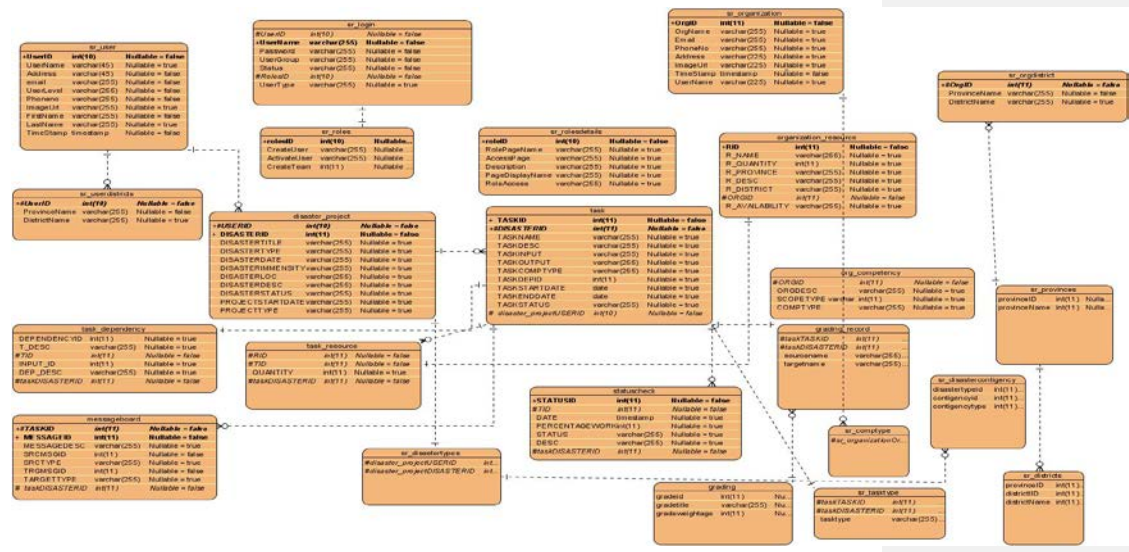


Figure 4-5 ER Diagram of Disaster Management Support System

Some important tables from DMSS database are described below.

The *sr_login* table contains all user information for logging into a system.

sr_organization table contains information about all organizations currently registered with the system or organizations waiting to subscribe to our web services.

disaster_project table contains all the basic information about the disasters. All disasters related records are kept in this database.

message_board table helps to maintain a record of communication taking place between organizations and manager operations. All the messages posted are logged by this database table.

4.11 Detailed Design

In order to ensure that the correct software is being build according to requirements defined in chapter 3; low level design of DMSS is build using requirements, component diagram and class diagram. In this section a few of these diagrams related to a particular system scenario are described. The rest of system diagrams are attached as Appendix A.

4.11.1 Activity Diagram: Task creation

The Figure 4-6 shows how the activity of task creation is performed. When a new project is created a list of tasks is created. Once a task is created it can be assigned to an organization and multiple resources can be assigned to it. After a task is appointed its progress is tracked. The exit can be by two ways , task is completed or aborted.

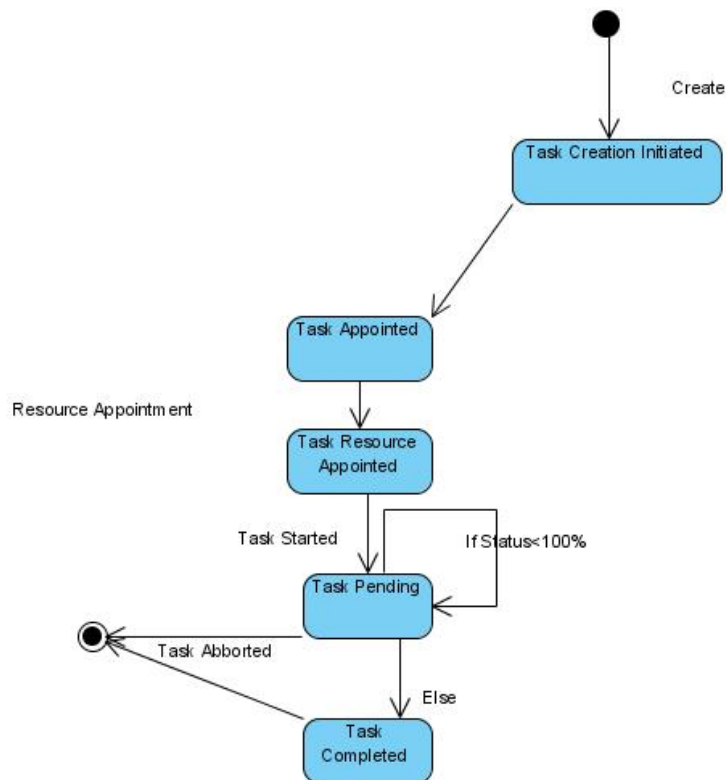


Figure 4-6 Activity Diagram of Creating Task Process

4.11.2 Sequence Diagram: Adding resource to task

Figure 4-7 shows the sequence of steps and timeline of different objects involved in adding resource to a task. First resource selection method is invoked that shows a list of available resources. Then the resources selected and database is updated. Once a resource is selected its assigned to a resource as shown in the timeline. Also the task creation process is completed and the respective organization is notified.

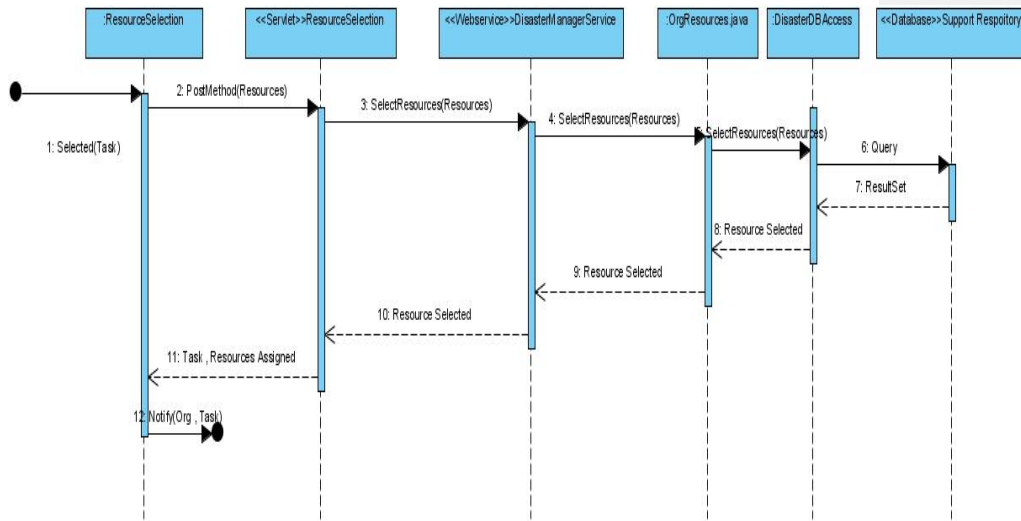


Figure 4-7 Sequence Diagram of adding resource to task

The above diagram shows the sequence of steps and timeline of different objects involved in adding resource to a task.

4.11.3 State Machine Diagram: Organization

State machine diagram shows the states an object can move between when it's active in a software system. The diagram below shows the different states an organization object can acquire in its lifetime.

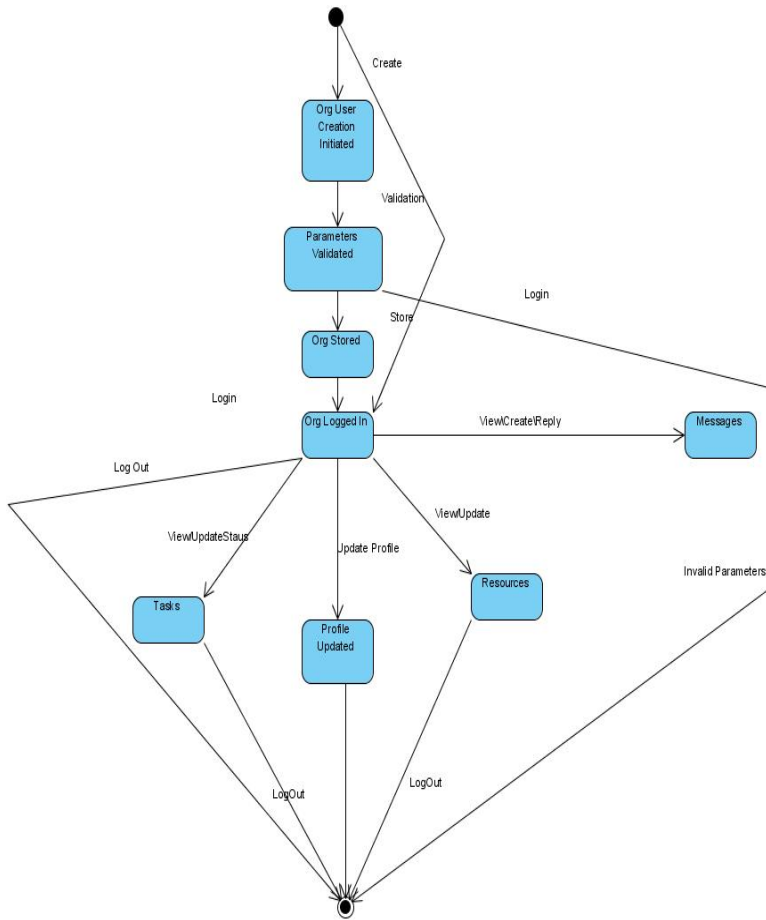


Figure 4-8 State Machine Diagram: Organization

4.12 Web Modeling

This section provides design details of web application interface. It includes navigation model and presentation model.

4.12.1 Presentation Model

Below figure 4-9 shows the presentation of different objects and navigation links on dashboard page of an admin.

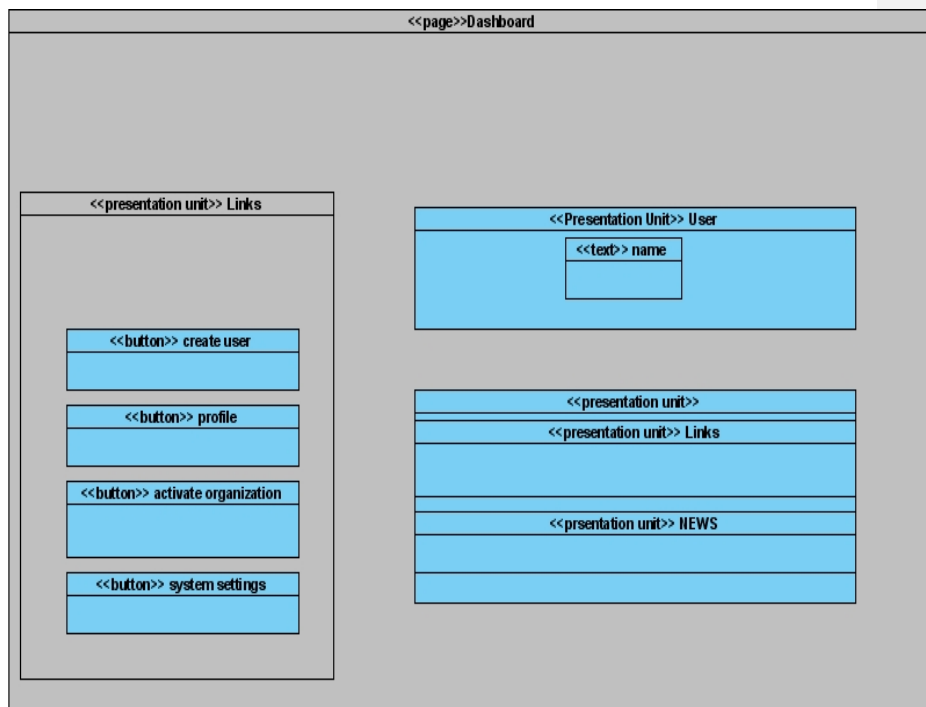


Figure 4-9 Presentation Model: Admin Dashboard

4.12.2 Hypertext Structure Model

Figure 4-10 explains the hypertext structure model for an organization user. The figure shows different classes associated with an organization user and organization user can navigate through in hypertext model. The organization user can access only the projects its currently working on through projects navigation class. The resource class basically keeps a record of all the resources an organization has and allows an organization to update resources while profile class allows it to change different attributes of its profile like business address, telephone number and so on.

Comment [AM2]: What is org, projects etc.)

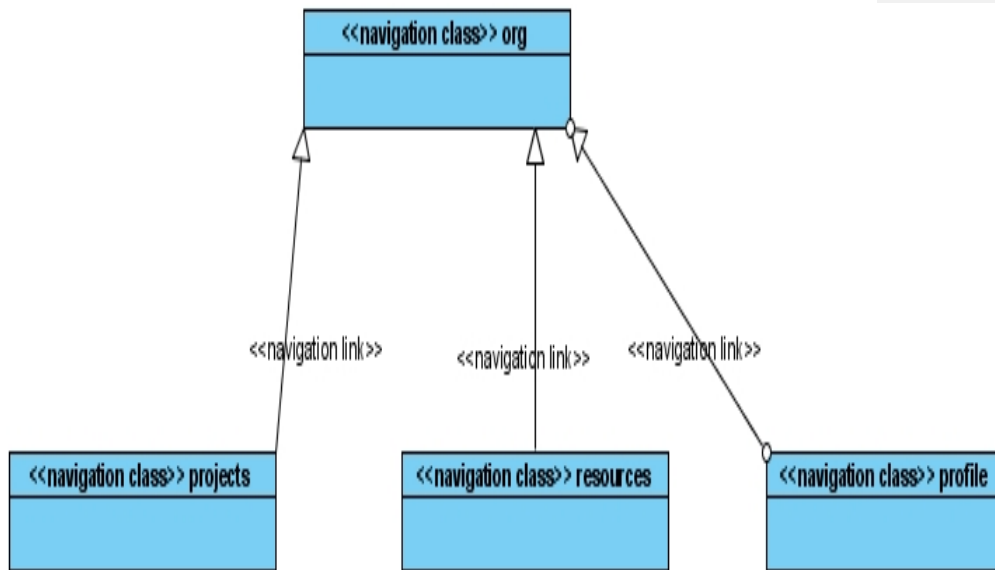


Figure 4-10 Hypertext Model: Organization

4.13 Summary:

DMSS has to perform in real time environment and has to be available 24x7 to its users. This chapter described the design of the software taking into consideration different assumptions and constraints that applied on the system because of its goals and requirements. Component Diagram, Class Diagram and Use case diagram have been added to explain system functionalities. ER Diagram has been added to explain database design and relationships between different database objects. A few low level design diagrams are shown to elaborate how system behaves internally and how objects change states when the system is put into operational environment.

Chapter 5

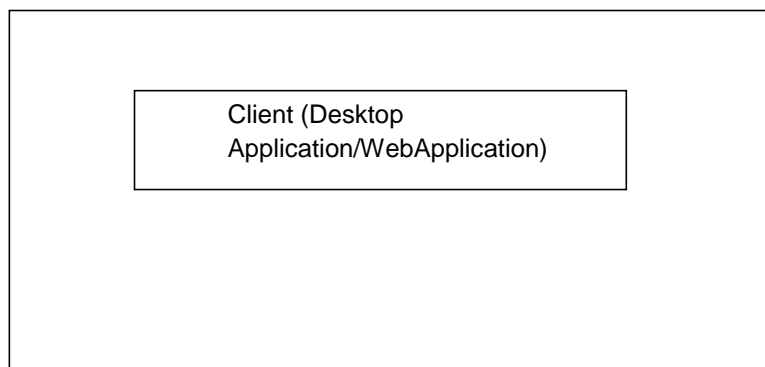
Implementation Details

5.1 Introduction

Detailed design of the Disaster Management Support System (DMSS) are discussed in the previous chapter. This design is transformed into an application by using various technologies. The implementation details are discussed in the following sections giving details of the system's internal working.

5.2 Tools and Technologies

Web applications can be implemented using various technologies. These include server side technologies (JSP, Servlets, PHP, ASP.NET) and different client side technologies (JavaScript). However, the selection of these technologies depends on the system architecture and its detailed design. Figure 5-1 shows an overview of different tools and technologies used in the DMSS along with their interaction. JSP provides the user interface of the application and handles the data exchange made by the user to the application. This data exchange is in the form of parameters passed to the system. The Servlet then initiates a call to a required function of the Web Service. The call is handled by the Web Service End Point. The core functionality of the system is handled by Java Classes at the Service End Point. JAVA Classes then communicates with the database to perform different transactions. The Web Service End Point returns result to the Servlet. The Servlet returns dynamic content to the JSP using JSON and jQuery.



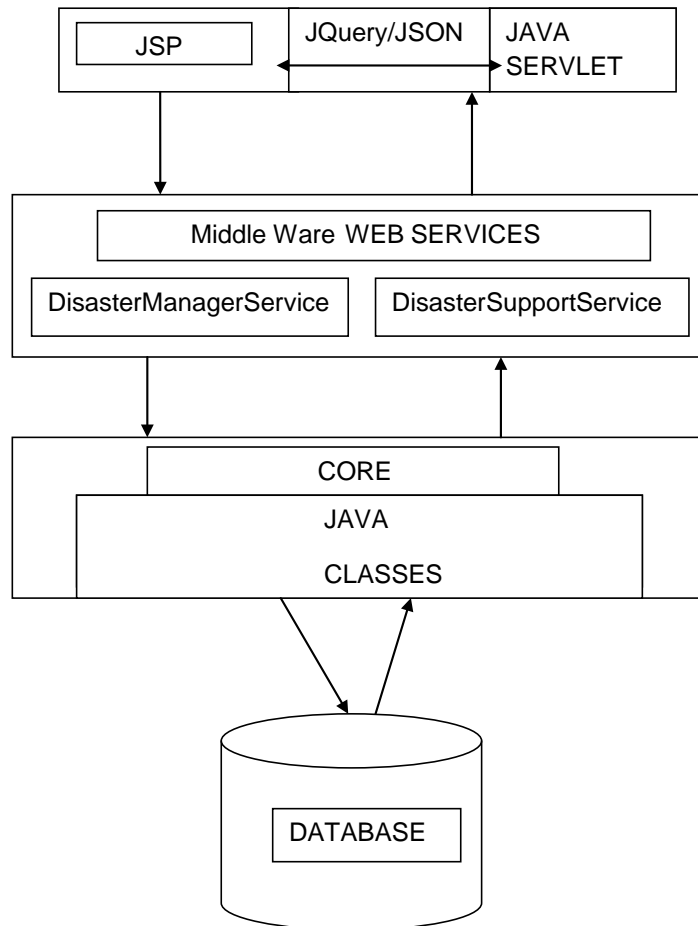


Figure 5-1 Overall System Diagram

The functionalities provided by the main system are exposed as Web services. JAX-WS has been used to develop web services instead of RPC web services. JAX-WS API provides more advantages as compared with AXIS-2 and JAX RPC. JAX-WS provides better platform independence and developer doesn't have any concern with implementation details of functions which are exposed in Web service. JAX-WS has support for SOAP 2.1. JAX-WS also has support for most XML schema since its data mapping model is based on JAXB [18].

5.2.1 Web Services

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks [6]. Web services have an interface and a client can first register with web service and then access the functionalities through its interface.

5.2.2 WSDL

Web Service Directory Language (WSDL) is Web Service's language which contains information about a specific web service. WSDL contains information like transport, protocol, service location, operations or functions available and payloads transferring to and from web service.

A client program connecting to a Web service can read the WSDL file to determine what operations are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the operations listed in the WSDL file using XML or HTTP. A WSDL document uses the following elements in the definition of network services:

Types— a container for data type definitions using some type system (such as XSD).

Message— an abstract, typed definition of the data being communicated.

Operation— an abstract description of an action supported by the service.

Port Type—an abstract set of operations supported by one or more endpoints.

Binding— a concrete protocol and data format specification for a particular port type.

Port— a single endpoint defined as a combination of a binding and a network address.

Service — a collection of related endpoints [7].

5.2.3 XML BINDING

XML data binding refers to a means of representing information in an XML document as an object in computer memory. This allows applications to access the data in the XML from the object [11].

XML Binding for Disaster Manger Service is given in figure 5-2 below.

```
<xs:schema version="1.0"
targetNamespace="http://DisasterManager.Core/">
<xs:element name="CreateNewTask" type="tns:CreateNewTask"/>
<xs:element name="CreateNewTaskResponse"
type="tns:CreateNewTaskResponse"/>
<xs:complexType name="CreateNewTask">
<xs:sequence>
<xs:element name="task" type="tns:disasterTask"
minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:sequence>
<xs:element name="return" type="tns:disasterTask"
minOccurs="0" maxOccurs="unbounded"/></xs:sequence>
</xs:complexType>
<xs:complexType name="disasterTask"><xs:sequence><xs:element
name="TaskID" type="xs:int"/><xs:element name="TaskDepID"
type="xs:int"/>
<xs:element name="DisasterID" type="xs:int"/>
<xs:element name="TaskPriority" type="xs:string"
minOccurs="0"/>
<xs:element name="TaskName" type="xs:string" minOccurs="0"/>
<xs:element name="TaskDesc" type="xs:string" minOccurs="0"/>
<xs:element name="TaskCompType" type="xs:string"
minOccurs="0"/>
<xs:element name="TaskInput" type="xs:string"
minOccurs="0"/>
<xs:element name="TaskOutput" type="xs:string"
minOccurs="0"/>
<xs:element name="TaskStatus" type="xs:string"
minOccurs="0"/>
<xs:element name="TaskStartDate" type="xs:string"
minOccurs="0"/>
<xs:element name="TaskEndDate" type="xs:string"
minOccurs="0"/>
<xs:element name="TaskType" type="xs:string" minOccurs="0"/>
<xs:element name="resourcelist" type="tns:taskResource"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence></xs:complexType></xs:schema>
```

Figure 5-2 XML Binding Disaster Support Service

The Figure 5-2 is a schema that populates global element and attributes of Core.DisasterManager namespace. The element CreateNewTask refers to the function exposed in the Web Service. The element task refers to an

object of the class DisasterTask. The object task is sent as a parameter in the method DisasterTask.CompleteType is used when transferring user defined objects or multiple data type. Between the complex types tags are defined the attributes of the object DisasterTask. Type refers to the data type of the attribute.

XML Schema for remaining web services are given in Appendix B.

5.2.4 SOAP MESSAGE

Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) for its message format, and usually relies on other Application Layer protocols, most notably Remote Procedure Call (RPC) and Hypertext Transfer Protocol (HTTP), for message negotiation and transmission. SOAP can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built. This XML based protocol consists of three parts: an envelope, which defines what is in the message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing procedure calls and responses [8].

It requires profuse attribute specification tags, namespaces, and other complexities, to describe exactly what is being sent. This has its advantages and disadvantages. SOAP involves significantly more overhead but adds much more information about what is being sent. If you require complex user defined data types and the ability to have each message define how it should be processed then SOAP is a better solution than XML-RPC.

5.2.4.1 HTTP GET AND POST MESSAGE

GET Requests a representation of the specified resource. Requests using GET (and a few other HTTP methods) "SHOULD NOT have the significance of taking an action other than retrieval". [9]

POST Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both [10].

5.2.4.2 SOAP MESSAGE EXAMPLE

The following is an example of a soap message from DMSS.

<p>Method parameter(s)</p> <table><thead><tr><th>Type</th><th>Value</th></tr></thead><tbody><tr><td>java.lang.String</td><td>null</td></tr><tr><td>java.lang.String</td><td>null</td></tr><tr><td>int</td><td>2</td></tr><tr><td>java.lang.String</td><td>MO</td></tr></tbody></table> <p>Method returned</p> <pre>java.util.List : "[core.disaster.UserSystem@147a765]"</pre> <p>SOAP Request</p> <pre><?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header/> <S:Body> <ns2:lsituser xmlns:ns2="http://Disaster.Core/"> <province>null</province> <district>null</district> <parameter1>2</parameter1> <usergroup>MO</usergroup> </ns2:lsituser> </S:Body> </S:Envelope></pre> <p>SOAP Response</p> <pre><?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:lsituserResponse xmlns:ns2="http://Disaster.Core/"></pre>	Type	Value	java.lang.String	null	java.lang.String	null	int	2	java.lang.String	MO
Type	Value									
java.lang.String	null									
java.lang.String	null									
int	2									
java.lang.String	MO									


```

<return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:userSystem">
  <UserName>rawalpindi</UserName>
  <Phoneno>11111</Phoneno>
  <PassWord>Ahwaz</PassWord>
  <Email>abc@gm.com</Email>
  <Address>rawalpindi</Address>
  <Province>Punjab</Province>
  <District>Faisalabad</District>
  <UserID>2</UserID>
  <FirstName>Ahwaz</FirstName>
  <LastName>Sadeque</LastName>
  <UserLevel>MO</UserLevel>
  <OrgID>0</OrgID>
  <UserType>NDMA</UserType>
  <RolesID>0</RolesID>
  <UserGroup>2</UserGroup>
  <Status>11111</Status>
</return>
</ns2:lsituserResponse>
</S:Body>
</S:Envelope>

```

Figure 5-3 SOAP Message Example

The example in figure 5-3 has two parts SOAP Request and SOAP Response. In soap request list user function is called with some parameters. It also send location of the resource i.e. Core.Disaster. In soap response the requested data is returned it calls the lsituserResponse function at client end it also initializes schema of array structure to return complex data i.e. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns2:userSystem">

5.2.5 JAX-WS

The Java API for XML Web Services (JAX-WS) is a Java programming language API for creating web services. JAX-WS uses annotations to simplify the development and deployment of web service clients and endpoints [13]. Figure 5-2 shows how JAX-WS API implements Web services. The communication between client and server takes place over HTTP using SOAP message. JAX-B is used for data mapping. It unmarshals the request of client and maps it to the appropriate function

described in web service interface.

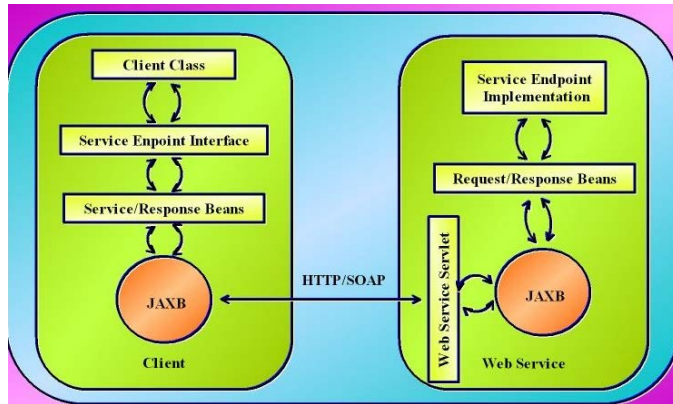


Figure 5-4 JAX-WS API In Action

5.2.6 JAX-B

Java Architecture for XML Binding (JAX-B) allows Java developers to map Java classes to XML representations. JAX-B provides two main features: the ability to marshal Java objects into XML and the inverse, i.e. to unmarshal XML back into Java objects. In other words, JAXB allows storing and retrieving data in memory in any XML format, without the need to implement a specific set of XML loading and saving routines for the program's class structure. [14]

5.2.7 SAAJ

The SOAP with Attachments API for Java (SAAJ) provides a standard way to send XML documents over the Internet from the Java platform. SOAP messages require considerable processing power and memory. All parts of a SOAP message must conform to XML rules for allowed characters and character sequences so binary data cannot be included directly. Furthermore, SOAP implementations typically parse the entire SOAP message before deciding what to do with the contents, so large data fields could easily exceed available memory. For all these reasons it was recognized that SOAP requires some mechanism for carrying large payloads and binary data as an attachment rather than inside the SOAP message envelope.

5.2.8 jQuery

jQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML. jQuery is free, open source software. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications [12].

5.2.9 JSON

JSON (an acronym for JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. The JSON format is often used for serializing and transmitting structured data over a network connection. [4]

5.2.10 AJAX

Asynchronous JavaScript and XML (Ajax) is a group of interrelated web development methods used on the client-side to create interactive web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. [15]

5.2.11 Java Servlet

A Java Servlet is a class which is used to extend capabilities of servers which are based on request response model. Servlets are used to extend applications hosted on web servers. It can be compared to an Applet; the only difference is that it runs on server instead of client machine. It is usually used to separate presentation logic from business logic. The table 5-1 below shows some of the Java Servlets implemented in DMSS along with their brief description.

Table 5-1 Java Servlets

Name	Description
AddResouce	This Servlet retrieves data from the addResource.jsp and calls the web service operation getResources.
PostGrade	This Servlet retrieves data from viewtask.jsp while grading the users. The Servlet calls the web service operation postgrading.
getResources	This Servlet is automatically called once AddTask page is loaded.The sevlet calls the web service operation

5.3 User Interface

Since DMSS is a web application so user interface is an important part of it and it is treated as a separate module. The issues addressed in the user interface design are that no functionality is hidden from user and data should be presented in a clear way to end user so nothing is missed by him.

The interface is developed using JSP, JQuery and AJAX. The interface is kept separate from business layer by use of Java Servlet. The data is parsed to the Java Servlet using JSON and servlet then communicates with server and performs the desired functions. CSS 3.0 is used so that most old browsers are compatible with the web application but still a few features won't show up correctly on browsers older than Internet Explorer 6.0.

JQuery is an open source JavaScript library used for client side scripting (Writing the program of web application which will run on client side). JQuery is used for form validation in user interface, so when a user is creating a new project for a disaster, registering into the system or using any other form for entering data he shouldn't enter a wrong value or type. This will ensure data is in its correct format.

Ajax standards for asynchronous JavaScript and XML. It is also used to write client side code of a web application and main purpose is to make web applications interactive. Ajax was used in development of user interface for DMSS. The reason to use Ajax was it sends data to and retrieves from server asynchronously without affecting the web page or redirecting user to a new page.

In DMSS JSON (JavaScript Object Notation) is used in conjunction with JavaScript to generate dynamic data. This data is then send or received from sever using AJAX (POST and GET HTML methods) to avoid redirection of user on other page.

5.4 User Management Module

This module covers all functionalities associated with a user. It provide functionalities to create a new user , edit user , authorize user to login into the system and display system features to a user based on his role in the organization. The user management module essentially consists of a web service which acts as an interface between client side code and server side code. This web service methods than interacts with java classes which further interact with database through a single database access class to make up the whole hierarchy.

The web service directory language (WSDL) for DisasterSupportService, the web service responsible for all functions related to a system user is shown in Figure 5-5.

```
<definitions targetNamespace="http://Disaster.Core/"
name="DisasterSupportServiceService">
<types><xsd:schema><xsd:import namespace="http://Disaster.Core/"
schemaLocation="http://localhost:8080/DissasterSupportSystemServer/DisasterSupport
```

```

ServiceService?xsd=1"/></xsd:schema></types>
<message name="Reg_UserSys"><part name="parameters"
element="tns:Reg_UserSys"/></message>
<message name="Reg_UserSysResponse"><part name="parameters"
element="tns:Reg_UserSysResponse"/></message>
<portType name="DisasterSupportService">
<operation name="Reg_UserSys">
<input
wsam:Action="http://Disaster.Core/DisasterSupportService/Reg_UserSysRequest"
message="tns:Reg_UserSys"/>
<output
wsam:Action="http://Disaster.Core/DisasterSupportService/Reg_UserSysResponse"
message="tns:Reg_UserSysResponse"/>
</operation>
</portType>
<binding name="DisasterSupportServicePortBinding"
type="tns:DisasterSupportService"><soap:binding
transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="Reg_UserSys"><soap:operation soapAction=""/><input><soap:body
use="literal"/></input><output><soap:body use="literal"/></output></operation>
</binding><service name="DisasterSupportService">
<port name="DisasterSupportServicePort"
binding="tns:DisasterSupportServicePortBinding"><soap:address
location="http://localhost:8080/DisasterSupportSystemServer/DisasterSupportServiceS
ervice"/>
</port>
</service></definitions>

```

Figure 5-5 WSDL Disaster Support Service

As shown in Figure 5-5, the message “Reg_UserSys” refers to the function name at the service endpoint, the port type disastersupportservice refers to the unique port in the wsdl document. Operation refers to the actions being performed i.e. Reg_User, Action refers to the location of the resource. The binding defines the message format and protocol details for operation and messages defined by a particular portType i.e. DisasterSupportService. Port name DisasterSupportServicePort is an individual endpoint specifying a single address binding.

5.5 Disaster Management Module

This module provides all functionalities that will help user in disaster management and manage different tasks and resources related to a specific disaster. It provides functions to create a new disaster, define a

workflow for different tasks related to a disaster, and appoint tasks to resources. Monitor project and task status and collaboration in form of a messaging system are also a part of this module. Also a grading system has been implemented to keep a track of performance of different organizations on different tasks.

```

definitions targetNamespace="http://DisasterManager.Core/"
name="DisasterManagerServiceService">
<types><xsd:schema><xsd:import
namespace="http://DisasterManager.Core/"
schemaLocation="http://localhost:8080/DissasterSupportSystemServer
/DisasterManagerServiceService?xsd=1"/></xsd:schema>
</types>
<message name="getResourceList">
<part name="parameters" element="tns:getResourceList"/>
</message>
<message name="getResourceListResponse">
<part name="parameters" element="tns:getResourceListResponse"/>
</message>
<portType name="DisasterManagerService">
<operation name="getResourceList">
<input
wsam:Action="http://DisasterManager.Core/DisasterManagerService/ge
tResourceListRequest" message="tns:getResourceList"/><output
wsam:Action="http://DisasterManager.Core/DisasterManagerService/ge
tResourceListResponse" message="tns:getResourceListResponse"/>
</operation>
</portType>
<binding name="DisasterManagerServicePortBinding"
type="tns:DisasterManagerService"><soap:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
<operation name="getResourceList">
<soap:operation soapAction=""/><input><soap:body
use="literal"/></input><output><soap:body use="literal"/>
</output>
</operation>
</binding>
<service name="DisasterManagerServiceService">
<port name="DisasterManagerServicePort"
binding="tns:DisasterManagerServicePortBinding"><soap:address
location="http://localhost:8080/DissasterSupportSystemServer/Disas
terManagerServiceService"/>
</port>
</service>
</definitions>

```

Figure 5-6 WSDL Disaster Manager Service

The wsdl snippet for this module's web service, DisasterManagerService is shown in figure 5-6 above.

As shown in Figure 5-6, the message "getResourceList" refers to the function name at the service endpoint, the port type

disastermanagerservice refers to the unique port in the wsdl document. Operation refers to the actions being performed i.e. getResourceList, Action refers to the location of the resource. The binding defines the message format and protocol details for operation and messages defined by a particular portType i.e. DisasterManagerService. Port name DisasterManagerServicePort is an individual endpoint specifying a single address binding.

5.6 Disaster Support Repository

This is the database system for DMSS which contains all the records related to different users, user type, project, tasks, messages, task status. The database has been designed focusing on data integrity and confidentiality.

The database has been developed using MySQL version 5.0. The reason for choosing MySQL RDBMS was that its available as a free standalone version and DMSS has to be a free software whose functionality can be enhanced by different open source communities. Prepared statement objects and JDBC is used for SQL execution and result set object contains the final object.

Prepared statement objects are used to compile SQL statement. These statements then can be executed multiple times. Prepared statement help in security by separating SQL from data. This separation helps to prevent SQL Injection attacks. Figure 5-7 shows how a prepared statement works. The query from prepared statement object is parsed to generate SQL query which is then sent to JDBC driver which compiles and then sends the query to MySQL server which executes the results. The results are then stored in a 'result set' objects.

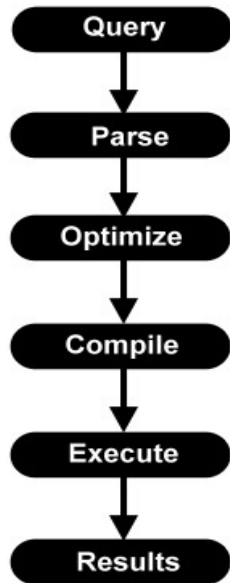


Figure 5-7 Prepared Statement Execution

5.7 Grading Service

This service basically provides an interface for trust engine which is used to grade how different organizations have performed rescue and relief activities. (discuss parameters).

5.8 Disaster Alert Service

This service basically is a messaging service which provides alerts to organizations that have subscribed to DMSS. The alerts include the information about a new disaster along with the location and scale of the disaster.

5.9 Summary

Implementation details of DMSS are discussed in this chapter. Different functionalities and strategies to develop the system are also discussed. A brief introduction to different tools and technologies employed is also given.

Testing**6.1 Introduction**

To ensure quality of the product, testing is conducted. Accuracy of functions performed by DMSS has to be tested and maintained to improve quality of software. Software testing techniques and results obtained are discussed in the coming sections.

6.2 Testing Levels

Separate modules are developed to provide different functionalities of DMSS. All of these modules are tested at different levels in their development and after integration. Different levels at which DMSS has been tested and results obtained are described in this section.

6.2.1 Unit Testing

Each module is developed and tested individually. Different sets of sample data are used to test all functionalities. The module for user login was developed first. This module is tested for both user login and organization login. Setting up different organizations and user's and admin the login module was tested to see if each type of system user gets has the correct responsibilities assigned and each user type should only be able to view system functionalities he has been assigned to. Also no user should be able to view data on another user. All the tests confirmed that data integrity and confidentiality is maintained and user(s) only see information intended for them. Another aspect is that user shouldn't be able to access any page and session handling was done at all levels. Different tests confirmed that accessing a restricted page without authentication is not possible and user is redirected to login page.

Project creation is the second module developed which has sub modules of workflow management; appoint tasks, communication system, resource management. All these sub modules are developed and tested using relevant sample data and then tested at integration level to complete the whole project module. Regression testing is used so that at any level of project creation no wrong data is fed into the system. The expected output

was that once all steps of project creation are followed, the system should show all data related to a project and shouldn't miss any relevant data. Also system should be able to generate reports regarding a project by a number of ways; for example if we want system to display resources it should be able to do that by city, district or province level. All the tests were successful showing that data entry and output from system was correct in all fashion.

6.2.2 Integration Testing

DMSS's different modules which were developed and tested independently were also tested during integration to ensure system stability. Integration testing helped in ensuring that different modules when combined give complete functionality and nothing is missed or some functionality doesn't give error when integrated with other modules. Integration testing gave us more than 90% results ensuring that most modules were integrated with others as well as compatible. This shows that errors were minimized during integration testing.

6.2.3 System Testing

System testing was performed at the end of development and integration of DMSS. Complete system was tested using sample data. User registration, user roles and responsibilities, creating new project and monitoring project all sub modules were tested as a whole using sample data. Almost 90% of test cases were successful ensuring that most of errors and bugs in the system were removed and system was stable enough to perform optimally.

6.3 Box Approach

To test whether DMSS functionality is in accordance with the code written box approach testing was done. The two box testing approaches used to test software were black box testing and white box testing. Black box testing is done at various stages of testing by inserting sample data in various components, checking outputs and removing errors. White box testing is done when a separate module is developed (unit level), integration of different modules and sub-modules and at system level.

Once the system is developed white box testing was the most performed testing technique.

6.4 Test Cases

The system is thoroughly checked for consistency and for errors using different test cases. Overall system is checked for the following attributes; parameters and resources are displayed correctly in given scenarios, the forms are validated correctly, session handling is done correctly, user interface is easy to use and users get features and options defined by their roles.

6.4.1 Test Case 1

Each form is checked to ensure that for a given a set of inputs the expected output was received. The related test case is shown below:

Scenario: In the Create New MO page if a user selects a province then the options in the district dropdown are loaded based on that.

Given Input: Punjab

Expected Output: All districts filed under Punjab come up in the districts dropdown

Actual Output: All districts filed under Punjab come up in the districts dropdown.

Test status: PASS

6.4.2 Test Case 2

Each form has been validated using the JQuery validation plugin. If a required field is missed or a wrong input is filled in then the form is not submitted and an appropriate error message is displayed. The related test case for one of the scenarios is given below:

Scenario: In the Create New MO page if a field is missed or text is filled in the phone number field which should be all digits an error message should be displayed.

Given Input: Text input in phone number field and form submitted.

Expected Output: Error message is displayed and form is not submitted.

Actual Output: Error message displayed and form not submitted.

Test status: PASS.

6.4.3 Test Case 3

Whenever a user logs in a new session is created. If that session times out or is corrupted the user is logged out of the system and has to sign in again. Also without a valid session ID a user cannot access any page.

Given Input: Username and Password submitted.

Expected Output: Session ID and Cookie are created in browser's temporary file.

Actual Output: Cookie created with a session ID as shown in figure 6-1.

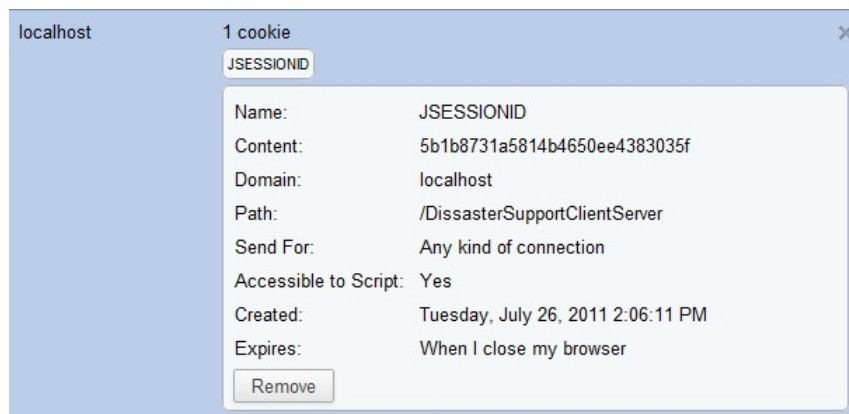


Figure 6-1 Session Cookie

Test status: PASS.

6.4.4 Test Case 4

Five students who had not used the system before were given different scenarios and told to execute them. They successfully navigated the system and were able to accomplish their tasks with little or no assistance.

6.4.5 Test Case 5

There are three different types of users who can log into the system: Admin, MO and Organization. They each have their own roles and there are different features which are accessible only to them. To ensure that no user type got access to any page restricted for them rigorous testing was done.

Scenario: While logged in as MO the system settings page was tried to be accessed through the address bar.

Expected Output: Error displayed

Actual Output: Error displayed

Test status: PASS.

6.5 Summary

Testing not only maintains the software quality but also improves over all usability of the project. At different stages of development suitable testing techniques were used to ensure product works accurately and efficiently. All errors detected during testing were removed.

Chapter 7

Results and Analysis

7.1 Introduction

DMSS has been developed to work in real time environment. Since disaster can happen any second DMSS should always be available to help organizations track disasters and plan rescue and relief operations efficiently. The data integrity and confidentiality should be maintained at all levels. Another aspect of DMSS is that should be platform independent and support interoperability.

7.2 Results

DMSS has been developed to facilitate organizations in post disaster activities or rescue or relief. The idea was to develop a web application which is at the center of disaster management activities of rescue and relief and helps organization to execute these activities efficiently and effectively. DMSS performs all the functionalities defined in chapter 3 system requirements.

All the functionalities have been achieved using the most advanced tools and technologies for development of Web services. Older web services technology like xml-rpc and API like jax-rpc have been ignored for SOAP message based communication. This will help in further extension of existing functionalities easily and reusability of code. SOAP message can be modified to include every communication details and protocols and is more verbose than RPC; this will make improve communication with different clients that have subscribed to DMSS.

7.3 Analysis

Since DMSS is a web application performance, robustness and usability are important features.

DMSS code is optimized so that page loading time is minimum and has been tested using multiple connections to server to test its load and stress and how system will perform. The results are more than 90% accurate showing it will perform fairly well with multiple client connections.

Usability is an important aspect of web application and in DMSS from the design phase this issue was paid special attention by developing navigation model for different web pages to show how information will be displayed and how different pages will link to each other.

Another important part of analysis is how DMSS compares with existing system. One of the existing systems that exist is SAHANA FOSS Disaster Management System. The functionalities that DMSS has and SAHANA doesn't are a central database of resources for all organizations. SAHANA also was developed for rehabilitation efforts and then extend to rescue and relief activities.

SAHANA isn't based on SOA, which means existing solutions of all independent organizations won't be able to fully integrate with it. With web services DMSS will save crucial times of other organizations and they can easily integrate with our system. DMSS also is focused on one single IT infrastructure for collaboration between all organizations an objective on which it was built and SAHANA wasn't.

7.4 Summary

DMSS performs all the functionalities functional and nonfunctional provided in the system requirements. All the important nonfunctional requirements that are essential for a web application to perform effectively are present in the system.

Other systems that are similar to DMSS are built on older web technologies and do not use SOA. Also the scope or the disaster management activities those systems perform are different from DMSS.

Chapter 8

Conclusion and Future Work

8.1 Introduction

This chapter introduces the achievements of DMSS. Suggestions have been presented in further sections for improvement of features and adding

new features to DMSS. DMSS has been developed in such a way to make it easy to expand and reuse code. This will help in extension of DMSS to cover further disciplines of disaster management and also how web services can be used to solve real life problems in an effective way.

8.2 Concept

DMSS concept was initiated by the large amount of disasters humanity has faced in recent past and using web application for better management of resources to tackle such conditions. The existing solutions as explained in chapter 2 had limitations which were basically the technologies used for solutions of disaster management weren't compatible with each other. Using web services to develop DMSS has helped to make different technologies.

8.3 Future Work

DMSS is a start to a wide field of using disaster management DMSS can achieve a lot of milestones that can help people manage post disaster activities and resources effectively. Although Google Maps are already integrated into the system but maps could be made interactive by tagging resources and during disaster tagging which organization is located where or doing operations in which area.

Workflow plans to organizations, thus allowing organizations to develop their own plans for the tasks they have been assigned to. Implementing a Mailing and SMS service for better communication between involved parties. This can further include applications for mobile devices like tablets and smart phones. The scope of system can be extended to include rehabilitation tasks and maintaining a database of shelters. Live streaming can be another future work so the possibilities to extend this system are limitless and without doubt will help in better management of post disaster activities of relief and rescue.

Dynamic team formation can also be included in the system where a system can suggest on its own the best possible team for a particular

disaster. For this purpose rule based system can be developed to choose appropriate teams for a particular disaster.

8.4 Summary

Web services can be used conveniently to help us in real life problems like disaster management. There is a wide range of options in web services to allow organizations to collaborate and work towards the common goal of disaster management. This will not only help in better management of post disaster activities but also prevent another disaster of mismanagement which can lead to not only issues of wasting resource but the bigger picture of losing more human lives.

Appendix A

Low Level Design

Sequence Diagram

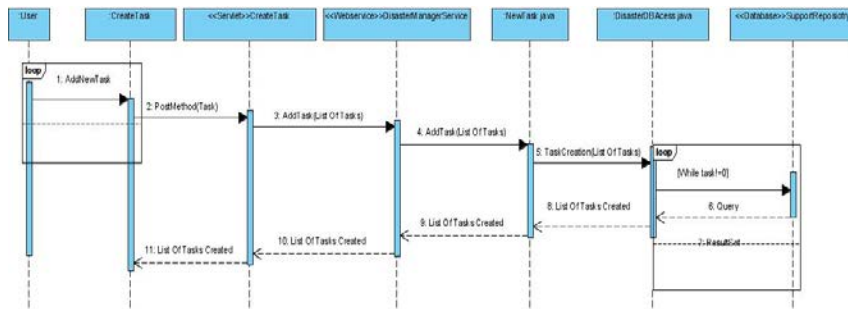


Figure 1 Add Resource

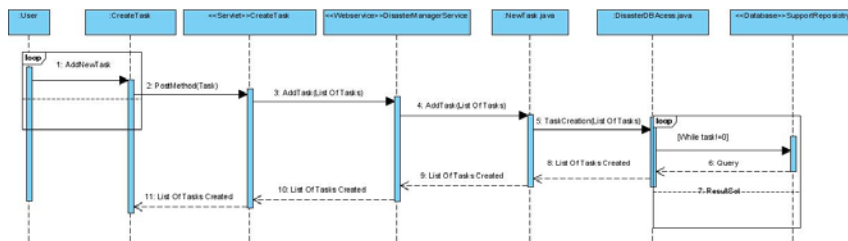


Figure 2 Add Task

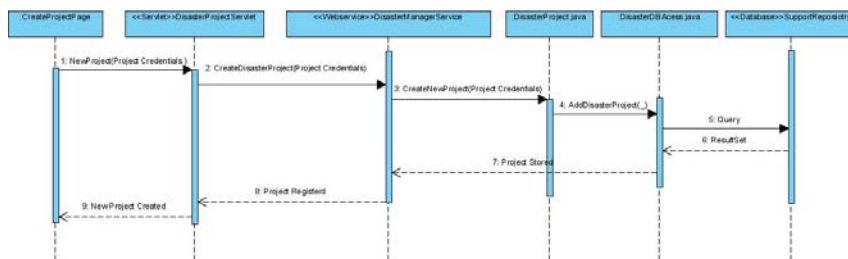


Figure 3 Create Disaster Project

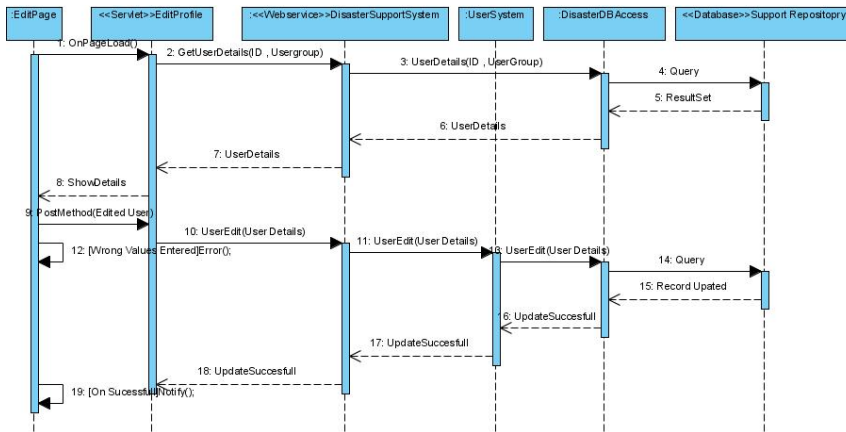


Figure 4 Edit User Profile

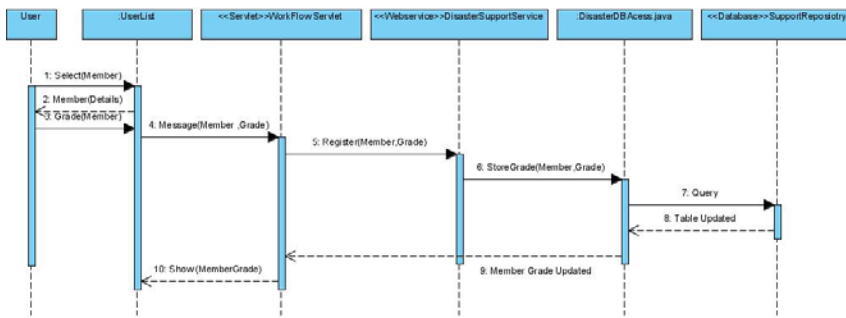


Figure 5 Grade Peers

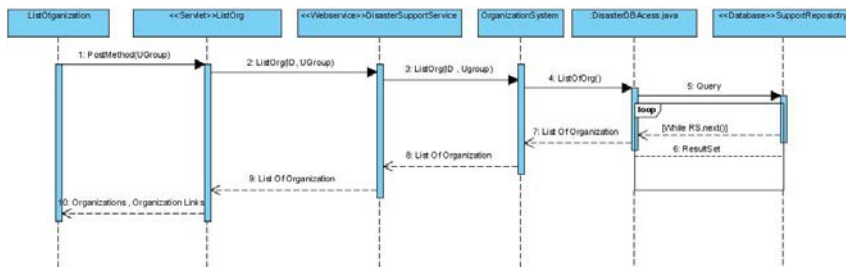


Figure 6 List Organizations

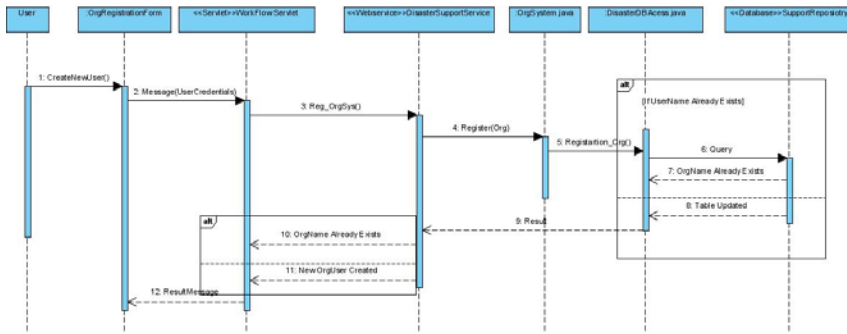


Figure 7 Register Organization

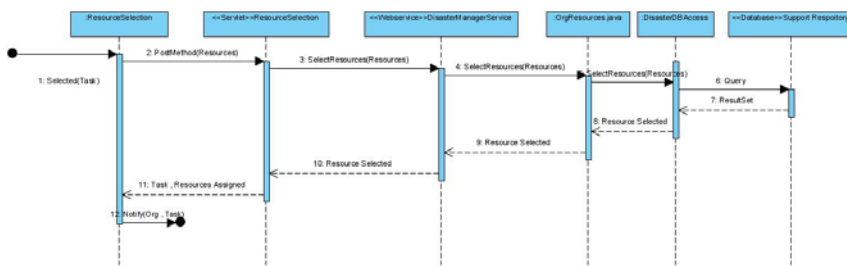


Figure 8 Resource Appointment

Communication Diagram

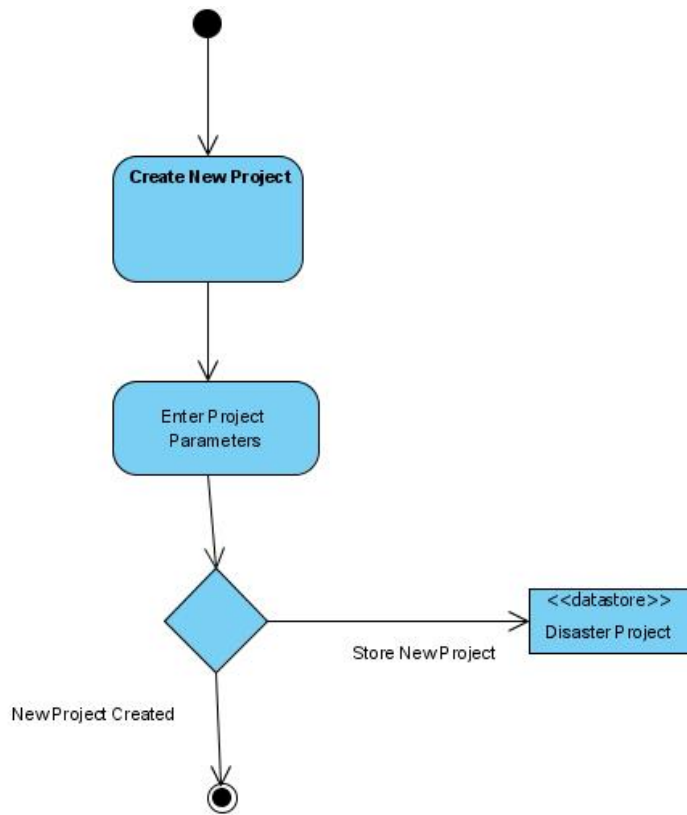


Figure 9 Create New Project

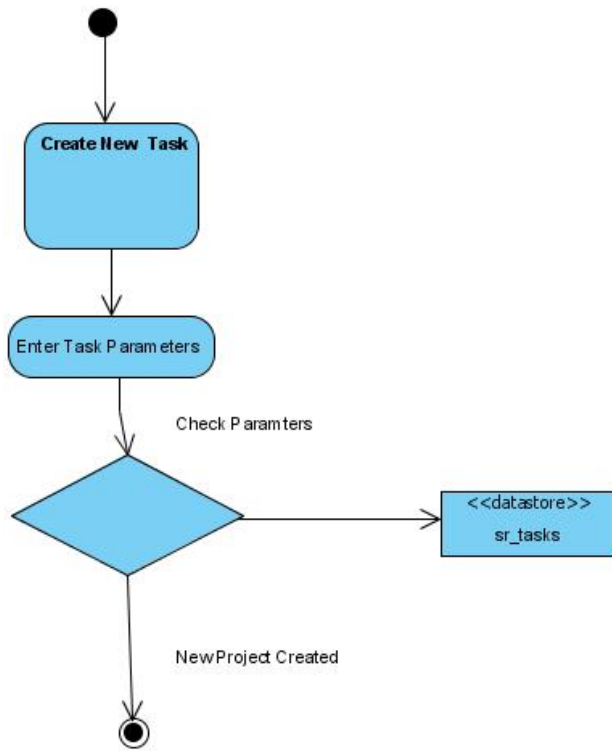


Figure 10 Create New Task

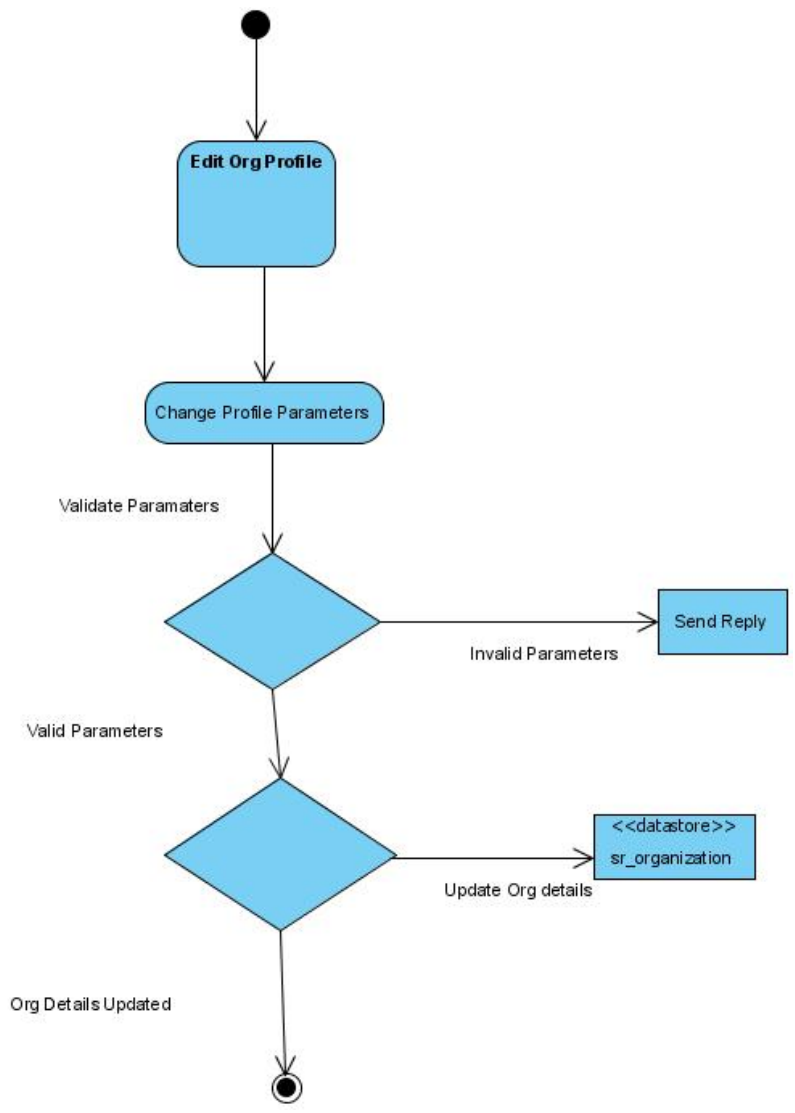


Figure 11 Edit Organization User Profile

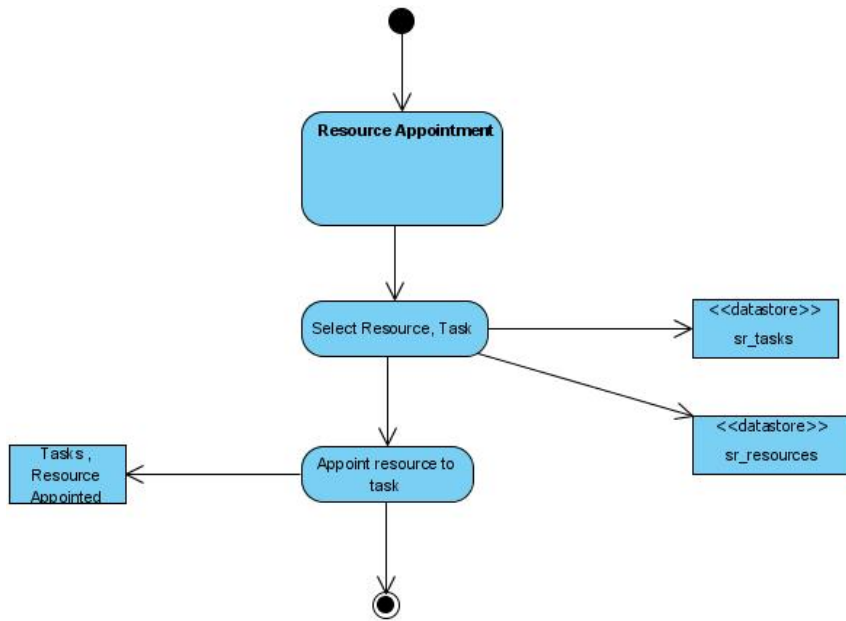


Figure 12 Add Resource to Task

State Machine Diagram

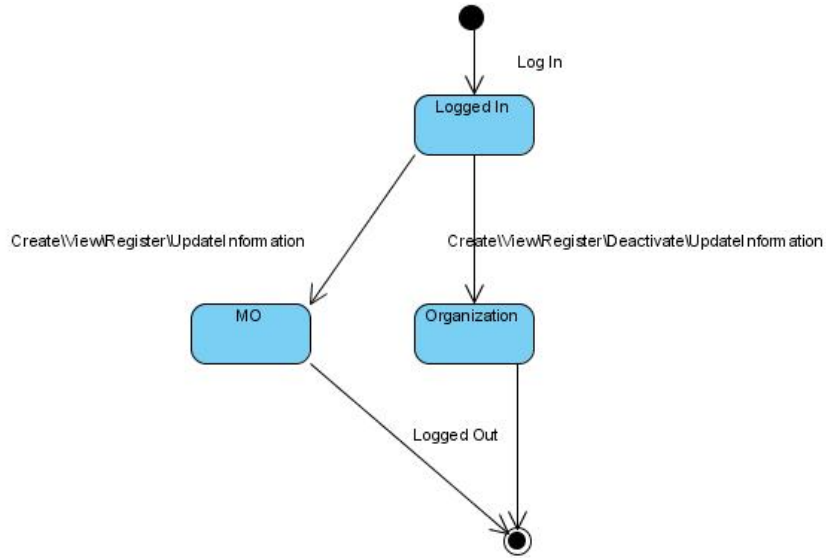


Figure 13 Manager Operation User

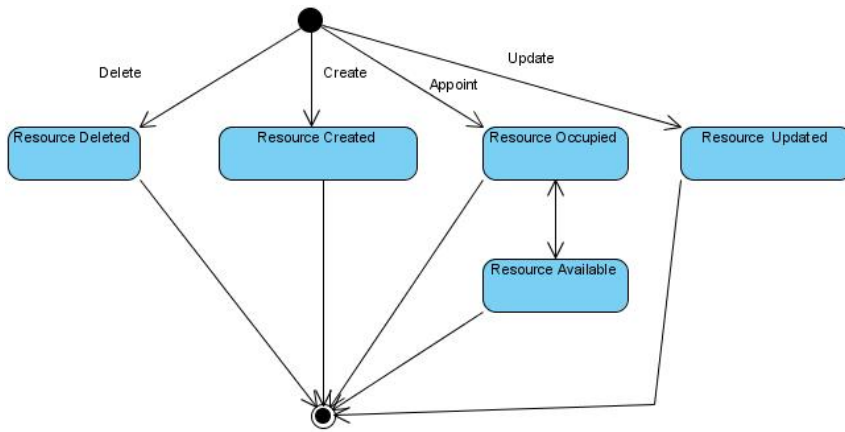


Figure 14 Resource

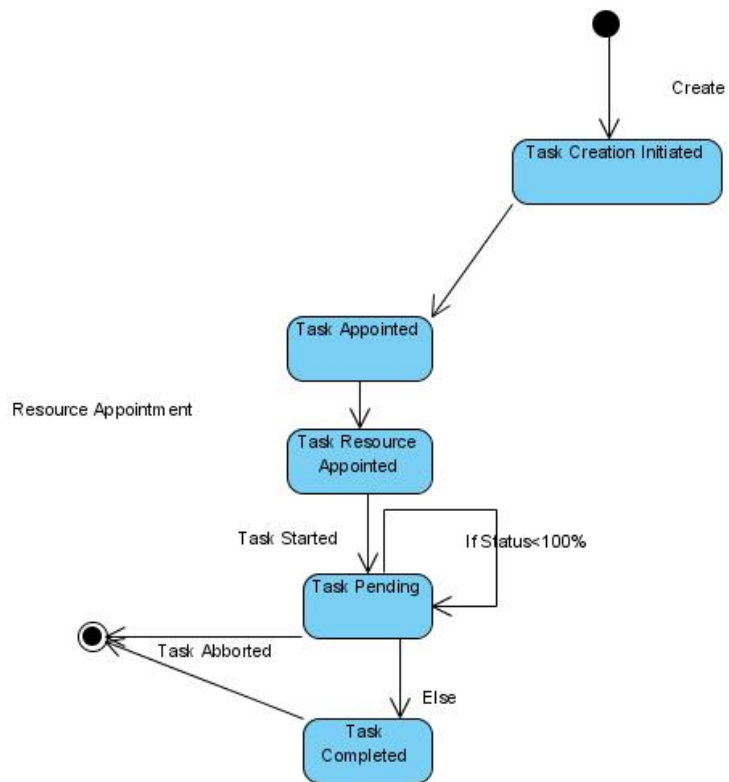


Figure 15 Task

Appendix B

User Manual

Overview of User Manual for Disaster Management Support System

This is the user manual for Disaster Management Support System. DMSS is a web application having three different navigation pathways for three different user types: Admin, Manager Operation (MO) and Organization.

B.1 Navigation Path for Admin

B.1.1 Login

Figure 1 shows the login page which is same for all three user types. It has an additional link for organization to create a new user which is only active after Admin verifies the request.

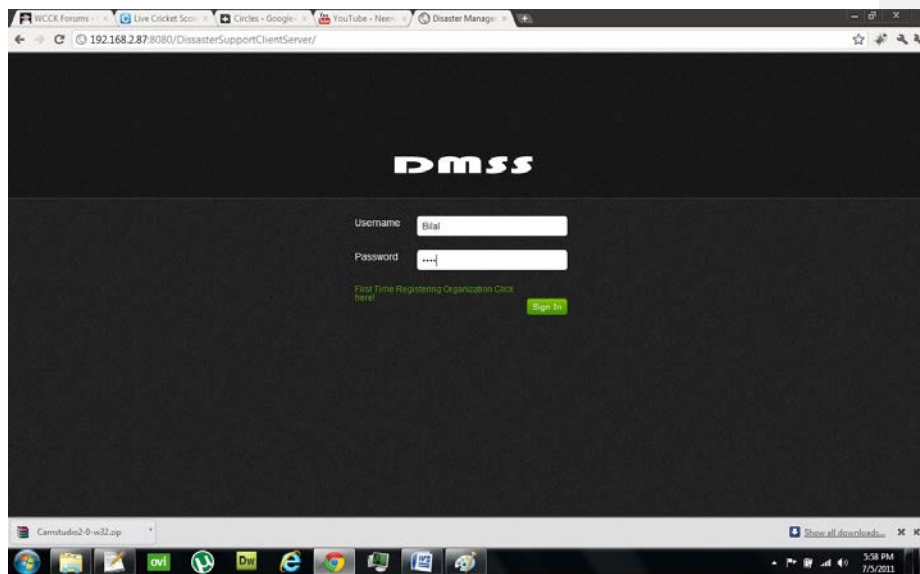


Figure 1 Main Login Page

B.1.2 Admin Dashboard

This is the dashboard for admin providing admin with different functionalities that only he/she can access. Figure 2 shows main dashboard for admin. Only admin can create new users and activate organizations and change system changes.

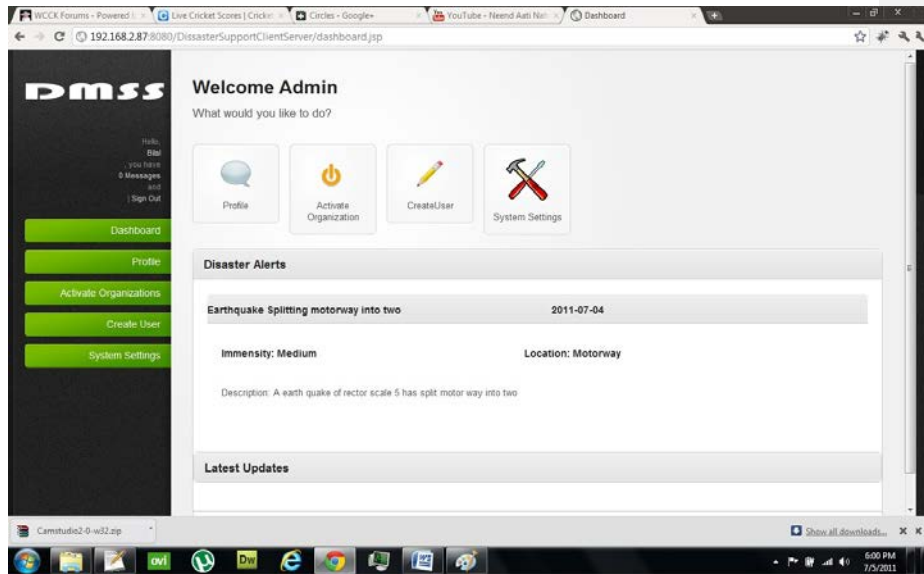


Figure 2 Admin Dashboard

B.1.3 System Settings

This is system settings panel only admin can view this page and change different system settings.

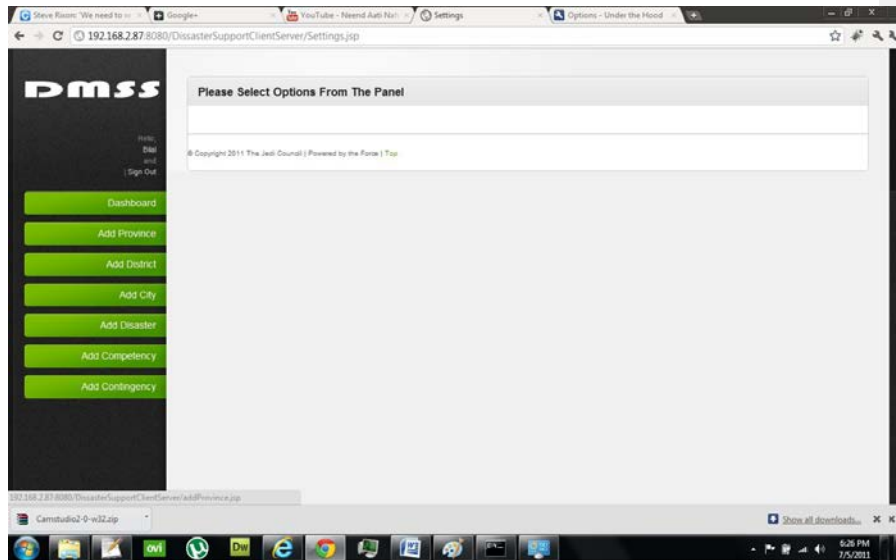


Figure 3 Admin Dashboard

B.1.4 Create MO

Only admin can create a new MO user through this page shown in Figure 4. The fields with asterisk must be filled. After entering data press submit to complete action.

The screenshot shows a web browser window displaying the 'Create New Manager Operations' form. The browser's address bar shows the URL: 192.168.2.87/DisasterSupportClientServer/CreateUser.jsp. The form is titled 'Create New Manager Operations' and includes a 'Form' button in the top right corner. The form fields are as follows:

- Username ***: Input field containing 'nofel'.
- Password ***: Input field containing 'Wt!Sn&7'.
- First Name ***: Input field containing 'nofel'.
- Last Name ***: Input field containing 'elahi'.
- E-mail ***: Input field containing 'nofelalahi@gmail.com'.
- Phone No. ***: Input field containing '032140350'.
- Address ***: Input field containing 'MCS, Rawalpindi'.
- Select from the following user levels ***: Dropdown menu with 'Manager Operations' selected.
- Select from the following organizations ***: Dropdown menu with 'NDMA' selected.
- Province ***: Dropdown menu with 'Punjab' selected.
- District ***: Dropdown menu with 'Rawalpindi' selected.

At the bottom of the form is a green 'Submit' button. The browser's taskbar at the bottom shows the system tray with the date and time: 6:11 PM, 7/5/2011.

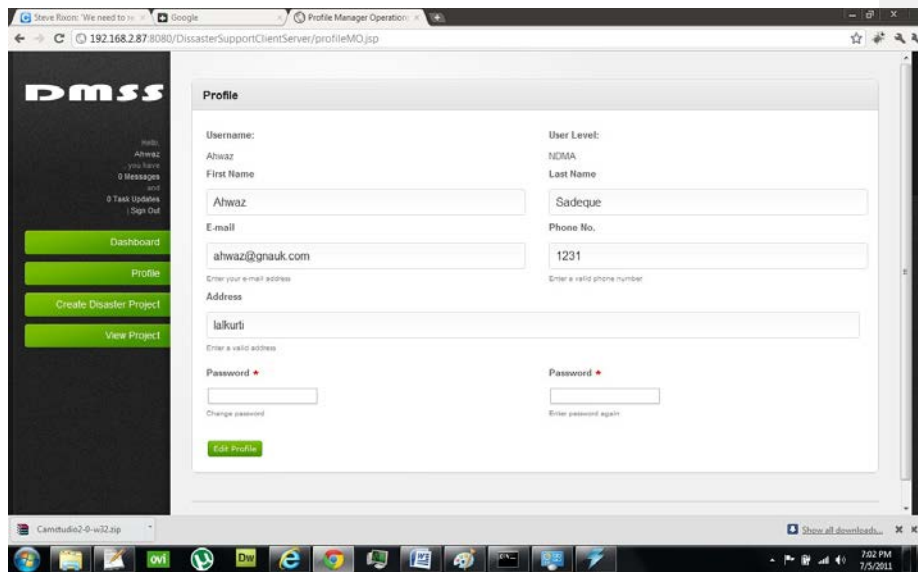
Figure 4 Create New Manager Operation

B.2 Navigation Path for Manager Operation (MO)

This sections shows navigation paths and different functions a manager operation user can perform.

B.2.2 Manager Operation Profile

The Figure 5 shows the profile options and editing page for MO. The fields with asterisk must be filled. After entering data press submit to complete action.



The screenshot shows a web browser window displaying the 'Profile Manager Operation' page. The page title is 'Profile Manager Operation' and the URL is '192.168.2.87:8080/DisasterSupportClientServer/profileMO.jsp'. The page features a dark sidebar on the left with the 'DMSS' logo and navigation links: 'Dashboard', 'Profile', 'Create Disaster Project', and 'View Project'. The main content area is titled 'Profile' and contains a form with the following fields and values:

Field	Value
Username	Ahwaz
User Level	NDMA
First Name	Ahwaz
Last Name	Sadeque
E-mail	ahwaz@gnauk.com
Phone No.	1231
Address	lakurbi
Password *	
Password *	

Below the form, there is a green 'Edit Profile' button. The browser's taskbar at the bottom shows the system tray with the time 7:02 PM and date 7/5/2011.

Figure 5 MO Profile

B.2.3 Create New Project

Only a manager operation user type can create a new project. Selecting the new project link will open a form to fill, the options help in developing a workflow for a new disaster. The fields with asterisk must be filled. After entering data press submit to complete action.

The screenshot shows a web browser window displaying the 'Create New Project' form. The browser's address bar shows the URL: 192.168.2.87:8080/DisasterSupportClientServer/NewProject.jsp. The form is titled 'Create New Project' and contains the following fields and options:

- Project Name ***: Text input field containing 'Floods'.
- Select from the following provinces ***: Dropdown menu with 'Punjab' selected.
- Select from the following districts ***: Dropdown menu with 'Attock' selected.
- Select from the following cities ***: Dropdown menu with 'Attock' selected.
- Location ***: Text input field containing 'Soan River'.
- Select the level of disaster immensity ***: Dropdown menu with 'Low' selected.
- Select from the following disaster types ***: Dropdown menu with 'Floods' selected.
- Today's date ***: Text input field containing '07-06-2011'.

The left sidebar of the application shows the 'DMSS' logo and a navigation menu with the following items: Dashboard, Profile, Create Disaster Project, and View Project. The bottom of the browser window shows the Windows taskbar with various application icons and the system clock displaying 6:41 PM on 7/5/2011.

Figure 6 Create New Project

B.2.4 Contingency Plan (s)

After creating a new project for a disaster, the MO is directed to a new page where he/she can make a contingency plan to be followed to perform activities related to a disaster. The options can be selected from a predefined list or can add tasks by it. This page is shown in figure 7.

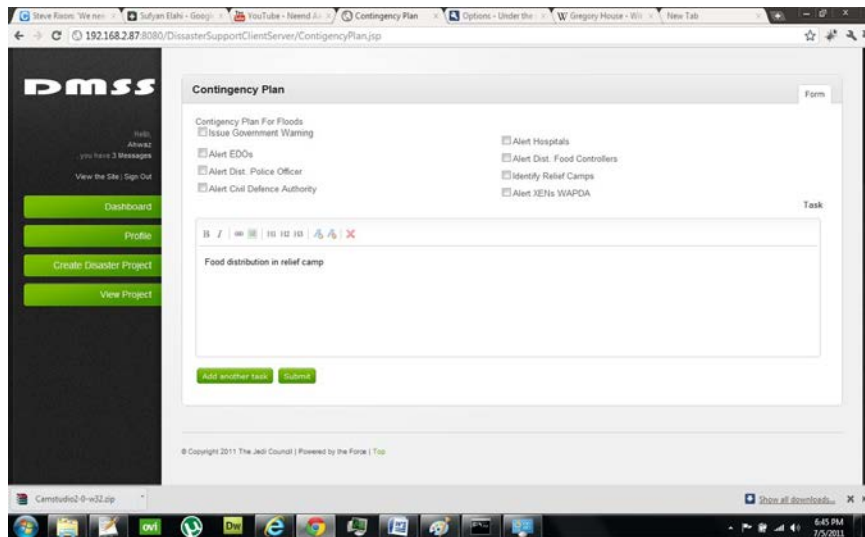
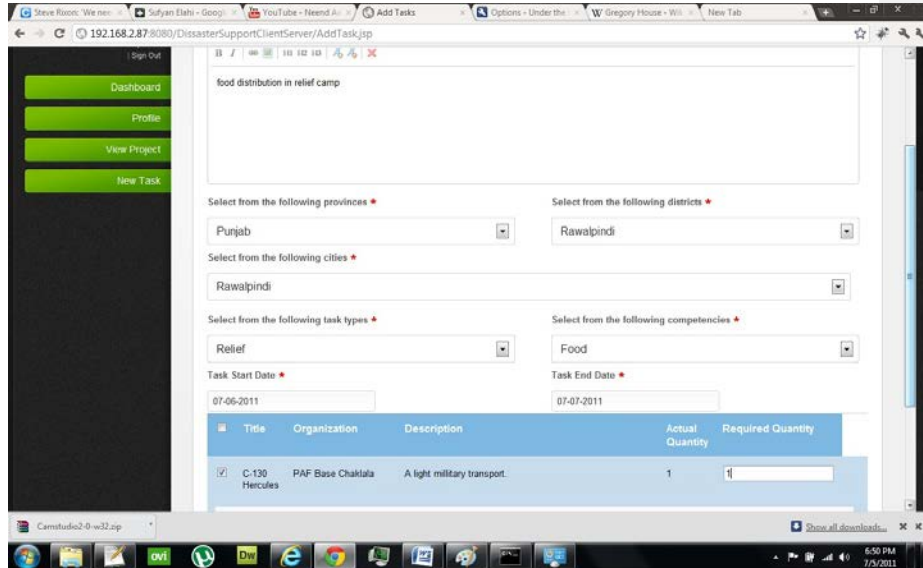


Figure 7 Contingency Plan

B.2.5 Add Task(s) Detail

The next navigation page is for adding details specific to a task(s). The fields with asterisk must be filled. After entering data press submit to complete action.



The screenshot displays a web browser window with the URL `192.168.2.87:8080/DisasterSupportClientServer/AddTask.jsp`. The page title is "Add Task(s) Detail". The main content area contains a form with the following fields:

- Text area: "food distribution in relief camp"
- Dropdown: "Select from the following provinces" (Punjab)
- Dropdown: "Select from the following districts" (Rawalpindi)
- Dropdown: "Select from the following cities" (Rawalpindi)
- Dropdown: "Select from the following task types" (Relief)
- Dropdown: "Select from the following competencies" (Food)
- Date field: "Task Start Date" (07-06-2011)
- Date field: "Task End Date" (07-07-2011)

Below the form is a table with the following data:

Title	Organization	Description	Actual Quantity	Required Quantity
<input checked="" type="checkbox"/> C-130 Hercules	PAF Base Chaklala	A light military transport.	1	<input type="text"/>

Figure 8 Add Task(s) Details

B.2.6 Add Resources to Tasks

Once MO has fed all details about all the tasks listed in contingency plan the next step is to add resources to those tasks. This webpage is shown in Figure 9. This is the last step of creating a task.

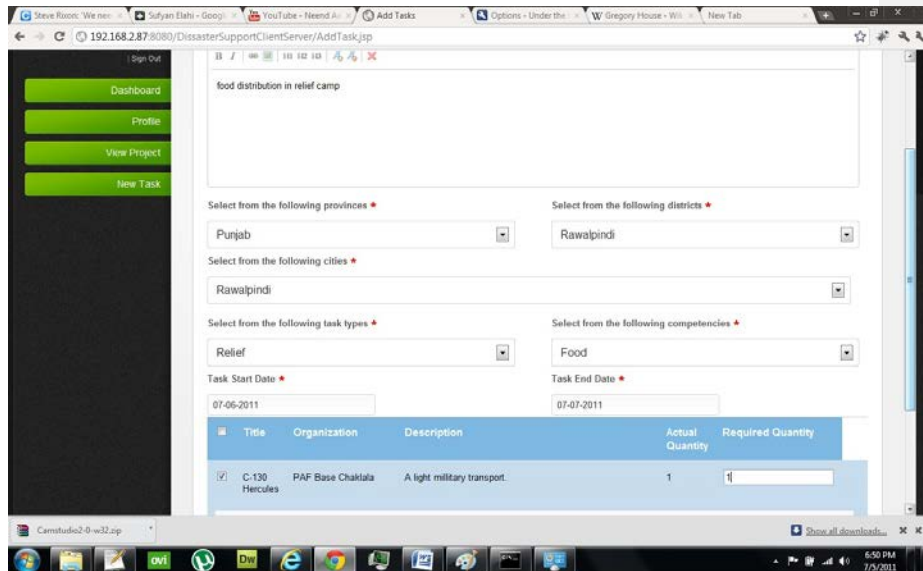


Figure 9 Add Tasks

An MO after creating a task can see different projects and their respective tasks and resources assigned to those tasks. To close a project he/she first has to free resources from tasks and close all tasks to close a project. Also an MO has option for status check to check status of different tasks and a message board for collaboration between different organizations.

B.3 Navigation Path for Organization

This sections shows navigation paths and different functionalities provided to an organization user type. The main dashboard is almost similar to

admin and MO while the different navigation links available are described here

B.3.1 Current Projects

The Figure 10 shows the current project page visible to an organization , to help them see what project they are engaged on currently.

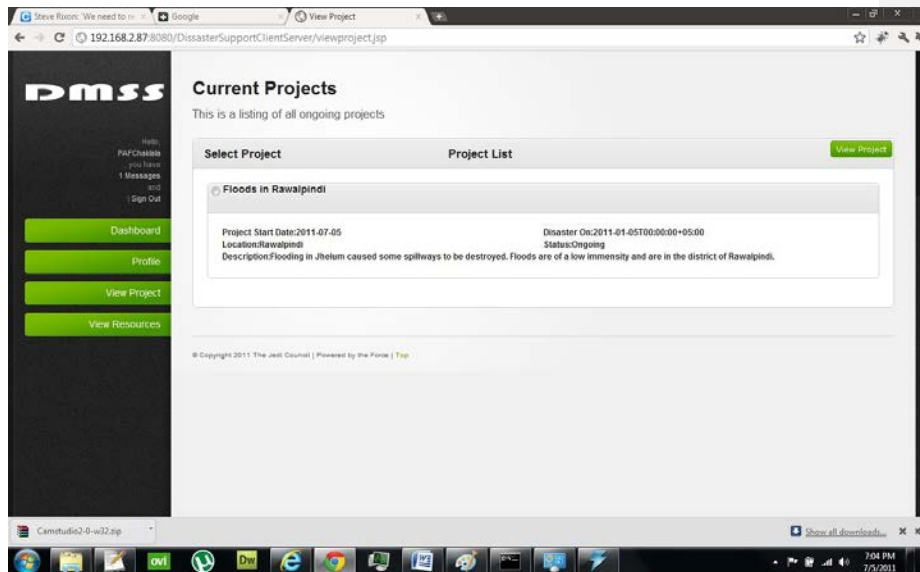


Figure 10 Current Projects for Organization

B.3.2 Add Resource

This web page adds resource shown in Figure 11 is used to update resources for an organization. The fields with asterisk must be filled. After entering data press submit to complete action.

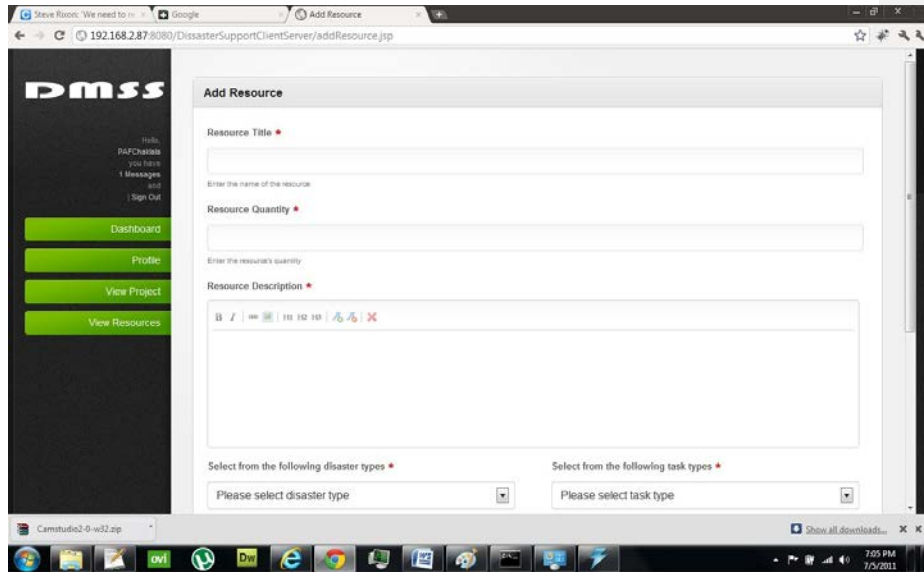


Figure 11 Add Resource

The other navigation links for organization helps in organization to see projects and tasks it's engaged on. Resource list showing occupied and free resources.

References

[1]W3C (2011), "Web Service architecture", available from :
<http://www.w3.org/TR/ws-arch/#whatis> [Accessed: 05-01-2011]

[2]Wikipedia (2011), "SAHANA FOSS" available from:
http://en.wikipedia.org/wiki/Sahana_FOSS_Disaster_Management_System
m [Accessed: 07-01-2011]

[3] Secure-irfc (2011), "Disaster Management Information System"
Available from: www.secure-irfc.org/DMIS [Accessed: 07-01-2011]

[4] Wikipedia (2011), "Java Script Object Notation" available from:
<http://en.wikipedia.org/wiki/JSON> [Accessed: 10-02-2011]

[5] Wikipedia (2011), "Service Oriented Architecture" available from:
http://en.wikipedia.org/wiki/Service_oriented_architecture [Accessed: 04-01-2011]

[6] W3C (2011), "Web Service" available from:
[www.w3.org/TR/ws-arch\](http://www.w3.org/TR/ws-arch/) [Accessed: 04-01-2011]

[7] W3C (2011), "Web Services Directory Language" available from:
[www.w3.org/TR/ws-arch\](http://www.w3.org/TR/ws-arch/) [Accessed: 13-02-2011]

[8] Wikipedia (2011), "SOAP Message" available from:
<http://en.wikipedia.org/wiki/SOAP> [Accessed: 16-02-2011]

[9] Wikipedia (2011), "GET Message" available from:
<http://en.wikipedia.org/wiki/SOAP> [Accessed: 16-02-2011]

[10] Wikipedia (2011), "POST Message" available from:
<http://en.wikipedia.org/wiki/POST> [Accessed: 16-02-2011]

[11] W3C (2011), "XML Binding" available from:
<http://www.w3.org/TR/xml> [Accessed: 20-02-2011]

[12] Wikipedia (2011), "jQuery" available from:
<http://en.wikipedia.org/wiki/JQuery> [Accessed: 12-02-2011]

[13] Wikipedia (2011), "JAX-WS" available from:
http://en.wikipedia.org/wiki/Java_Architecture_for_XML_Web_Services
[Accessed: 18-02-2011]

[14] Wikipedia (2011), "JAX-B" available from:
http://en.wikipedia.org/wiki/Java_Architecture_for_XML_Binding
[Accessed: 18-02-2011]

[15] Wikipedia (2011), "AJAX" available from:
[http://en.wikipedia.org/wiki/AJAX_\(Programming\)](http://en.wikipedia.org/wiki/AJAX_(Programming)) [Accessed: 22-02-2011]

[16] Wikipedia (2011), "RESTful Web Services" available from:
http://en.wikipedia.org/wiki/Representational_State_Transfer [Accessed:
10-02-2011]