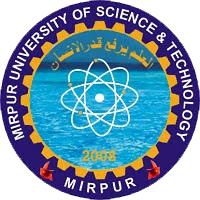
**MUST SOCIAL NETWORK**



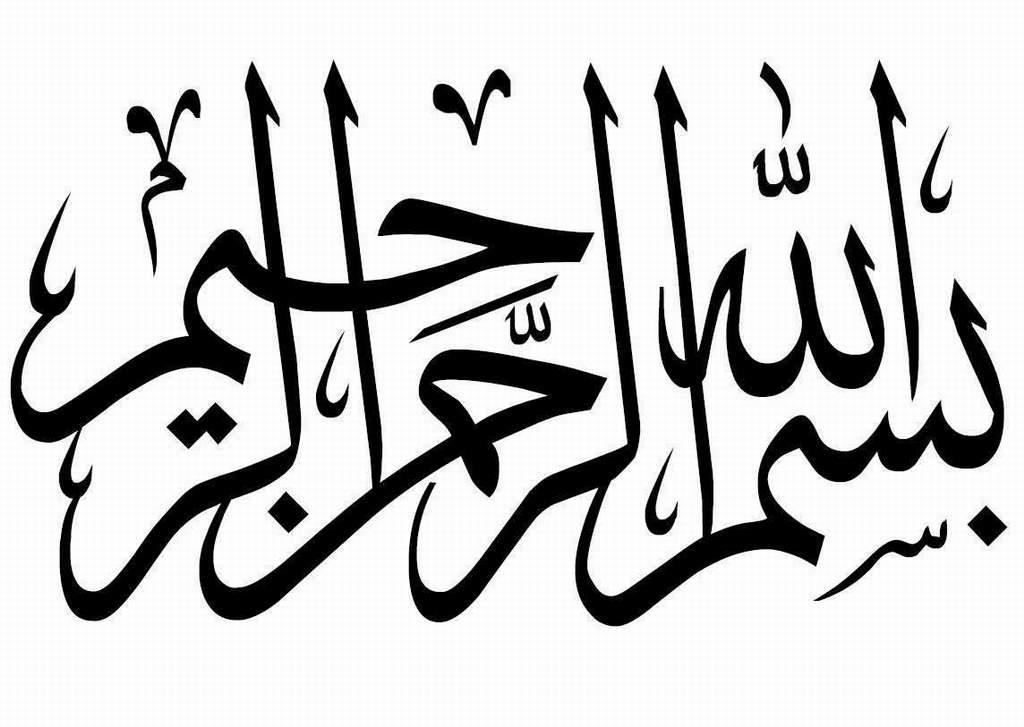
|  |  |  |
| --- | --- | --- |
| **SUBMITTED BY:** | **Name** | **Registration Number** |
|  | Muhammad Ubaid Raza | 13-SE-59 |
|  | Karamat Hussain | 13-SE-64 |
|  | M Junaid Shabbir | 13-SE-54 |

***SUPERVISOR:*** Engr. Samiullah Khan

**Department of Software Engineering**

**Mirpur University of Science and Technology (MUST)**

**A project report submitted in partial fulfillment of the requirements for the B.Sc. Software Engineering**



**CERTIFICATION**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It is certify that the following students have completed their project on

“**MUST Social Network**”

Submitted By:

**Muhammad Ubaid Raza 13-SE-59**

**Karamat Hussain 13-SE-64**

**M Junaid Shabbir 13-SE-54**

**Session: 2013-17**

This report is submitted in partial fulfillment of the requirements of the award of degree of **Bachelors of Software Engineering**. Mirpur University of Science and Technology (MUST)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**External Examiner Project Supervisor**

(Name of Supervisor)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Chairman**

Department of Software Engineering

MUST Mirpur (A.K)

# UNDERTAKING

I certify that research work titled “*MUST social Network*” is our own work. The work has not, in whole or in part, been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged/ referred.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Muhammad Ubaid Raza**

Registration No.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Karamat Hussain**

Registration No.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**M Junaid Shabbir**

Registration No.

**DECLARATION**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

We declare that this system, neither as a whole nor as a part has been copied from any other source. It is further declared that we have completed our project entirely on the basis of our personal effort made under the sincere guidance of our teachers. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning. If any part of this project and write up is proved to be copied out or there is any duplication of code then will be responsible for the consequences.

**Muhammad Ubaid Raza**

**Karamat Hussain**

**M Junaid Shabbir**

**ABSTRACT**

Our aim is to develop an **"MUST Social Network"** which is easily accessible to the teacher, students, staff and administration as well. This system regulate the activities between students and teachers and provide a single platform in order to communicate with each other. Students can communicate with teachers that are available in this network and ask questions about their problems.

**ACKNOWLEDGEMENT**

All the praise and glory to Allah, Who blessed us with the courage and knowledge to achieve our goal. We can never thank Him enough for His countless blessings upon us. Praise to Prophet Mohammad (S.A.W), who is and will always be a source of guidance and knowledge for humanity as a whole.

For us, a mile stone of this nature would never have been possible to achieve without the support of galaxy of some truly loving and kind people in our life. No words can fully describe our feelings of respect and gratitude for our affectionate parents, supporting sibling and friends, whose love, encouragement and prayers invariably buoyed us up. Their concern, love and support can never be paid back.

We owe a lot of sincere gratitude to our respected supervisor Mr. Samiullah Khan, whose true guidance, positive criticism and sincere encouragement made us to get to our destination. He became a source of inspiration for us and kept us moving in the right direction towards our goal.

**PREFACE**

This report presents a description of analysis, design, implementation and testing of the "MUST Social Network" developed for the Mirpur University of Sciences and Technology Mirpur. These chapters include the steps involved in the study to development a brief description of these chapters is given below:

**Chapter 1:** This chapter contains the introduction of the concerned organization and project.

**Chapter 2:** This chapter elaborates the feasibility study.

**Chapter 3:** This chapter contain the description of requirement specification.

**Chapter 4:** This chapter is about system design.

**Chapter 5:** This chapter is about implementation of the software.

**Chapter 6:** This chapter describe the software testing.

**Chapter 7:** This chapter is about conclusions, future improvements and system limitations.

**Chapter 8:** This chapter is about references.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **Table Of Contents** |  |

[1. **Introduction**](#_Toc459808776)

[1.1 Problems in Existing System: 2](#_Toc459808777)

[1.2 Solution of these Problem: 2](#_Toc459808778)

[1.3 System Conception 2](#_Toc459808780)

[1.3.1 Who is the application for? 3](#_Toc459808781)

[1.3.2 What problem will it solve? 3](#_Toc459808782)

[1.3.3 Where it will be used? 3](#_Toc459808783)

[1.3.4 When it is needed? 3](#_Toc459808784)

[1.3.5 Why it is needed? 3](#_Toc459808786)

[1.4 Advantages for Students 3](#_Toc459808787)

[1.5 Advantages for Faculty 4](#_Toc459808788)

[1.7 Software Process Model 4](#_Toc459808789)

[1.8 Project Boundaries 5](#_Toc459808790)

[1.9 Scope of Document 5](#_Toc459808791)

[1.10 Software Used 5](#_Toc459808792)

[1.11 Deployment 5](#_Toc459808793)

[1.12 Hardware Specification 5](#_Toc459808794)

[2. FEASIBILITY STUDY 6](#_Toc459808795)

[2.1 User Needs: 7](#_Toc459808796)

[2.2 How we fulfill Needs: 7](#_Toc459808797)

[2.3 Project Feasibility: 7](#_Toc459808798)

[2.3.1 Technical Feasibility 7](#_Toc459808800)

[2.3.2 Technical Feasibility-Hardware 7](#_Toc459808801)

[2.3.3 Technical Feasibility-Software 7](#_Toc459808802)

[2.3.4 Schedule Feasibility 7](#_Toc459808803)

[2.3.5 Economic Feasibility 8](#_Toc459808804)

[2.4 Social and Ethical Considerations: 8](#_Toc459808805)

[2.5 Project Planning: 8](#_Toc459808807)

[2.5.1 Work Breakdown Structure of Project: 8](#_Toc459808809)

[2.6 Gant Chart 9](#_Toc459808821)

[3. REQUIREMENT SPECIFICATION 10](#_Toc459808822)

[3.1 Purpose 11](#_Toc459808824)

[3.3 Objective: 11](#_Toc459808826)

[3.4 Document Conventions 11](#_Toc459808827)

[3.4.1 Intended Audience and Reading Suggestions 11](#_Toc459808829)

[3.4.2 Product Scope 11](#_Toc459808831)

[3.4.3 References 12](#_Toc459808833)

[3.5 Overall Description 12](#_Toc459808834)

[3.5.1 Product Perspective 12](#_Toc459808835)

[3.5.2 Product Functions 12](#_Toc459808837)

[3.5.3 User Classes and Characteristics 12](#_Toc459808838)

[3.5.4 Operating Environment 13](#_Toc459808839)

[3.6 Design and Implementation Constraints 13](#_Toc459808840)

[3.7 User Documentation 13](#_Toc459808841)

[3.8 Assumptions and Dependencies 14](#_Toc459808842)

[3.9 External Interface Requirements 14](#_Toc459808844)

[3.9.1 User Interfaces 14](#_Toc459808845)

[3.9.2 Hardware Interfaces 14](#_Toc459808847)

[3.9.3 Software Interface: 14](#_Toc459808848)

[3.9.4 Communications Interfaces 15](#_Toc459808850)

[3.9.5 System Features 15](#_Toc459808852)

3.9.5.[1 Event alert 15](#_Toc459808853)

3.9.5.[2 Event gallery 15](#_Toc459808854)

[3.9.5.3 Post status 16](#_Toc459808855)

3.9.5.4 [Online chat 16](#_Toc459808856)

3.9.[5.5 Security Requirements 16](#_Toc459808857)

3.9.5.[6 Software Quality Attributes 16](#_Toc459808858)

[3.10 Other Nonfunctional Requirements 17](#_Toc459808859)

[3.10.1 Performance Requirements 17](#_Toc459808860)

[3.10.2 Safety Requirements 17](#_Toc459808861)

[3.10.3 Other Requirements 17](#_Toc459808862)

[4. SYSTEM DESIGN 18](#_Toc459808863)

[4.1 Design Approach: 19](#_Toc459808865)

[4.2 Database Design: 19](#_Toc459808867)

[4.3 Object Modeling Technique (OMT): 19](#_Toc459808869)

[4.3.1 Class Model 19](#_Toc459808871)

[4.3.2 Interaction Model 19](#_Toc459808872)

[4.4. Classes and their Attributes: 20](#_Toc459808873)

[4.5 Class Diagram 21](#_Toc459808874)

[4.6 Logical Design 23](#_Toc459808876)

[4.6.1 Relation 23](#_Toc459808878)

[4.6.2 Associating Tables 23](#_Toc459808880)

[4.7 Physical Design 23](#_Toc459808882)

4.8 Data types and Variable Lengths……………………………………………………….39

[4.9 Database Diagram 27](#_Toc459808884)

[4.11 Sequence Diagrams 36](#_Toc459808885)

[4.12 Activity Diagram 42](#_Toc459808887)

[5. IMPLEMENTATION 50](#_Toc459808888)

[5.1 Member Login and sign-up: 51](#_Toc459813175)

[5.2 Member.php 54](#_Toc459813176)

[5.3 Members panel: 56](#_Toc459813177)

[5.3.1 Coding of index page: 56](#_Toc459813178)

[5.4 My Profile: 63](#_Toc459813179)

[5.5 Event Gallery: 65](#_Toc459813180)

[5.6 Event Alert: 66](#_Toc459813181)

[6. SOFTWARE TESTING 69](#_Toc459813182)

[6.1 Psychology of Testing 70](#_Toc459813183)

[6.2 Testing Objective 70](#_Toc459813184)

[6.3 The Box Approach 70](#_Toc459813185)

[6.3.1 White Box testing 70](#_Toc459813186)

[6.3.2 Black Box testing 71](#_Toc459813187)

[6.4 Levels of Testing 71](#_Toc459813188)

[6.5 Unit Testing 71](#_Toc459813189)

[6.6 Integration Testing 72](#_Toc459813190)

[6.7 System Testing 72](#_Toc459813191)

[6.8 Alpha Testing 72](#_Toc459813192)

[6.9 Beta Testing 72](#_Toc459813193)

[6.13 Test Cases 73](#_Toc459813194)

[6.13.1 Test Case 1: Sign-up 73](#_Toc459813195)

[6.13.2 Test Case 2: Login 74](#_Toc459813196)

[6.13.3 Test Case 3: Create Conversation 74](#_Toc459813197)

[6.13.4 Test Case 4: View Conversation 75](#_Toc459813198)

[6.13.6 Test Case 6: Check Conversation 76](#_Toc459813199)

[6.13.7 Test Case 7: Add Member 76](#_Toc459813200)

[6.13.8 Test Case 8: Ban Member 77](#_Toc459813201)

[6.13.9 Test Case 9: Show Member 77](#_Toc459813202)

[6.13.10 Test Case 10: Status Check 78](#_Toc459813203)

[6.13.11 Test Case 11: view Profile 78](#_Toc459813204)

[6.13.12 Test Case 12: Update Profile 79](#_Toc459813205)

[6.13.13 Test Case 13: Delete Member 79](#_Toc459813206)

[6.13.14 Test Case 14: View Status 80](#_Toc459813207)

[6.13.15 Test Case 15: Delete Status 80](#_Toc459813208)

[6.13.16 Test Case 16: Update Status 81](#_Toc459813209)

[6.13.17 Test Case 17: Insert Status 81](#_Toc459813210)

[6.13.18 Test Case 18: Insert Profile Picture 82](#_Toc459813211)

[6.13.19 Test Case 19: Show Profile Picture 82](#_Toc459813212)

[6.13.20 Test Case 20: Change Profile Picture 83](#_Toc459813213)

[6.13.21 Test Case 21: Delete Profile Picture 83](#_Toc459813214)

[6.13.22 Test Case 22: Insert PhotoIn Event Gallery 84](#_Toc459813215)

[6.13.23 Test Case 23: Delete PhotoFrom Album 84](#_Toc459813216)

[6.13.24 Test Case 24: Show Album Photo 85](#_Toc459813217)

[6.13.25 Test Case 25: Slider 85](#_Toc459813218)

[6.13.26 Test Case 26: Add Event 86](#_Toc459813219)

[6.13.27 Test Case 27: Edit Event 86](#_Toc459813220)

[6.13.28 Test Case 28: Delete Event 87](#_Toc459813221)

[6.13.29 Test Case 29: Show Event 87](#_Toc459813222)

[6.13.30 Test Case 30: Like 88](#_Toc459813223)

[6.13.31 Test Case 31: Unlike 88](#_Toc459813224)

[6.13.32 Test Case 32: Like Show 89](#_Toc459813225)

[6.13.33 Test Case 33: Like Count 89](#_Toc459813226)

[6.13.34 Test Case 34: Insert Message 90](#_Toc459813227)

[6.13.35 Test Case 35: Delete Message 90](#_Toc459813228)

[6.13.36 Test Case 36: Show Message 91](#_Toc459813229)

[6.13.37 Test Case 37: Check Message Status 91](#_Toc459813230)

[6.13.38 Test Case 38: Add Comment 92](#_Toc459813231)

[6.13.39 Test Case 39: Edit Comment 92](#_Toc459813232)

[6.13.40 Test Case 40: Delete Comment 93](#_Toc459813233)

[6.13.41 Test Case 41: Show Comment 93](#_Toc459813234)

[7. CONCULISION & FUTURE Work 94](#_Toc459813235)

[7.1 Conclusion 95](#_Toc459813236)

[7.2 Future Improvements 96](#_Toc459813237)

[7.3 Limitations 96](#_Toc459813238)

[8 REFERENCES 97](#_Toc459813240)

**Chapter 1**

# Introduction

We want to make a project that will help the university staff to alert the students about any event that is going to happen. We are going to introduce a social network website which will gather the university staff and students on a platform. On that platform students are informed about any event that is going to happen in the university. This will notify the students about events on MUST social network as well as they will receive text SMS on mobile phone without having internet connection. In this website we will also provide the pictures of events in events gallery.

### 1.1 Problems in Existing System:

Currently there is no social network of MUST. There is no organized photo collection. Students face different problems. For example if a student wants to communicate with a teacher then there is no effective way of communication. Notice board concept is too old. It is often ignored by students. If any announcement is make then many students did not aware of it. There is no discussion portal for the MUST students.

**1.2 Solution of these Problem:**

### Solution of these problems can be overcome in the following way:

#### 1.2.1 Proposed System & Its Modules:

The aim of the MUST Social Network is to provide communication among teacher, student and staff. The system after careful analysis has been identified to be presented with the following modules:

* **Event alert:** Admin and staff can generate event alert. The users can receive an event alert on their mobile phone through an SMS.
* **Post Status:** Everyone can post status.Users are also able to comment and like on the status.
* **Event Gallery:** Events photos will be uploaded in the network website. Everyone can view photo and play slides**.**
* **Online chat:** Every member can chat with other members using chat feature. Last seen and message seen features will be provided.

### 

### 1.3 System Conception

Most of the system start as vague ideas that need more substance. A good system concept must answer the following questions as our system have:

### 

### 1.3.1 Who is the application for?

This network is for MUST University. It is for education purpose. This is the social network for the MUST which will provide basic feature to its members. There are some unique features for example event alert and event gallery. Which are not present in other social networks. MUST will be the first university which will have its social network. This network can be used by students, teachers and staff of the university.

### 

### 1.3.2 What problem will it solve?

Our system overcome the different kinds of problems such as

1. Old notice board concept
2. Albums for the events
3. Deficiency of communication
4. Discussion portal

### 

### 1.3.3 Where it will be used?

Our system will be used in Mirpur University of Sciences & Technology.

### 1.3.4 When it is needed?

### It is needed to minimize the investment and maximize the revenue in economic sense. Our system remove the old notice board concept. It will provide effective communication among MUST members.

### 1.3.5 Why it is needed?

Our system is according to modern concept. Students will be notified by website as well as by an SMS. Chat is provided among members so student can ask question from teachers. Teacher can make announcement about quiz and assignment.

## 

## **1.4 Advantages for Students**

Advantages for the students are explained below

1. **On time alert:** Student will be alerted on time. Students can receive alert about an event in few seconds.
2. **Gallery:** If anyone miss an event then he can view photos of event.
3. **Chat:** Students can chat with teachers and can discuss their problems.
4. **Discussion:** Student can discuss any problem using post feature. On which every member can comment.

## **Advantages for Faculty**

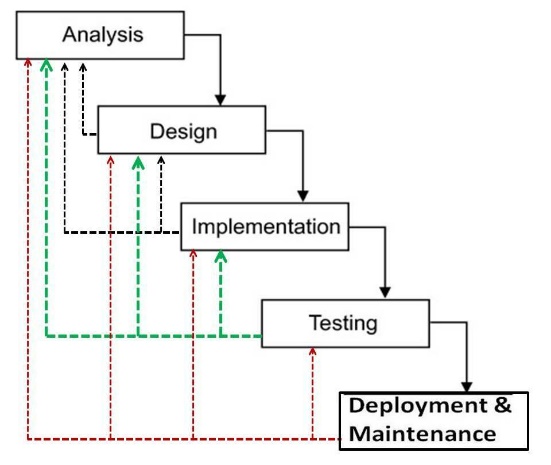
1. **Alert:** Faculty can alert student about an event in one click during holiday
2. **Easy to contact:** Faculty can contact any member of university easily.
3. **Project Need:** Communication between university teachers and students.

**1.6 Proposed Project’s Need**

1. To remove students problems which they are facing due to absence of an effective network.
2. To provide alerts to students in time.

## **1.7 Software Process Model**

Software process model is an abstract representation of a software process, which is a set of activities that leads to the production of a software product. Some of the generic process models are waterfall model, evolutionary development and component based software engineering. For this project we will follow iterative waterfall process model, because of clear and unchanging business requirements.



**Figure 1: Iterative Waterfall Model**

## 

## **1.8 Project Boundaries**

The overall project scope is within our university, but it can be used in any college, school and university.

## **1.9 Scope of Document**

This document will theoretically describe how the application is designed, implemented, and the scope of the application. This documentation can be used by teacher and students for study purpose.

## **1.10 Software Used**

We will use following software tools to implement this project:

1. **Database Server:** My-SQL
2. **Client:** Microsoft Internet Explorer or any web browser
3. **Development Tools:** Dreamweaver
4. **Programming Language:** PHP, AJAX, JQUERY, JAVASRCIPT

## 

## **1.11 Deployment**

1. **Operating System Server:** Window 8.1

## 

## **1.12 Hardware Specification**

The following hardware tools will be required to implement this project:

1. **Processor:** Intel Core i3
2. **RAM:** 4GB
3. **Hard Disk:** 260 GB

**Chapter 2**

# FEASIBILITY STUDY

This document contain the feasibility of the proposed project and describes in details the schedule plan we have opted to develop current project.

## 

## **2.1 User Needs:**

User wants the fully automated system with all the requirements mentioned in the scope of this project. He wants the application which can give optimal results, with improved performance and less development cost.

## 

## **2.2 How we fulfill Needs:**

User needs will be fulfilled by using up to marks techniques. Moreover, the application will be developed within the timeframe and cost, estimated before the project development. We will develop centralized application which is easy to maintain and which can perform efficiently as compared to the manual computation of user’s project.

## 

## **2.3 Project Feasibility:**

## This section define that whether the project is worth solving or not.

### 2.3.1 Technical Feasibility

This project does not need such hardware for development, design and continued operation that is to be bought, but only the hardware that both the developers and the organization use in daily life like laptop. And so the project is technically feasible with respect to hardware.

### 2.3.2 Technical Feasibility-Hardware

No extra hardware required, Pentium 4 or higher required which is easily available in market.

### 2.3.3 Technical Feasibility-Software

* Window8
* Dream Weaver
* My-Sql

### 2.3.4 Schedule Feasibility

The proposed project is medium level but its modules are time consuming, our estimate time after paper works is about 4 months, this project will complete in four months (120 Days) with its full specifications. With hard work and full dedication, we will be able to deliver the system on time.

### 

### 2.3.5 Economic Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available. There is nominal expenditure and economic feasibility for certain.

## 

## **2.4 Social and Ethical Considerations:**

## The under described project does not contradict to any of the social or ethnic value. Our project does not affect the social and moral values of our society. It does not harm the feelings of other in fact, it is social network that provide a single platform to communicate with each other in order to share problems with teacher. Our network is limited to Mirpur University of Sciences and Technology so, this network does not violate ethics and does not perform any unethical activity that can harm others.

## **2.5 Project Planning:**

## Project planning is listed below

### 2.5.1 Work Breakdown Structure of Project:

### Project Definition

### Project Approval

### System Conception

### System Analysis

### Domain Model

### Application Model

### System Design

### Class Design

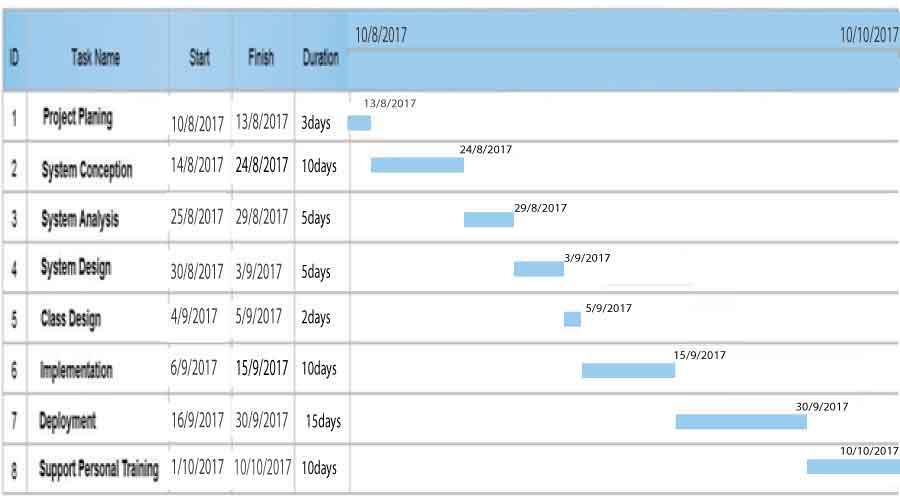
### Implementation

### Deployment

### Support Personal Training

## 

## **2.6 Gant Chart**



**Figure 2: Gant Chart**

**Chapter 3**

# REQUIREMENT SPECIFICATION

### This document, Software Requirements Specification (SRS), describes the overall requirements of the project. It states the scope of the project and details its functional and non-functional requirements

### 3.1 Purpose

### MUST social network will help the university staff to alert the students about every event. We are going to introduce a social network website which will gather the university staff and students on a single platform. Students are informed about every event that is going to happen in the university. Other features like online chatting, post status and event gallery are also provided to everyone who is connected with this network.

**3.2 Scope**

The scope MUST social network includes:

1. Students and teachers of MUST can be connected with this network anywhere in the world using internet.
2. Text messaging service is easily available.
3. All events photos are easily available in the event gallery.
4. Online chat between students and teachers can be done easily.
5. Efficient and time saving portal of MUST.

**3.3 Objective:**

The objectives of this social network are to improve communication between teachers, students and faculty. This network will allow members to contact each other. Post status to discuss any problems. SMS alert will enable every member to receive SMS while online or offline.

### 3.4 Document Conventions

### This document covers the conventions as described by IEEE SRS Template.

## **3.4.1 Intended Audience and Reading Suggestions**

## Intended Audience of this document includes project members, project supervisor and target clients.

## **3.4.2 Product Scope**

## The scope MUST social network includes:

1. Students and teachers of MUST can be connected with this network anywhere in the world using internet.
2. Text messaging service is easily available.
3. All events photos are easily available in the event gallery.
4. Online chat between students and teachers can be done easily.
5. Efficient and time saving portal of MUST.

## **3.4.3 References**

* IEEE STD 830-1993, Recommended Practice for Software Requirements Specifications, December 2, 1993.

# 3.5 Overall Description

## **3.5.1 Product Perspective**

## Existing system is just based on manual noticeboard. A single noticeboard is not enough to alert all the university students. During holidays noticeboard is not a good option. The noticeboard concept is too old, students often ignore noticeboard and they didn’t get alert about any event. In MUST social network every member is notified about every event through website and text SMS. MUST has no discussion portal. This network will provide a discussion portal where every member can share his problem. There was no organized collection of events. This network provide event gallery that will contain events photos.

## **3.5.2 Product Functions**

1. Event alert through SMS
2. Online chat

* Online status
* Last seen
* Message seen

1. Post status

* Comment
* Like

1. Event gallery

* Photo Albums
* Slider

## **User Classes and Characteristics**

**i Administrator**

* Administrator of this network will have all the authorities.
* He can add or remove a member, status and photos.
* He can alert all the members through event alert service.
* He can manage chat.
* Chat with other members.
* Able to post status.
* Able to upload event photos.

1. **Staff**

* They can alert all the members through event alert service.
* Chat with other members.
* Able to post status.
* Able to upload event photos.

1. **Teacher**

* Chat with other members.
* Able to post status.
* Able to view event photos.

1. **Student**

* Chat with other members.
* Able to post status.
* Able to view event photos.

### 3.5.4 Operating Environment

It can be operate on server having hardware requirement.

* Processor: Intel Pentium IV
* RAM: 1GB
* Hard Disk: 20GB
* Operating System**:** Win XP, Win 7, Win 8, Win 10

### 3.6 Design and Implementation Constraints

The Product is developed using PHP, AJAX, JAVASCRIPT, JQUERY and HTML . The backend database for this My-SQL Server. The product is accomplished with login facility so that specific function is available to specific member.

### 3.7 User Documentation

The network will include user manual. The user manual will include network overview, complete configuration of the used software (such as MY-SQL), Technical details, backup

Procedure and contact information which will include email address. The databases will be created in the MY-SQL.

### 3.8 Assumptions and Dependencies

### The MUST social network needs the following third party product.

* MY-SQL to store data in database
* PHP
* SMS API

### 3.9 External Interface Requirements

### 3.9.1 User Interfaces

### The Graphical User Interface of MUST social network Provide following functionalities.

* Online chat
* View event gallery
* Post status
* Event alert
* View and change profile

### Hardware Interfaces

1. **Server Side:**

* **Operating System:** Windows XP, Windows 7,8 or 10
* **Processor:** Pentium IV or higher
* **RAM:** 1 GB
* **Hard Drive:** 40 GB or more

1. **Client side:**

* **Operating System:** Windows XP, Windows 7,8 or 10
* **Processor:** Pentium III or 2.0 GHz or higher.
* **RAM:** 256 Mb or more

### 3.9.3 Software Interface:

### We will use the following software and tools while developing the MUST social network:

* Dreamweaver and sublime text
* WampServer

1. **Tools required:**

* Apache server
* My SQL
* PHP my admin

1. **Languages required:**

* HTML
* CSS
* PHP
* SQL

## **3.9.4 Communications Interfaces**

## Following are the communication interfaces that are required for this social network:

1. Internet connection
2. Computer system
3. Mobile phone

## 

## **3.9.5 System Features**

Following are the features of MUST Social Network:

## **Event alert**

* **Description and Priority:** This function will provide benefit to both to the administration of campus and to the Student. When the event is placed then the every subscribed person will receive SMS.
* **Stimulus/Response Sequences:** The subscribed users will receive a text SMS having an information about that event or they can view event on the website.
* **Functional Requirements**: Member should subscribe the SMS alert service.

## **Event gallery**

* **Description and Priority:** It contains event photos, which are properly organized in the albums and sliders.
* **Stimulus/Response Sequences:** The member who want to access the event gallery should be a member of this network.
* **Functional Requirements:** The photos should be access able to members.

## **Post status**

* **Description and Priority:** Every member can post status.
* **Stimulus/Response Sequences:** Other member can view the posts and can comment and like on that status.

## **Online chat**

* **Description and Priority:** A member can send message to other member. He can view the status of receiver either he is offline or online.
* **Stimulus/Response Sequences:** When the receiver will receive the message then the sender will receive the notification of message seen.
* **Functional Requirements*:*** Both sender and receiver should be the member of this network*.*

## **Security Requirements**

* **Login:**Teachers and students of MUST can login to this network by using their registered username and password.
* **New user authentication:** New teachers and students are authenticated by their CNIC number that are already stored in the database of this network and then they will be added in the member list.
* **Administrator:** The administrator of this network can manage all the members of the network and he can block a member who perform illegal activities.

## **Software Quality Attributes**

* **Reliability:** The MUST data will be store in database so there will be no chance of duplication or redundancy of data. In case of any accidental lose the data will be recovered by server. So there is no chances of loss of data.
* **Maintenance:** The system will be developed in PHP object oriented. This network can be maintain easily because of using OOP approach. Addition of new function will not affect the performance of network.
* **Portability**: This website is access from any operating system having Web browser.

## **3.10** **Other Nonfunctional Requirements**

### Performance Requirements

* The user can send and receive the receive message in real time.
* The member should post status in real time.
* SMS should be send through efficient API.

### Safety Requirements

* The user should use secure network to prevent themselves from intrusions.
* Students cannot access the account of other students because of their unique username and password.

## 

## **3.10.3 Other Requirements**

The development and testing requirements are: -

* Wampserver
* Dreamweaver/Sublime text
* Google Chrome

**Chapter 4**

# SYSTEM DESIGN

## Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of a class that will later be built. Designing is the foundation of whole development process. If we obtain a perfect design of a system then an efficient and effective software can be developed on the other hand if there are some concepts we have not clear them the result will be opposite to previous case. A software with the bugs and failure will be achieved. So designing phase is really important in this phase we translate requirements into the software structure. This is one of the important step in development every functionality of program depends upon it.

## 

## **4.1 Design Approach:**

## Design approach for this software is conventional, that is, database model used is relational and on the other side Object modeling technique (OMT) is used.

## 

## **4.2 Database Design:**

## This system would have a database as its data store, so the design of the database is very important. We chose relational database model for our database. Database design includes conceptual, logical and physical design phases, which are discussed in this section.

## 

## **4.3 Object Modeling Technique (OMT):**

## OMT is object oriented design approach, here we used two fundamental modeling techniques which are class and interaction modeling.

### 4.3.1 Class Model

The class model shows static class objects in a system and the relationships between them. Two particularly important relationships are generalization (inheritance) and aggregation each class object on the diagram often shows the class name, its attributes and operations. Here we capture the real world concept that are important to our application.

### 4.3.2 Interaction Model

This model describe the interaction between the objects. Its diagrams are

* Use Case Diagram
* Sequence Diagram
* Activity Diagram

**4.4 Conceptual Design:** Conceptual design is about understanding the organization, the real world modeling; it captures the essence of the real world scenario. In this phase all the important classes of the system are extracted with their essential attributes and the relationship between them are established. This results in class diagram

**4.4.1 Association**

The relationship between the extracted tables are tabulated

**Figure 3: Association**

## **Classes and their Attributes:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class Name** | **Multiplicity** | **Relationship** | **Multiplicity** | **Class Name** |
| **Admin** | **1** | **Can send/receive** | **\*** | **message** |
| **Student** | **1** | **Can send/receive** | **\*** | **message** |
| **Staff** | **1** | **Can send/receive** | **\*** | **message** |
| **Teacher** | **1** | **Can send/receive** | **\*** | **message** |
| **Admin** | **1** | **Can post** | **\*** | **Status** |
| **Teacher** | **1** | **Can post** | **\*** | **Status** |
| **Staff** | **1** | **Can post** | **\*** | **Status** |
| **Student** | **1** | **Can post** | **\*** | **Status** |
| **Admin** | **1** | **Can like** | **\*** | **Like** |
| **Teacher** | **1** | **Can like** | **\*** | **Like** |
| **Staff** | **1** | **Can like** | **\*** | **Like** |
| **Student** | **1** | **Can like** | **\*** | **Like** |
| **Admin** | **1** | **Can comment** | **\*** | **Comment** |
| **Teacher** | **1** | **Can comment** | **\*** | **Comment** |
| **Student** | **1** | **Can comment** | **\*** | **Comment** |
| **Staff** | **1** | **Can comment** | **\*** | **Comment** |
| **Admin** | **1** | **Can generate** | **\*** | **eventAlert** |
| **Staff** | **1** | **Can generate** | **\*** | **eventAlert** |
| **Admin** | **1** | **Can upload** | **\*** | **eventGallery** |
| **Staff** | **1** | **Can upload** | **\*** | **eventGallery** |
| **Admin** | **1** | **Can send** | **\*** | **Message** |
| **Staff** | **1** | **Can send** | **\*** | **Message** |
| **Student** | **1** | **Can send** | **\*** | **Message** |
| **Teacher** | **1** | **Can send** | **\*** | **Message** |
| **Admin** | **1** | **Can block** | **\*** | **Members** |

The classes that are used in our social network and their attributes are identified. Their description is also discussed. Every class with its attribute are given in the following tables:

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Description** | **Attributes** |
| Profile pic | User profile picture | currentStatus, name, picHolderId,PicDate |
| Member | User of the network | Id, firstName, lastName, email, password, lastseen, cnic, city, dob, gender, position |
| Status | Status which a user wants to post | Id, status, posterName, date |
| Conversation | To check the conversation exist in two members | Id, firstPersonId, secondPersonId, date, del1 ,del2 |
| Event gallery | Consist of events photo | Id, eventName, photoName, date |
| Event Alert | Provide event alert through a post and sms | Id, event, date alertReciever |
| Likes | Provide option to like a status | Id, likerId, likerName, likes |
| Comment | Provide option to comment on a status | commenterId, commenterName, date, comment |
| Message | It consist of messages which are stored after starting conversation | Id, message, senderId, senderName, recieverId, status, date,del1, del2 |

**Figure 4: Classes and Attributes**

## **4.5 Class Diagram**

## After the identification of classes their attributes and the relationships among them our class diagram is given below



**Figure 5: Class diagram**

## **4.6 Logical Design**

## Logical database design is the process of transforming the conceptual data model into a logical data model. The logical model we will use is object data model. It represents data in the form of tables/relations.

### 4.6.1 Relation

### A relation is a named, two-dimensional table of data. Each relation (or table) consists of a set of named columns and an arbitrary number of unnamed rows. An attribute is a named column of a relation. Each row of a relation corresponds to a record that contains data (attribute) values for a single class.

* **Primary key** is an attribute or a combination of attributes that uniquely identifies each row in a Relation.
* **Foreign key** is an attribute (possibly composite) in a relation that serves as the primary key of another relation.

### 4.6.2 Associating Tables

### During logical design, we transform the class diagram that was developed during conceptual design into relational database schemas. The inputs to this process are the class diagram and the outputs are the relational schemas. Regular tables are classes that have an independent existence and generally represent real-world objects, such as persons and products. Each regular table type in a class diagram is transformed into a relation. The name given to the relation is generally the same as the table type. Each simple attribute of the table type becomes an attribute of the relation. The identifier of the table type becomes the primary key of the corresponding relation.

## **4.7 Physical Design**

## Physical database design produces the technical specifications that programmers, database Administrator, and others involved in information systems construction will use during the implementation phase. Physical database design requires critical decisions that will affect the performance of the application system like choosing the storage format (called data type) for each attribute from the logical data model. The format is chosen to minimize storage space.

**4.8 Data types and Variable Lengths**

Attributes along with their specific data type and length that are used in our social network are identified and given are as follows:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Length** |
| Id | Int | 30 |
| Name | Varchar | 30 |
| PicHolderId | Int | 30 |
| PicDate | Varchar | 30 |
| currentStatus | Varchar | 30 |

**Figure 6: Profile Pic**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Length** |
| Id | Int | 30 |
| FirstName | Varchar | 100 |
| LastName | Varchar | 100 |
| Email | Varchar | 100 |
| Password | Varchar | 30 |
| Lastseen | Varchar | 50 |
| Cnic | Int | 13 |
| Dob | Varchar | 30 |
| Gender | Varchar | 10 |
| Position | Varchar | 30 |
| Ban | Varchar | 2 |
| Department | Varchar | 30 |
| City | Varchar | 30 |
| Phone | Varchar | 13 |
| Security Question | Varchar | 30 |
| Answer | Varchar | 30 |

**Figure 7: Members**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Length** |
| Id | Int | 30 |
| Status | Varchar | 255 |
| statusPosterId | Int | 30 |
| posterName | Varchar | 30 |
| Date | Varchar | 30 |

**Figure 8: Status**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Length** |
| Id | Int | 30 |
| firstPersonId | Int | 30 |
| SecondPersonId | Int | 30 |
| Date | Varchar | 30 |
| Del1 | Int | 1 |
| Del2 | Int | 1 |

**Figure 9: Conversation**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Length** |
| Id | Int | 30 |
| EventName | Varchar | 255 |
| PhotoName | Varchar | 200 |
| PhotoUploaderId | Int | 30 |
| Date | Varchar | 30 |

**Figure 10: Event Gallery**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Length** |
| Id | Int | 30 |
| Event | Varchar | 255 |
| alertReciever | Varchar | 30 |
| alertGeneratorId | Int | 30 |
| Date | Varchar | 30 |

**Figure 11: Event Alert**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Length** |
| Id | Int | 30 |
| LikerId | Int | 30 |
| LikerName | Varchar | 30 |
| StatusId | Int | 30 |

**Figure 12: Status Like**

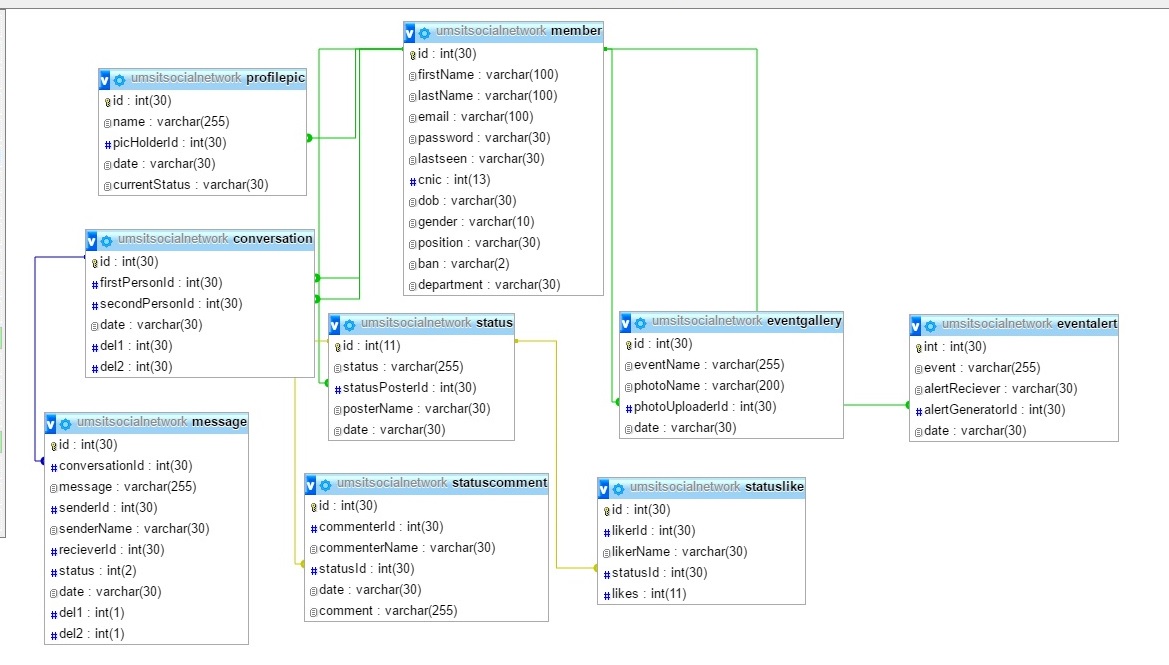
|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **length** |
| Id | Int | 30 |
| commenterId | Int | 30 |
| commenterName | Varchar | 30 |
| StatusId | Int | 30 |
| Like | Int | 1 |

**Figure 13: Status Comment**

|  |  |  |
| --- | --- | --- |
|  | **Datatype** | **Length** |
| Id | Int | 30 |
| Message | Varchar | 255 |
| SenderId | Int | 30 |
| senderName | Varchar | 30 |
| recieverId | Int | 30 |
| Status | Int | 2 |
| Date | Varchar | 30 |
| Del1 | Int | 1 |
| Del2 | Int | 2 |

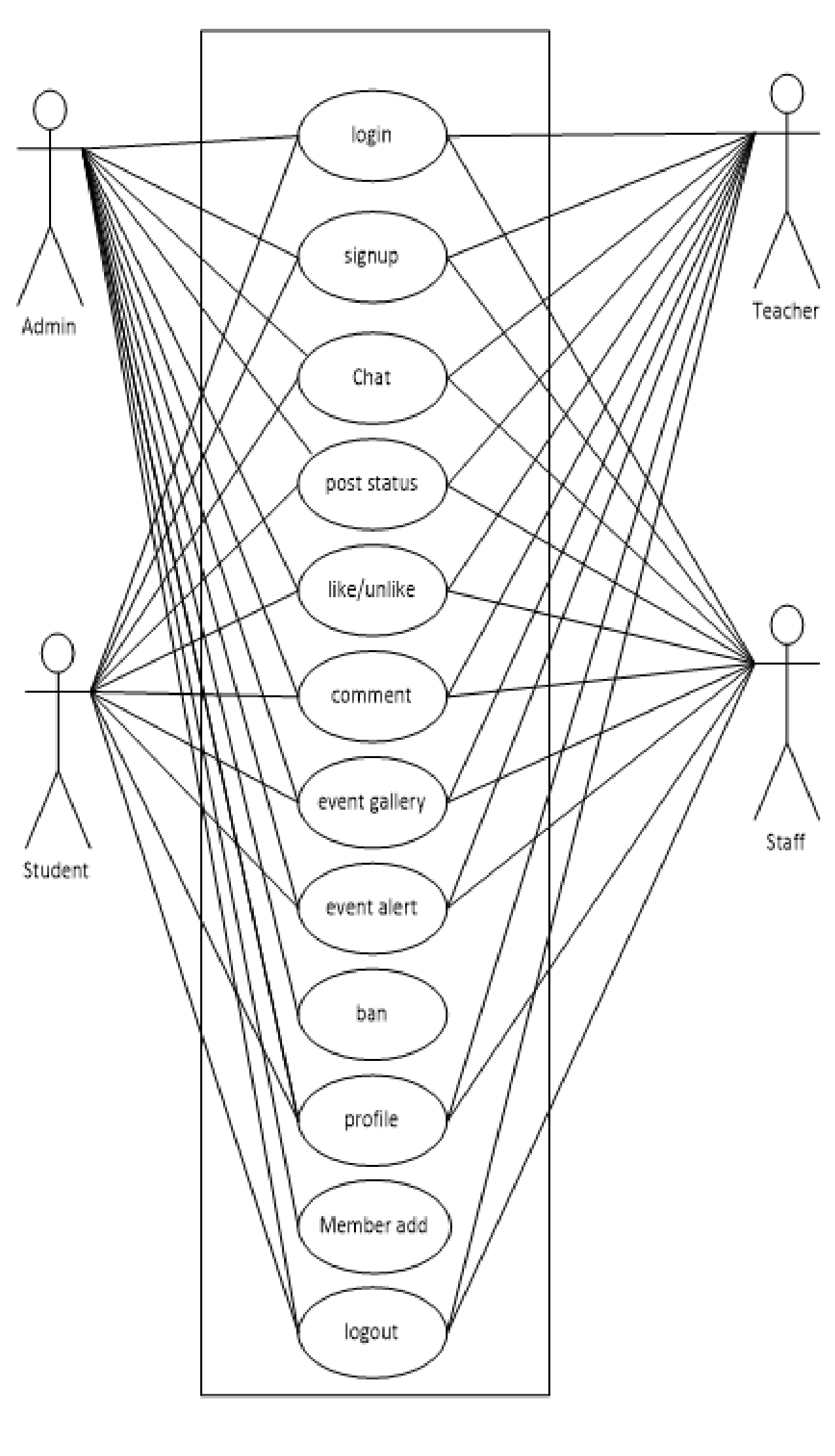
**Figure 14:** **Message**

### 4.9 Database Diagram

 **Figure 15: Database**

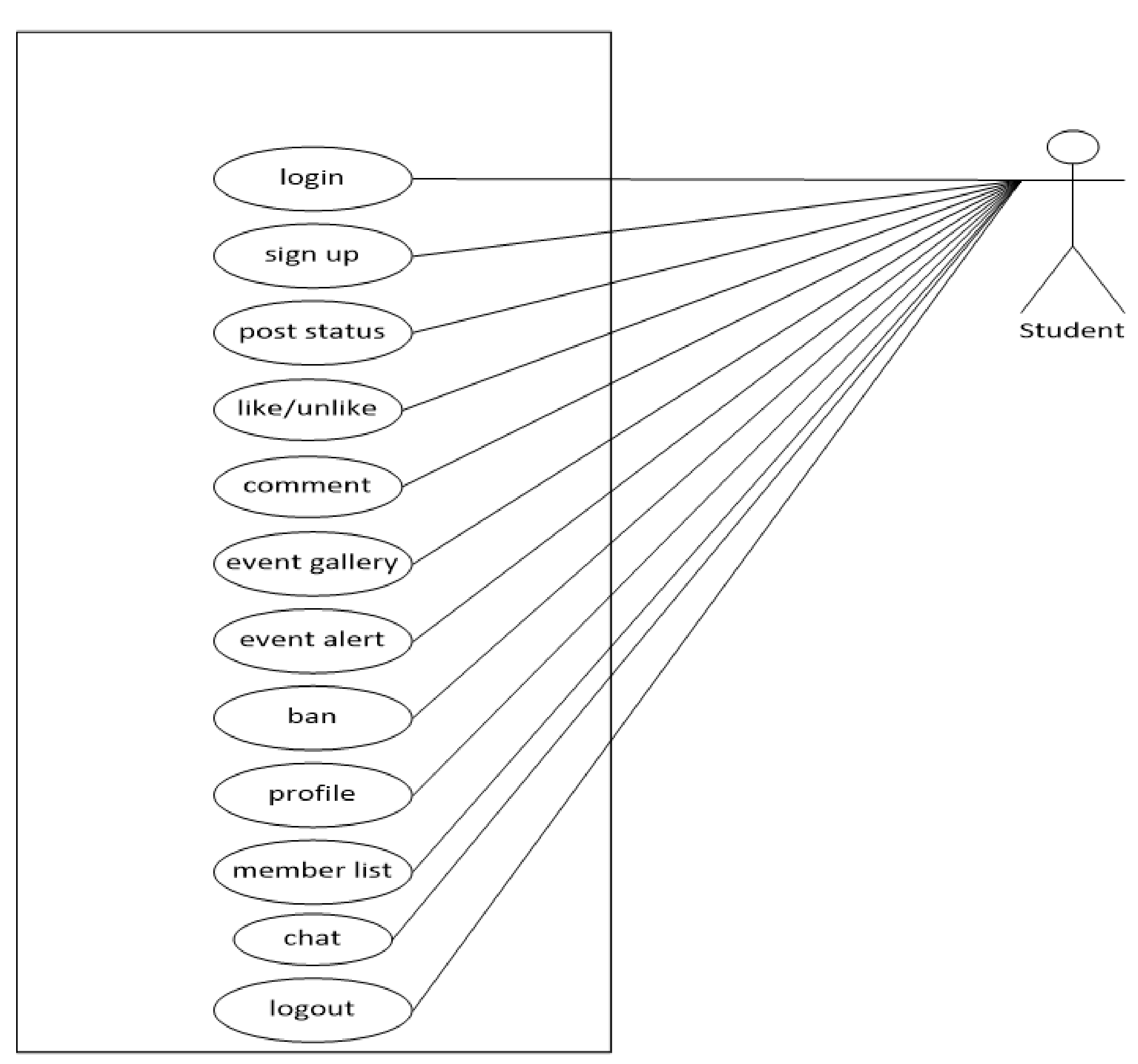
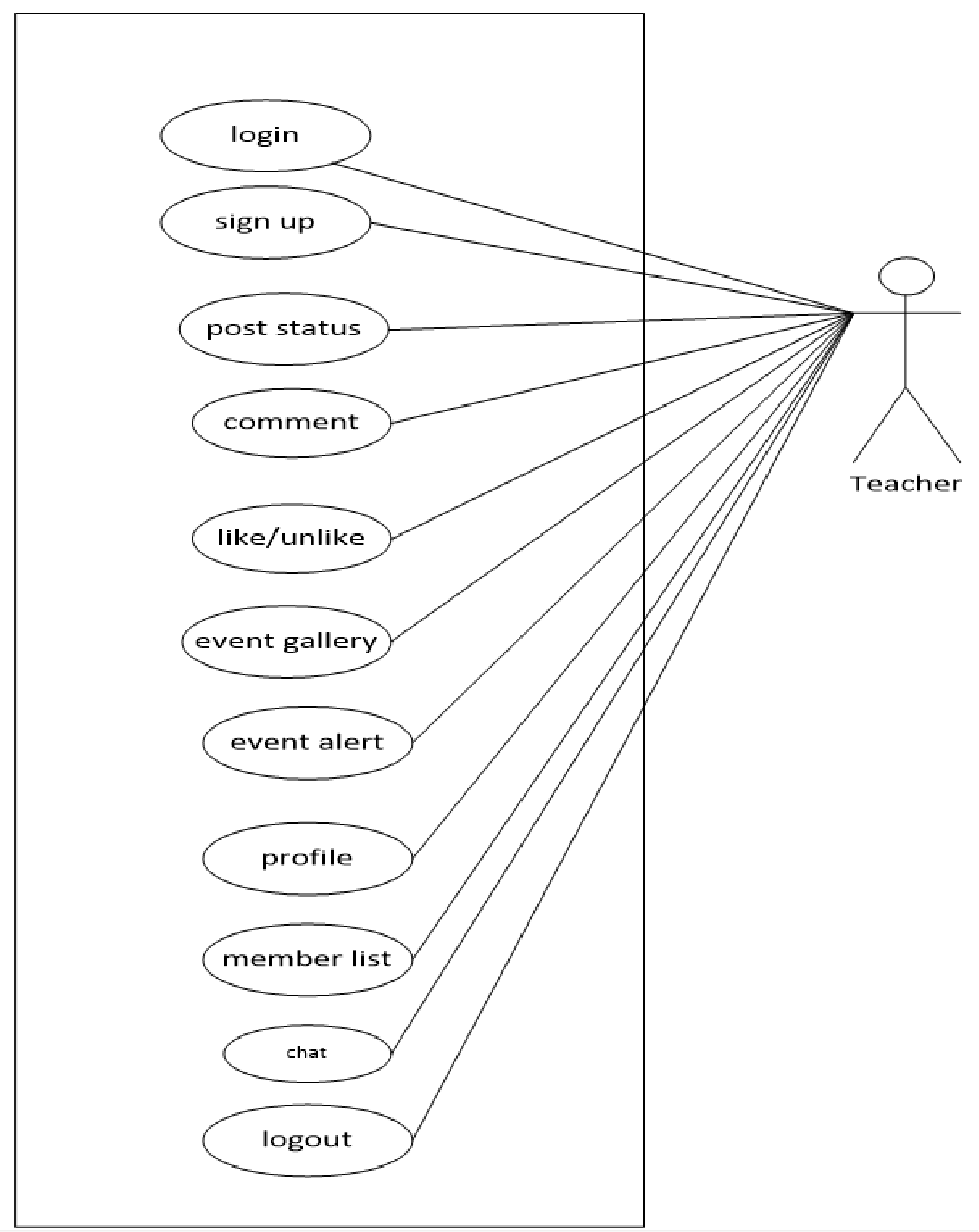
**4.10 Use Case Diagrams:**

Use case diagram is used to capture the dynamic nature of a system. It consists of use cases, actors and their relationships. It is used at a high level design to capture the requirements of a system.

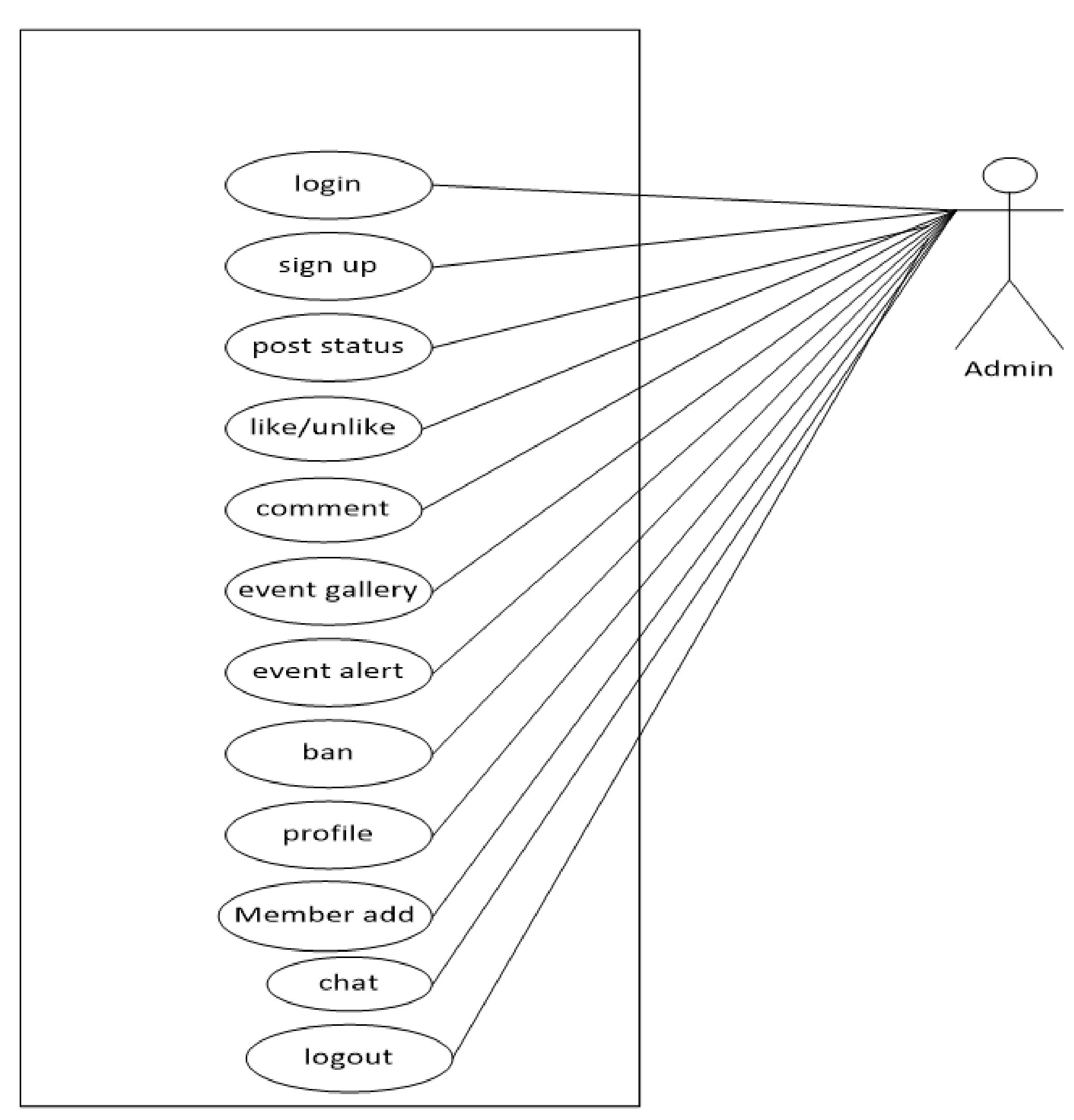


**Figure 16: Use case**

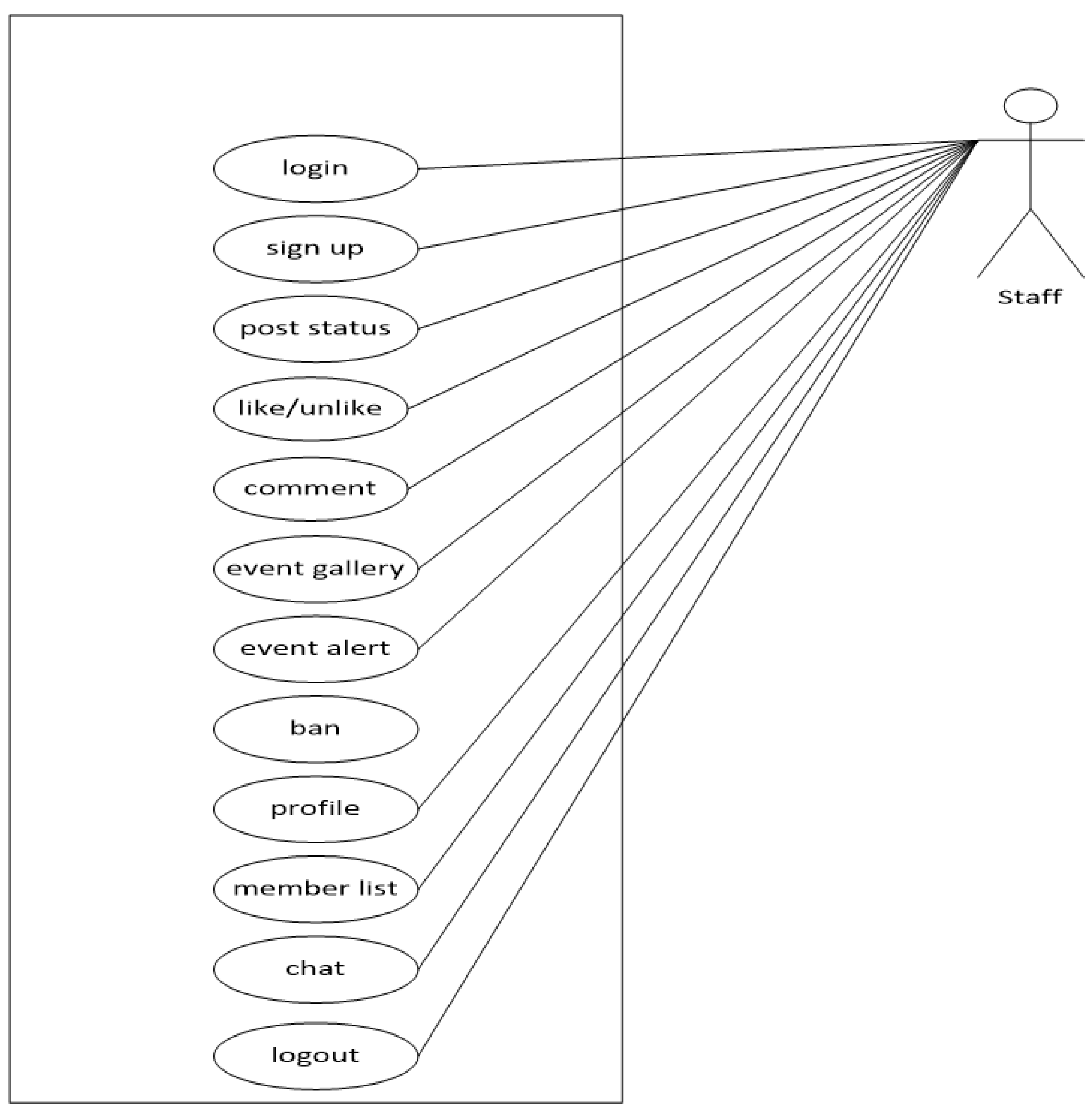
**Figure 17: Teacher Use case**



**Figure 18: Student Use case**



**Figure 19: Admin Use case**



**Figure 20: staff Use case**

**4.10.1 Use Cases Description**



**Figure 21: Description of Use Case logout**



**Figure 22: Description of Use Case Chat**



**Figure 23: Description of Use Case Login**



**Figure 24: Description of Use Case Status**



**Figure 25: Description of Use Case Profile**



**Figure 26: Description of Use Case Event Gallery**



**Figure 27: Description of Use Case Sign Up**



**Figure 28: Description Use Case of Comment**



**Figure 29: Description of Use Case Like/ Unlike**



**Figure 30: Description of Use Case member list**



**Figure 31: Description of Use Case event alert**

## **4.11 Sequence Diagrams**

## Sequence diagrams are used to capture time ordering of message flow in a system. Message from user actions flow towards system and system sends response message back to user.



**Figure 32: Login Scenario Sequence Diagram**



**Figure 33: login exception Scenario & Sequence Diagram**



**Figure 34: Sign Up Scenario & Sequence Diagram**



**Figure 35: Sign up exception Scenario & Sequence**



**Figure 36: Status Upload Scenario & Sequence Diagram**



**Figure 37: like/unlike Scenario & Sequence Diagram**



**Figure 38: Comment on status Scenario & Sequence Diagram**



**Figure 39:Event Gallery Scenario & Sequence Diagram**



**Figure 40: BAN Scenario & Sequence Diagram**



**Figure 41: Profile Update Scenario & Sequence Diagram**



**Figure 42: Chat Scenario & Sequence Diagram**



**Figure 43: Picture Upload Scenario & Sequence Diagram**

## **4.12 Activity Diagram**

Activity diagrams represent workflows in an graphical way.



**Figure 44: Activity Diagram for Login**



**Figure 45: Activity Diagram for Member Registration**



**Figure 46: Activity Diagram for Upload Profile Picture**



**Figure 47: Activity Diagram for Status Post**



**Figure 48: Activity Diagram for delete status**



**Figure 49: Activity Diagram for Status update**



**Figure 50: Activity Diagram for like status**



**Figure 51: Activity Diagram for Comment Status**



**Figure 52: Activity Diagram for Event Alert**



**Figure 53: Activity Diagram for Event Gallery Photo upload**



**Figure 54: Activity Diagram for User Ban**



**Figure 55: Activity Diagram for Message**



**Figure 56: Activity Diagram for delete Status**



**Figure 57: Activity Diagram for logout**

**Chapter 5**

# IMPLEMENTATION

This chapter consist of coding behind main functions of MUST social network.

## **5.1 Member Login and sign-up:**

When a member will login to the network he will be provided with login page and after login he will redirect to his related page according to his position. Following is the coding of front interface of this page:

<!doctype html>

<?php

<html>

<head>

<title>MUST Social Network</title>

<meta charset="utf-8">

<title>MUST social network</title>

</head>

<body>

<div id="head"><div id="left">

<span id="logo"><img src="img/logo.png" height="90px" width="90px" id="img"></span>

<span id="heading">MUST Social Network</span></div>

<div id="right">

<form action="chat/member.php" method="post">

<table>

<tr><td>User name</td><td>password</td></tr>

<tr><td><input type="text" name="user"></input></td>

<td><input type="password" name="pwd"></input></td>

<td colspan="1">

<input type="submit" value="login" name="submit"></input></td>< tr>

<td id="forget" role="button" style="color:white;">Forget password?</td>

</tr></tr></table></form></div></div>

<div id=main style="height:;">

<div id="lmain" style="height:650px;">

<img src="img/social-networking.jpg" width="1000" height="650px" id="mainimg"/>

</div>

<div id="rmain" style="height:650px;">

<h1>Sign Up</h1>

<form action="chat/member.php" method="post">

<table id="rmain1">

<tr><td><input type="text" name=fname id="input" placeholder=" Enter First Name" required></td></tr>

<tr><td><input type="text" name=lname id="input" placeholder=" Enter Last Name" required></td></tr>

<tr><td><input type="text" name=email id="input" placeholder=" Enter Email Address" required></td></tr>

<tr><td><input type="text" name=phone id="input" placeholder="Enter Mobile Number" required></td></tr>

<tr><td><input type="text" name=pwd id="input" placeholder=" Enter Password" required></td></tr>

<tr><td><input type="text" name=rpwd id="input" placeholder=" Re-Enter Password" required></td></tr>

<tr><td><input type="text" name=city id="input" placeholder=" Enter City Name" required></td></tr>

<tr><td> <select name="department" style="width:200px; text-align:center; " id="input" required>

<option selected disabled>select department</option>

<option value="BSSE">BSSE</option>

<option value="Banking">Banking</option>

<option value="Business">Business</option>

<option value="Math">Math</option>

</select></td></tr>

<tr><td><select name="position" style="width:200px; text-align:center; " id="input" required>

<option selected disabled>select Your Position</option>

<option value="Student">Student</option>

<option value="Teacher">Teacher</option>

<option value="Staff">Staff</option></select></td></tr>

<tr><td><input type="text" name=session id="input" placeholder=" Enter Your Session" required></td></tr>

<tr><td> <select name="question" style="width:200px; text-align:center; " id="input" required>

<option selected disabled>select security question</option>

<option value="Place of Birth?">Place of Birth?</option>

<option value="Favourite Movie">Favourite Movie?</option>

<option value="Favourite Place?">Favourite Place?</option>

<option value="First Pet Name?">First Pet Name?</option></select></td>

<tr><td><input type="date" name=dob id="input" placeholder="eg: 17/03/1990"></td></tr>

<tr><td>Gender</td></tr>

<tr><td>&nbsp&nbsp Male<input type="radio" name=gender value="male" id="gender" required></td></tr>

<tr><td>&nbsp&nbsp Female<input type="radio" name=gender value="female" id="gender" required></td></tr>

<tr><td><input type="submit" value="Sign Up" id="button" name="signup"></td></tr>

</table></form> </div>

<div style="height:650px; float:right; margin-right:70px; margin-top:20px; display:none;" id="forgetForm">

<table><form action="chat/member.php" method="post">

<tr><td><input type='number' name="cnic" placeholder="Enter CNIC" id="input" > </td></tr>

<tr> <td><select name="question" style="width:200px; text-align:center; " id="input" required>

<option selected disabled>select security question</option>

<option value="Place of Birth?">Place of Birth?</option>

<option value="Favourite Movie">Favourite Movie?</option>

<option value="Favourite Place?">Favourite Place?</option>

<option value="First Pet Name?">First Pet Name?</option></select></td></tr>

<tr><td><input type="text" name="answer" placeholder="Enter your answer" id="input">

</td></tr>

<tr><td><input type="password" name="password" placeholder="Enter new Password" id="input"></td></tr>

<tr><td><input type="submit" name="updPwdF" value="Change Password" class="btn btn-success" style="margin-top:5px;"> <tr><td></form></table></div></div>

</body>

<div class="text-center" style=" background:#555555; height:50px;float:left; color:white;">

<br> <p>MUST social Network © 2017</p>

</div>

</html>

## **Member.php**

function login()

{

session\_start();

$user=$\_POST['user'];

$pwd=$\_POST['pwd'];

include 'db.php';

$q8="select \* from member where email='$user' AND password='$pwd' AND ban='0'";

$result3=mysqli\_query($con,$q8);

while($row1=mysqli\_fetch\_array($result3))

{$id=$row1['id'];

$name=$row1['firstName'].' '.$row1['lastName'];

$position=$row1['position']; }

if (mysqli\_num\_rows($result3)==null)

{header("location:../login.php?msg=1");}

else{

$\_SESSION["myName"]=$name;

$\_SESSION["myUserName"]=$\_POST['user'];

$\_SESSION["pwd"]=$\_POST['pwd'];

$\_SESSION["myId"]=$id;

$\_SESSION["position"]=$position;

header("location:../index.php");}

}

function signup()

{

include 'db.php';

$firstName=$\_POST['fname'];

$lastName=$\_POST['lname'];

$email=$\_POST['email'];

$phone=$\_POST['phone'];

$password=$\_POST['pwd'];

$lastseen=0;

$cnic=$\_POST['cnic'];

$dob=$\_POST['dob'];

$gender=$\_POST['gender'];

$position=$\_POST['position'];

$ban=0;

$city=$\_POST['city'];

$session=$\_POST['session'];

$department=$\_POST['department'];

$question=$\_POST['question'];

$answer=$\_POST['answer'];

$updU="update member set firstName='$firstName', lastName='$lastName', phone='$phone', password='$password', lastseen='$lastseen', dob='$dob', gender='$gender', ban='$ban', department='$department', city='$city', securityQuestion='$question', answer='$answer' WHERE cnic='$cnic' AND password=0";

$updUrun=mysqli\_query($con,$updU);

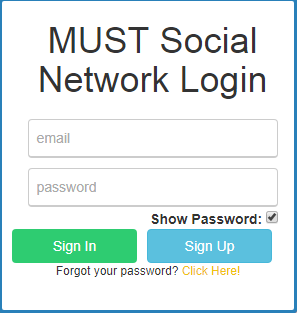
if($updUrun)

{header("location:../login.php?success");}

else

{header("location:../login.php?fail");}

}



**Figure: 58 Login.php**

## **Members panel:**

After login every member is directed to index page where related panel is displayed according to position.

## **Coding of index page:**

<?php

if ($\_SESSION["myUserName"]=='' AND $\_SESSION["pwd"]=='') {

{header("location:login.php");}}

<!DOCTYPE html>

<html lang="en">

<head>

<?php include 'chat/title.php';?>

<link rel="stylesheet" href="chat/stylesheet/bootstrap.min.css">

<script src="chat/javascript/jquery.min.js"></script>

<?php include 'chat/db.php'; ?>

</head>

<body >

<div class="container-fluid text-center">

<div class="row content" id="header">

<?php

include "chat/header.php";

include 'chat/status.php';

include 'chat/like.php';

include 'chat/comment.php';

include 'chat/member.php'; ?>

</div></div>

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header"> <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">

<span class="icon-bar"></span> <span class="icon-bar"></span><span class="icon-bar"></span>

</button></div>

<div class="collapse navbar-collapse" id="myNavbar">

<ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href=chat/timeline.php?profile=1&id=<?php echo $myId; ?>>My profile</a></li>

<li><a href="chat/chatEdit.php?name=''&msgEdit=2&sid=''">Message</a></li>

<li><a href="chat/eventGallery.php?view=all">Event Gallery</a></li>

<li><a href="chat/eventAlert.php">Event Alert</a></li>

<?php

if($\_SESSION['position']=='Admin') { ?>

<li><a href="chat/userAddBlock.php?add=1">Add User</a></li>

<li><a href="chat/userAddBlock.php?block=1">Block user</a></li>

<?php } ?>

</ul>

<ul class="nav navbar-nav navbar-right">

<li style=" background:url(img/msg.ico) no-repeat center; height:40px; width:40px; " id="msgnoti">

<span style=" margin-left:12px; ">

<?php

$count="select status from message where recieverId='$myId' AND status=1";

$countRun=mysqli\_query($con, $count);

echo "<font color='red'>".mysqli\_num\_rows($countRun)."</font>";?>

</span> </li>

<div id="newMsg" style="display:none; z-index:999999; height:px; width:340px; ;">

<?php $memberObj->showMember('\*'); ?>

<script type="text/javascript">

$('#msgnoti').click(function(){

$('#newMsg').slideToggle("slow");

});

$('.hidenew').click(function(){

$('#newMsg').hide(); });

</script> </div>

<li><a href="chat/logout.php"><span class="glyphicon glyphicon-log-in"></span> Logout</a></li>

</ul></div> </div></nav>

<div class="container-fluid text-center">

<div class="row content" id="first">

<div class="col-sm-2 sidenav" style="background:#E8E6E6; border-right:1px solid #CCC8C8">

<ul class="nav" style="margin-left:0; text-align:left; background:#FCFCFC;">

<li class="active"><a href="index.php">Home</a></li>

<li><a href=chat/timeline.php?profile=1&id=<?php echo $myId; ?>>My profile</a></li>

<li><a href="chat/chatEdit.php?name=''&msgEdit=2&sid=''">Message</a></li>

<li><a href="chat/eventGallery.php?view=all">Event Gallery</a></li>

<li><a href="chat/eventAlert.php">Event Alert</a></li>

</ul></div>

<div class="col-sm-7 text-left" style="background:#F7F5F5; border-right:1px solid #CCC8C8" id="centerDiv">

<div class="form-group">

<form action="chat/status.php" method="post" role="form">

<input name="status" type="text" required="required" class="form-control" id="status" placeholder=" What's on your mind?" maxlength="255"/>

<input type="submit" name="statusInsert" value="Post Status" class="btn btn-default" style="margin-top: 5px;" />

<input type="hidden" name="indexDel" value="1"></input></form>

<div id="statusbox" style="height:580px;" >

<div id="statuses" style="overflow-y:scroll; overflow:hidden;">

<?php

$q3="select \* from status ORDER BY id DESC";

$result=mysqli\_query($con,$q3);

while($row=mysqli\_fetch\_array($result))

{ ?>

<table style="background:white; box-shadow:5px 5px 5px rgba(197,190,191,1.00); margin-bottom:15px; border:1px solid #C4C4C4; " class="col-md-12" >

<tr > <td style="color:#338dcc; font-weight:bold; padding-top:10px;">

<?php $status\_posterid=$row['statusPosterId'];

$status\_poster=$row['posterName'];

$status\_date=$row['date'];

$status=$row['status'];

$statusId=$row['id'];

$statusObj->statusShow($status\_posterid,$status\_poster,$status\_date,$status,$statusId); ?>

<table class="statusTable" id="navDown<?php echo $statusId; ?>" border="1px" style=" width="100px" align="right" >

<tr style="height:25px;" align="center" ><td>

<form action="chat/status.php" method="post" style="padding-left:-10px;">

<input type="hidden" name="statusId" value="<?php echo $statusId; ?>">

<input type="submit" name="statusDel" value="delete" style="border:0px; background:#f7f5f5; font-weight:normal; color:#338dcc;" id="btnDel2">

</form></td></tr>

<tr id="editBtn<?php echo $statusId; ?>" style="height:25px; font-weight:normal;" align="center"><td>

<a role="button" class="btnEdit" id="<?php echo $statusId; ?>" style="margin-left:-12px; text-decoration:none;"> edit </a>

</td></tr></table>

</td></tr>

<tr><td id="td3" >

<span id="statustext3">

<input type="hidden" value="<?php echo $row['id']; ?>" name="statusId" >

<input type="hidden" value="<?php echo $myId; ?>" name="likerId" id="likerId<?php echo $row['id']; ?>">

<input type="hidden" value="<?php echo $\_SESSION["myName"]; ?>" name="likerName" id="likerName<?php echo $row['id']; ?>">

<div style="float:left">

<span id="like<?php echo $row['id']; ?>" style="float:left">

&nbsp&nbsp&nbsp <input type="submit" name="like" class="btnlike" id="<?php echo $row['id']; ?>" value="like" /><br>

</span></div>

<span id="unlike<?php echo $row['id']; ?>" style="float:left; display:none; margin-left:17px;">

<input type="submit" name="unlike" id="<?php echo $row['id']; ?>" value="like" /> </span>

<div style="margin-top:-19px;">

<div id="commentShow<?php echo $row['id']; ?>" > </div>

<div class="col-md-12" style="margin-top:5px;">

<input type="hidden" value="<?php echo $\_SESSION["myName"]; ?>" name="commenterName" id="commenterName<?php echo $row['id']; ?>" >

<input type="text" name="comment" id="commentBox<?php echo $row['id']; ?>"/>

<input type="submit" onClick="commentInsert(<?php echo $row['id'];?>)" name="btnComment" class="btnComment btn btn-default" id="cmt<?php echo $row['id']; ?>" value="Comment"/>

</div>

<div id="Comment<?php echo $row['id']; ?>" style="margin-top:-10px;">

<?php $comObj->commentShow($row['id']);?>

</div> </div></span>

</td></tr></table>

<?php } ?> </div>

</div> </div>

<div class="col-sm-3 sidenav" id="middle" style="background:#E8E6E6;">

<div id="chatwin" >

<div style="background:blue; width:305px; float:left;">

<span style=" float:right; margin-right:5px;" id="closeChatwin">

<a role="button" style="color:Red">X</a>

</span> </div>

<div class="" id="showMsg" style="width:320px; > </div>

<div style="width:300px; ">

<textarea type=text name="msgs" onkeyup="check(event, this)" cols="40" rows="5" required style="height:40px;"></textarea>

<button name="insert" onClick="insertd >Send</button></div></div>

<div style="background:#d9534f; padding:8px 10px"> Members </div>

<div id="member" class="row" style=" height:670px; overflow:hidden; overflow-y:scroll;">

<div style="border-bottom:4px solid white; width:100%" id="admin "> Admin </div>

<div style="display:none;"> <?php $memberObj->showMember("Admin"); ?> </div>

<div id="staff" style="border-bottom:4px solid white; width:100%"> Staff </div>

<div id="staffList" style="display:none;">

<?php $memberObj->showMember("Staff");?> </div>

<div id="teacher" style=" border-bottom:4px solid white; width:100%" " > Teacher</div>

<div id="teacherList" style="display:none">

<?php $memberObj->showMember("Teacher"); ?>

</div>

<div style=" width:100%" id="student" class="btn btn-warning"> Student </div>

<div id="studentList" style="display:none"> <?php $memberObj->showMember("Student");?> </div>

</table>

</div> </div></div>

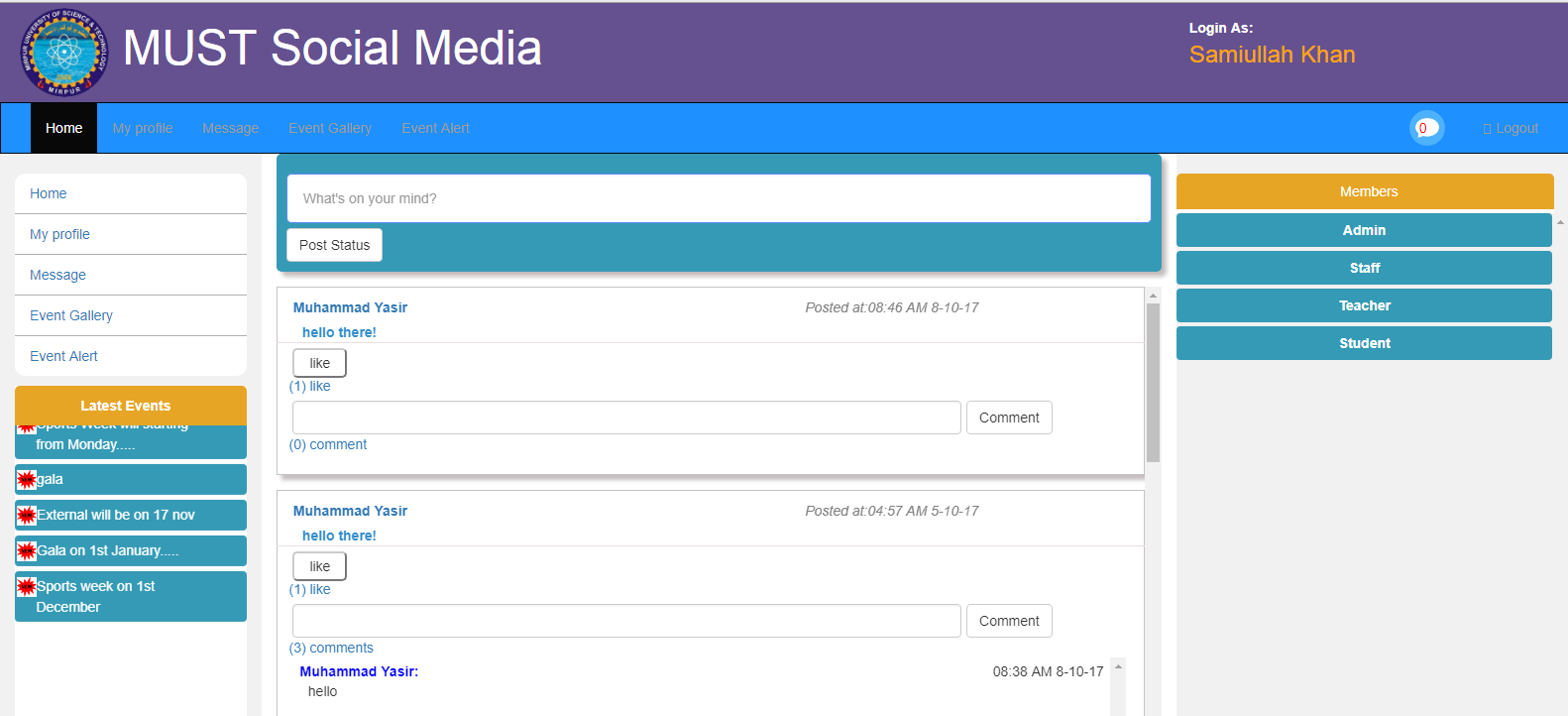
<footer class="container-fluid text-center">

<p>MUST social Network © 2017</p>

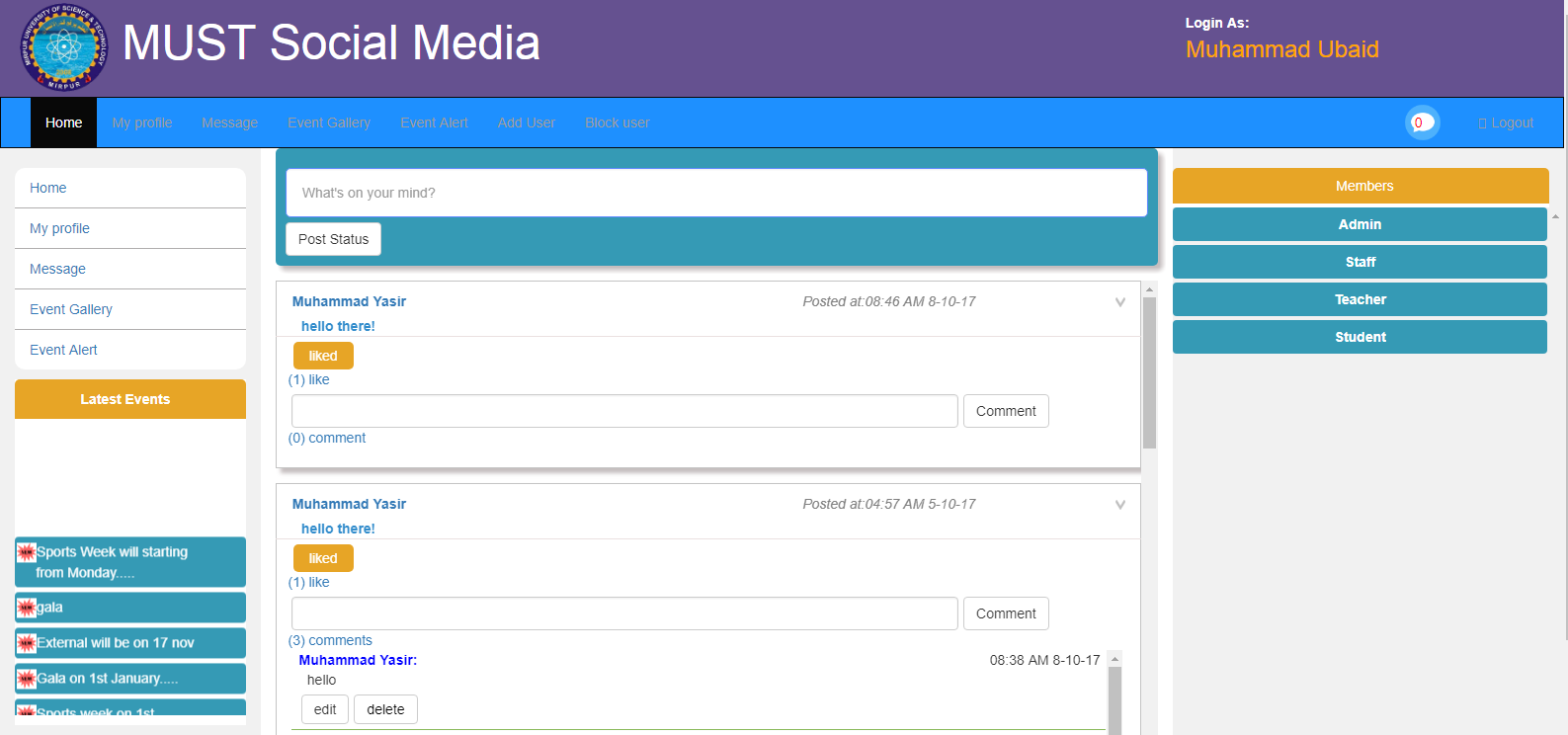
</footer>

</body>

</html>



**Figure: 59 Teacher, Staff & Student Panel**



**Figure: 60 Admin panel**

## **My Profile:**

function viewProfile($id)

{

include 'db.php';

$p="select \* from member where id='$id'";

$prun=mysqli\_query($con, $p);

$pResult=mysqli\_fetch\_array($prun); ?>

<?php if($\_SESSION['myId']==$id) { ?>

<button id="editBtn" > Edit Profile </button>

<button id="PwdChg"> Change password </button>

$myId=$\_SESSION['myId'];?>

<form action="member.php" method="post">

<table id="view">

<tr> <td style="width:30%"> First Name </td>

<td> <?php echo $pResult['firstName']; ?> </td> </tr>

<tr><td> Last Name </td>

<td> <?php echo $pResult['lastName']; ?> </td> <tr>

<tr><td>Email</td>

<td><?php echo $pResult['email']; ?> </td></tr>

<tr><td>Date of Birth </td>

<td> <?php echo $pResult['dob']; ?> </td></tr>

<?php if($\_SESSION['myId']==$pResult['id']) { ?>

<tr><td> Phone #</td>

<td><?php echo $pResult['phone']; ?> </td></tr>

<?php } ?>

<tr><td> Gender </td> <td><?php echo $pResult['gender']; ?></td> </tr>

<tr><td> Position </td> <td> php echo $pResult['position']; ?> </td></tr>

<tr><td>Department </td><td><?php echo $pResult['department']; ?></td> </tr>

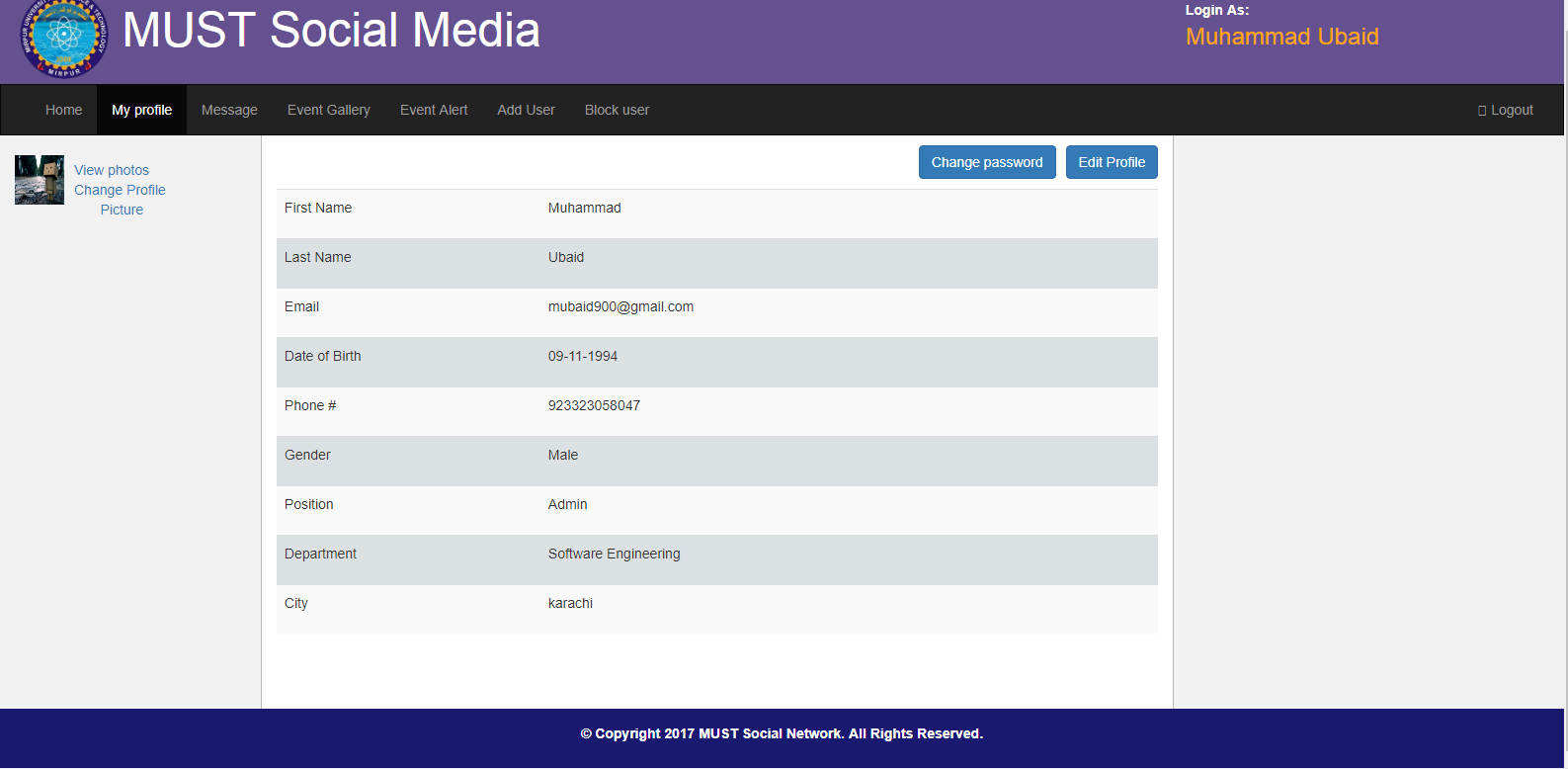
<tr><td> City </td><td> <?php echo $pResult['city']; ?> </td></tr>

<tr><td> Session </td><td> <?php echo $pResult['session']; ?> </td>

</tr>

</table?

<?php } ?>



**Figure 61: Profile.php**

## **Event Gallery:**

function photoShow()

{

include 'db.php';

if(isset($\_GET['p']))

{ $page=$\_GET['p']; $page=($page\*12)-12; }

else{ $page=0; }

$photo="select \* from eventgallery ORDER BY id DESC limit $page,12;";

$photoRun=mysqli\_query($con, $photo);

while($photos=mysqli\_fetch\_array($photoRun))

{ $name=$photos['photoName']; $photoId=$photos['id']; $date=$photos['date']; ?>

<div style=" width:170px; background:#FCF9F9; box-shadow:5px 5px 5px #ADADAD;">

<span style="float:left; margin-left:5px; margin-top:5px; margin-bottom:5px;">

<img src="eventGallery/<?php echo $name; ?>" height="160px" width="160px" onClick=zoom('eventGallery/<?php echo $name; ?>',400,500); role="button">

</span>

<span > uploaded at:<?php echo $date; ?> </span>

<?php if($\_SESSION['position']=='Admin' OR $\_SESSION['position']=='Staff') { ?>

<span style=" margin-left:50px; float:left; margin-top:5px; margin-bottom:5px;">

<form action="eventGalleryClass.php" method="post">

<input type="hidden" name="photoId" value="<?php echo $photoId; ?>">

<input type="hidden" name="photoName" value="<?php echo $name; ?>">

<input type="submit" value="delete" name="deletePhoto" class="btn btn-default">

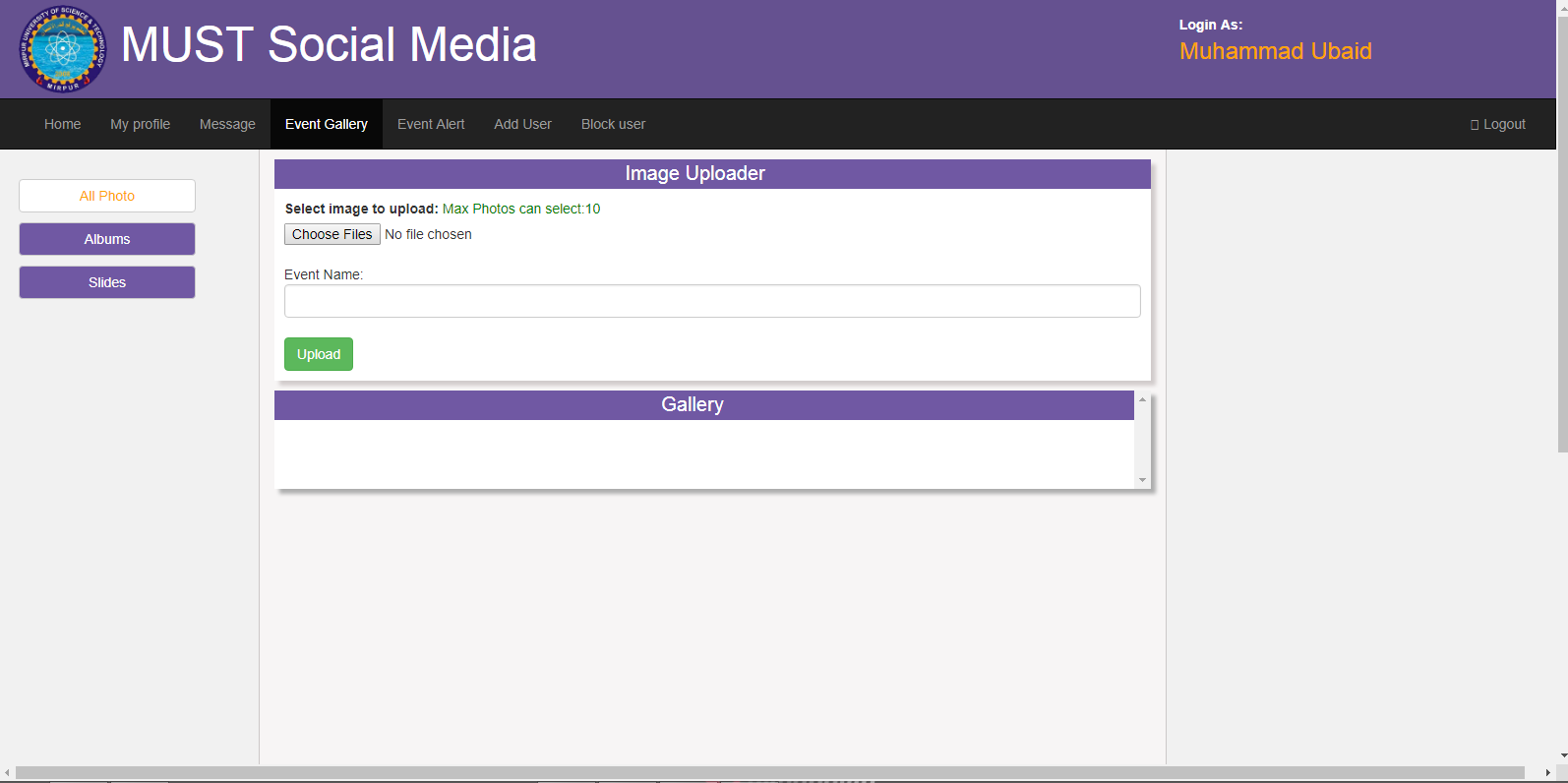
</form>

</span>

<?php } ?>

</div>

<?php } } } ?>



**Figure 62: Event Gallery**

## **Event Alert:**

function showEvent()

{

include 'db.php';

$alert2="select \* from eventalert order by id desc";

$showAlert=mysqli\_query($con, $alert2);

while($row2=mysqli\_fetch\_array($showAlert))

{ $id=$row2['alertGeneratorId'];

$q2="select \* from member where id='$id'";

$q2run=mysqli\_query($con, $q2);

$result2=mysqli\_fetch\_array($q2run);

$name=$result2['firstName']." ".$result2['lastName'];

$eventId=$row2['id'];?>

<div style=" background:white; box-shadow:5px 5px 5px #9A9999; padding:10px;

<span style="float:left; color:#789DF3; font-weight:600;"> <?php echo $name; ?> </span>

<span style="float:right;">

<?php echo $row2['date'];

if($\_SESSION['myId']==$id) { ?>

<span style="color:red" role="button" id="editbtn"> edit </span>

<form action="eventAlertClass.php" method="get" style="float:right;">

<input type="hidden" name="eventId" value="<?php echo $eventId; ?>">

<input type="submit" name="delete" value="X" style=" color:red; border:0;">

</form> <?php }?></span>

<span >

<?php echo wordwrap($row2['event'],90,"<br>\n", true); ?> </span>

<span style=" width:100%; display:none;" id="updField">

<form action="eventAlertClass.php" method="get">

<input type="hidden" name="eventId" value="<?php echo $eventId; ?>">

<input name="event" value="<?php echo $row2['event']; ?>" class="form-control">

<input type="submit" name="update" value="update" class="btn btn-success" style="margin-top:5px;">

</form>

</span>

<?php if($\_SESSION['position']=='Admin' OR $\_SESSION['position']=='Staff') { ?>

<form action="eventAlertClass.php" method="post">

<input type="hidden" name="department" value="<?php echo $row2['alertReciever']; ?>">

<input type="hidden" name="event" value="<?php echo $row2['event']; ?>">

<input type="submit" name="sendSms" class="btn btn-default" value="Send sms alert">

</form>

<?php } ?>

<span style="float:right; margin-top:-10px; color:#FF6366;">

Alert For:<?php if($row2['alertReciever']=='\*') echo "All"; else echo $row2['alertReciever']?>

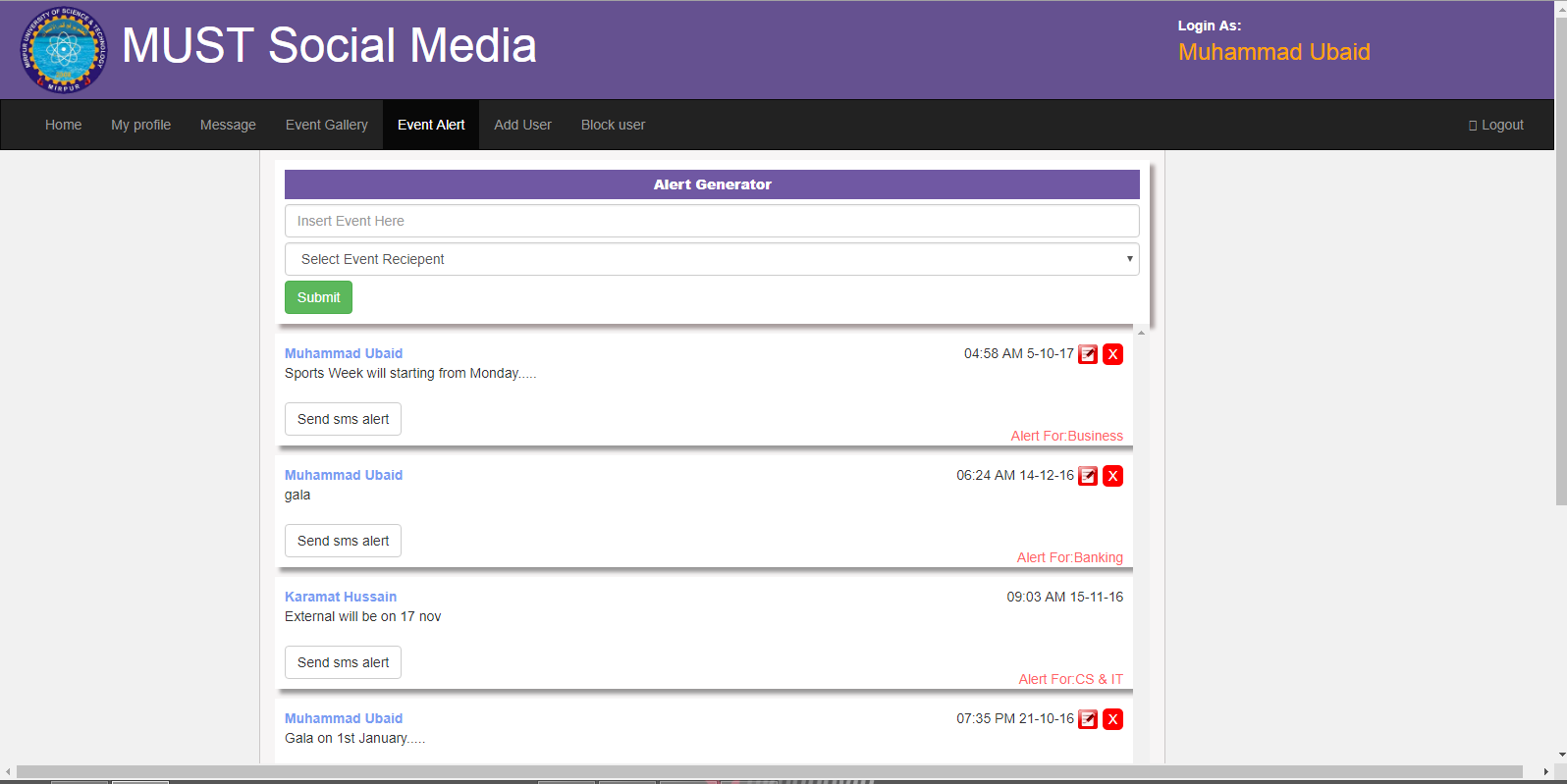
</span>

</div>

<?php }

}

?>



**Figure: 63 Event Alert**

**Chapter 06**

# SOFTWARE TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. As a secondary benefit, testing demonstrates that software functions appear to be working according to the specification. Testing provides a good indication of software reliability and some indication of software quality as a whole. Testing cannot show the absence of defects, it can only show that software defects are present. Finally we arrive at system testing and we tried our best to test each individual module and also as an integrated modules (as a whole) with sufficient data that may an organization have, Fulfilling the objective of our “MUST social network”.

## **6.1 Psychology of Testing**

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent

Should be to show that a program doesn’t work with the intent of finding errors

## **6.2 Testing Objective**

Objective of testing is to find maximum errors in the software. So, a good testing strategy is the one with high potential of finding errors and bugs.

* Testing is a process of executing a program with the intent of finding an error.
* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

## **6.3 The Box Approach**

Traditionally software testing methods are divided into white- and black-box testing. These two approaches describe the point of view of test designer.

## **6.3.1 White Box testing**



White-box testing, also known as clear box testing, glass box testing which tests internal workings of a program, instead of the functionality. In white-box testing detailed internal perspective of the system, as well as programming skills, are used to design test cases. . I tested step wise every piece of code, taking care that every statement in the code is executed at least once.

## **6.3.2 Black Box testing**



Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal software workings. The tester is only aware of what the software is supposed to do, not how it does it, like the figure depicts, tester only knows the input and output and nothing more.

## **6.4 Levels of Testing**

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are as shown below

Acceptance

Testing

Client Needs

Requirements

System Testing



Design

|  |  |  |
| --- | --- | --- |
| Code | Integration Testing |  |
|  |  |
|  | Unit Testing |  |



## **6.5 Unit Testing**

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

In this application we tests the programs up as system. Unit testing is first done on modules, independent of one another to locate errors. This enables to detect errors. All components of our project were tested with driven programs. Even each function was tested individually using the test cases which are described at the end of chapter.

## **6.6 Integration Testing**

After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions. In this project integrating all the modules forms the main system. When integrating all the modules we have checked whether the integration effects working of any of the services by giving different combinations of inputs. All modules were collected and integrated to each other and results of each test were up to the mark.

## **6.7 System Testing**

After testing each module and getting the expected results, here the modules are integrated to make sure that the application is well supporting its features. As a standalone application, its results are according to requirements, as expected. The application runs smoothly when executed.

* Application is reliable and less prone to crash.
* Application is capable enough to meet all the requirements of users.
* The application is easy to maintain.
* Application is easy to access when needed by user.

## **6.8 Alpha Testing**

Alpha testing is specially used by product development organizations. Developers observe the users and note problems. Alpha testing is testing of an application when development is about to complete. Minor design changes can still be made as a result of alpha testing.

During alpha testing of our project we find following minor errors and solve them such as

* 1. Broken links
  2. Spelling mistakes

## **6.9 Beta Testing**

Beta Testing is conducted at the client’s place i.e. developer is not present while testing as it errors and other issues encountered during this testing and reports to the developer.

**6.10 Acceptance Testing**

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system, the internal logic of program is not emphasized. In this project ‘MUST social network’ we tested whether project is working correctly or not.

**6.11 Interface Testing**

The system has been tested at interface level. The interface is tested to make sure that the form, menus, text boxes and buttons are operational and everything is displayed according to the requirement. For example menu strip of the Student Panel is tested.

**6.12 MUST Social Network Testing**

We tried our best to test each individual module and also integrated modules (as a whole) with

Sufficient data that the targeted organization may have, and debugged the software to an appreciable extent and we got success in this.

## **6.13 Test Cases**

* Test cases are built around specifications and requirements, i.e., what the application is

Supposed to do

* Test cases are generally derived from external descriptions of the software, including

Specification, requirements and design parameters.

* The test designer determines input, output and possible flow of events without any knowledge of the software’s internal structure.

We built and run following test cases for black box testing.

## **6.13.1 Test Case 1: Sign-up**

|  |  |
| --- | --- |
| ID | 1 |
|  |  |
| Name | Sign-up |
|  |  |
| Brief Description | Used to Sign-up |
|  |  |
| Expected Input | firstName, lastName, email, password, lastSeen. Cnic, city, dob,gender, position, ban, department |
|  |  |
| Expected flow of Events | Provide correct information and click on sign-up button |
|  |  |
| Alternate flow of Events | User Quits to sign-up form |
|  |  |
| Pre-Condition | New member must provide correct information |
|  |  |
| Post Condition | New Member is Added |
|  |  |
| Actor | Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Sign-up**

## **6.13.2 Test Case 2: Login**

|  |  |
| --- | --- |
| ID | 2 |
|  |  |
| Name | Login |
|  |  |
| Brief Description | Used to Login |
|  |  |
| Expected Input | email, password |
|  |  |
| Expected flow of Events | Provide correct information and click on Login button |
|  |  |
|  |  |
| Alternate flow of Events | User Quits to Login |
|  |  |
| Pre-Condition | User must provide correct information |
|  |  |
| Post Condition | User must Login |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Login**

## **6.13.3 Test Case 3: Create Conversation**

|  |  |
| --- | --- |
| ID | 3 |
|  |  |
| Name | Create Conversation |
|  |  |
| Brief Description | Used to Create Conversation |
|  |  |
| Expected Input | firstPersonId, secondPersonId , del1, del2, date |
|  |  |
| Expected flow of Events | Conversation created |
|  |  |
| Alternate flow of Events | User Quits to create Conversation |
|  |  |
| Pre-Condition | User Exits |
|  |  |
| Post Condition | Conversation Created |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: View Conversation**

## **6.13.4 Test Case 4: View Conversation**

|  |  |
| --- | --- |
| ID | 4 |
|  |  |
| Name | Conversation |
|  |  |
| Brief Description | Used to Show Conversation |
|  |  |
| Expected Input |  |
|  |  |
| Expected flow of Events | Show Conversation if exits. |
|  |  |
| Alternate flow of Events | Nill |
|  |  |
| Pre-Condition | Exits Conversation |
|  |  |
| Post Condition | Conversation Shown |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: View Conversation**

**6.13.5 Test Case 5: Delete Conversation**

|  |  |
| --- | --- |
| ID | 2 |
|  |  |
| Name | Delete Conversation |
|  |  |
| Brief Description | Used to Delete Conversation |
|  |  |
| Expected Input | Nill |
|  |  |
| Expected flow of Events | Click “Delete” button to delete Conversation |
|  |  |
| Alternate flow of Events | User Quits to delete Conversation |
|  |  |
| Pre-Condition | Conversation must exits |
|  |  |
| Post Condition | Conversation is deleted |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete Conversation**

## **6.13.6 Test Case 6: Check Conversation**

|  |  |
| --- | --- |
| ID | 6 |
|  |  |
| Name | Check Conversation |
|  |  |
| Brief Description | Used to Check Conversation |
|  |  |
| Expected Input | firstPersonId, secondPersonId , date |
|  |  |
| Expected flow of Events | Previous Conversation check |
|  |  |
|  |  |
| Alternate flow of Events | User Quits to check Conversation |
|  |  |
| Pre-Condition | Conversation must exits |
|  |  |
| Post Condition | Conversation checked And show to user |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Check Conversation**

## **6.13.7 Test Case 7: Add Member**

|  |  |
| --- | --- |
| ID | 7 |
|  |  |
| Name | Add Member |
|  |  |
| Brief Description | Used to Add Member |
|  |  |
| Expected Input | email, Cnic, |
|  |  |
| Expected flow of Events | New Member is Added in database by admin |
|  |  |
| Alternate flow of Events | User Quits to provide correct information |
|  |  |
| Pre-Condition | New member must provide correct information |
|  |  |
| Post Condition | User become able to join MUST network |
|  |  |
| Actor | Admin |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Add Member**

## **6.13.8 Test Case 8: Ban Member**

|  |  |
| --- | --- |
| ID | 8 |
|  |  |
| Name | Ban Member |
|  |  |
| Brief Description | Used to Ban Member |
|  |  |
| Expected Input | firstName, lastName, email, password, lastSeen. Cnic, city, dob,gender, position, ban, department |
|  |  |
| Expected flow of Events | User may or may not ban |
|  |  |
| Alternate flow of Events | Admin Quits to Ban |
|  |  |
| Pre-Condition | User Exits |
|  |  |
| Post Condition | User Banned |
|  |  |
| Actor | Admin |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Ban Member**

## **6.13.9 Test Case 9: Show Member**

|  |  |
| --- | --- |
| ID | 9 |
|  |  |
| Name | Show Member |
|  |  |
| Brief Description | Used to Show Member |
|  |  |
| Expected Input | firstName, lastName, email, password, lastSeen. Cnic, city, dob,gender, position, ban, department |
|  |  |
| Expected flow of Events | Members who are online are shown to other members |
|  |  |
| Alternate flow of Events | User Quits |
|  |  |
| Pre-Condition | Member Must Exits |
|  |  |
| Post Condition | Member shown |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Show Member**

## **6.13.10 Test Case 10: Status Check**

|  |  |
| --- | --- |
| ID | 10 |
|  |  |
| Name | Status Check |
|  |  |
| Brief Description | Used to Status Check |
|  |  |
| Expected Input | Status, posterName, date |
|  |  |
| Expected flow of Events | Status is Checked to find where user become online or seen the status |
|  |  |
| Alternate flow of Events | Status not check if any error is occur |
|  |  |
| Pre-Condition | Status must exits |
|  |  |
| Post Condition | Status checked |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Status Check**

## **6.13.11 Test Case 11: view Profile**

|  |  |
| --- | --- |
| ID | 11 |
|  |  |
| Name | View Profile |
|  |  |
| Brief Description | Used to View Profile |
|  |  |
| Expected Input | firstName, lastName, email, password, lastSeen. Cnic, city, dob,gender, position, ban, department |
|  |  |
| Expected flow of Events | User must views his own profile |
|  |  |
| Alternate flow of Events | User Quits to View profile |
|  |  |
| Pre-Condition | Profile must exits |
|  |  |
| Post Condition | Profile viewed by user |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: View Profile**

## **6.13.12 Test Case 12: Update Profile**

|  |  |
| --- | --- |
| ID | 12 |
|  |  |
| Name | Update Profile |
|  |  |
| Brief Description | Used to Update profile |
|  |  |
| Expected Input | firstName, lastName, email, password, lastSeen. Cnic, city, dob,gender, position, ban, department |
|  |  |
| Expected flow of Events | User should update his own profile |
|  |  |
| Alternate flow of Events | User Quits to update his profile |
|  |  |
| Pre-Condition | Profile must exits |
|  |  |
| Post Condition | Profile updated |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Update Profile**

## **6.13.13 Test Case 13: Delete Member**

|  |  |
| --- | --- |
| ID | 13 |
|  |  |
| Name | Delete Member |
|  |  |
| Brief Description | Used to delete Member |
|  |  |
| Expected Input | Member Id |
|  |  |
| Expected flow of Events | Member deleted |
|  |  |
| Alternate flow of Events | User Quits to sign-up form |
|  |  |
| Pre-Condition | Member must exist |
|  |  |
| Post Condition | New Member is Added |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete Member**

## **6.13.14 Test Case 14: View Status**

|  |  |
| --- | --- |
| ID | 14 |
|  |  |
| Name | View Status |
|  |  |
| Brief Description | Used to view Status. |
|  |  |
| Expected Input | Nill |
|  |  |
| Expected flow of Events | Status viewed by users that are available |
|  |  |
| Alternate flow of Events | Status not viewed by user |
|  |  |
| Pre-Condition | Status exists |
|  |  |
| Post Condition | Status Viewed |
|  |  |
| Actor | Administrator, Teacher, Staff, Student |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: View Status**

## **6.13.15 Test Case 15: Delete Status**

|  |  |
| --- | --- |
| ID | 15 |
|  |  |
| Name | Delete Status |
|  |  |
| Brief Description | Used to delete Status |
|  |  |
| Expected Input | Status, posterName, date |
|  |  |
| Expected flow of Events | Select Status And Delete the Status |
|  |  |
| Alternate flow of Events | User Quits |
|  |  |
| Pre-Condition | Status exist |
|  |  |
| Post Condition | Status deleted |
|  |  |
| Actor | Administrator, Teacher, Staff, Student |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete Status**

## **6.13.16 Test Case 16: Update Status**

|  |  |
| --- | --- |
| ID | 16 |
|  |  |
| Name | Update Status |
|  |  |
| Brief Description | Used to Update Status |
|  |  |
| Expected Input | Satus, posterName, date |
|  |  |
| Expected flow of Events | Select Status And Update the Status |
|  |  |
| Alternate flow of Events | User Quits |
|  |  |
| Pre-Condition | Status exist |
|  |  |
| Post Condition | Status Updated |
|  |  |
| Actor | Administrator, Teacher, Staff, Student |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Update Status**

## **6.13.17 Test Case 17: Insert Status**

|  |  |
| --- | --- |
| ID | 17 |
|  |  |
| Name | Insert Status |
|  |  |
| Brief Description | Used to Insert Status |
|  |  |
| Expected Input | Status, posterName, date |
|  |  |
| Expected flow of Events | Select Status ID And insert the Status |
|  |  |
| Alternate flow of Events | User Quits |
|  |  |
| Pre-Condition | Status exist |
|  |  |
| Post Condition | Status Inserted |
|  |  |
| Actor | Administrator, Teacher, Staff, Student |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Insert Status**

## **6.13.18 Test Case 18: Insert Profile Picture**

|  |  |
| --- | --- |
| ID | 18 |
|  |  |
| Name | Picture Insert |
|  |  |
| Brief Description | Used to Picture Insert |
|  |  |
| Expected Input | Name, picHolderid, picDate, currentStatus |
|  |  |
| Expected flow of Events | Profile Picture inserted |
|  |  |
| Alternate flow of Events | User Quits to insert profile picture |
|  |  |
| Pre-Condition | Profile exits |
|  |  |
| Post Condition | Profile Picture inserted |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Insert Profile Picture**

## **6.13.19 Test Case 19: Show Profile Picture**

|  |  |
| --- | --- |
| ID | 19 |
|  |  |
| Name | Show Profile Picture |
|  |  |
| Brief Description | Used to Show Profile Picture |
|  |  |
| Expected Input | name, picHolderid, picDate, currentStatus |
|  |  |
| Expected flow of Events | Profile picture is shown to the users who are the part of this network. |
|  |  |
| Alternate flow of Events | Profile picture is not shown due to some error |
|  |  |
| Pre-Condition | Picture exits |
|  |  |
| Post Condition | Profile picture is shown successfully |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Show Profile Picture**

## **6.13.20 Test Case 20: Change Profile Picture**

|  |  |
| --- | --- |
| ID | 20 |
|  |  |
| Name | Change Profile Picture |
|  |  |
| Brief Description | Used to Change Profile Picture |
|  |  |
| Expected Input | Name, picHolderid, picDate, currentStatus |
|  |  |
| Expected flow of Events | Every user change his own profile picture |
|  |  |
| Alternate flow of Events | User Quits to change picture |
|  |  |
| Pre-Condition | Profile exits in order to change profile picture |
|  |  |
| Post Condition | Profile picture changed |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Change Profile Picture**

## **6.13.21 Test Case 21: Delete Profile Picture**

|  |  |
| --- | --- |
| ID | 21 |
|  |  |
| Name | Delete Profile Picture |
|  |  |
| Brief Description | Used to Delete Profile Picture |
|  |  |
| Expected Input | Name, picHolderid, picDate, currentStatus |
|  |  |
| Expected flow of Events | User delete his own profile picture |
|  |  |
| Alternate flow of Events | User Quits to delete picture |
|  |  |
| Pre-Condition | Profile picture exits |
|  |  |
| Post Condition | Profile picture deleted |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete Profile Picture**

## **6.13.22 Test Case 22: Insert PhotoIn Event Gallery**

|  |  |
| --- | --- |
| ID | 22 |
|  |  |
| Name | Insert Photo In Gallery |
|  |  |
| Brief Description | Used to Insert Photo In Album |
|  |  |
| Expected Input | eventName, photoName, date |
|  |  |
| Expected flow of Events | Photo is inserted in album |
|  |  |
| Alternate flow of Events | User Quits |
|  |  |
| Pre-Condition | Photo album exits |
|  |  |
| Post Condition | News photos are inserted in album |
|  |  |
| Actor | Admin, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Insert Photo In Album**

## **6.13.23 Test Case 23: Delete PhotoFrom Album**

|  |  |
| --- | --- |
| ID | 23 |
|  |  |
| Name | Delete PhotoFrom Album |
|  |  |
| Brief Description | Used to Delete PhotoFrom Album |
|  |  |
| Expected Input | eventName, photoName, date |
|  |  |
| Expected flow of Events | Album photos deleted |
|  |  |
| Alternate flow of Events | User Quits |
|  |  |
| Pre-Condition | Photo album should exits |
|  |  |
| Post Condition | Photo album is deleted |
|  |  |
| Actor | Admin,Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete PhotoFrom Album**

## **6.13.24 Test Case 24: Show Album Photo**

|  |  |
| --- | --- |
| ID | 24 |
|  |  |
| Name | Show Album Photo |
|  |  |
| Brief Description | Used to Show Album Photo |
|  |  |
| Expected Input | eventName, photoName, date |
|  |  |
| Expected flow of Events | Album photos are shown to everyone |
|  |  |
| Alternate flow of Events | Photos are not shown due to some fault |
|  |  |
| Pre-Condition | Photo Album must exits |
|  |  |
| Post Condition | Photos in album are shown |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Show Album Photo**

## **6.13.25 Test Case 25: Slider**

|  |  |
| --- | --- |
| ID | 25 |
|  |  |
| Name | Slider |
|  |  |
| Brief Description | Used to Show Slider |
|  |  |
| Expected Input | eventName, photoName, date |
|  |  |
| Expected flow of Events | Pictures in slider are shown to everyone who join the network |
|  |  |
| Alternate flow of Events | Slider is not Shown to user |
|  |  |
| Pre-Condition | Pictures exits |
|  |  |
| Post Condition | Pictures are shown in slider |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Slider**

## **6.13.26 Test Case 26: Add Event**

|  |  |
| --- | --- |
| ID | 26 |
|  |  |
| Name | Add Event |
|  |  |
| Brief Description | Used to Add Event |
|  |  |
| Expected Input | event, alertReciever, date |
|  |  |
| Expected flow of Events | Event is added by admin and send it to user mobile phone through text message |
|  |  |
| Alternate flow of Events | No event is added by admin |
|  |  |
| Pre-Condition | User must exits in order to send an event |
|  |  |
| Post Condition | Event is received by the user |
|  |  |
| Actor | Admin, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Add Event**

## **6.13.27 Test Case 27: Edit Event**

|  |  |
| --- | --- |
| ID | 27 |
|  |  |
| Name | Edit Event |
|  |  |
| Brief Description | Used to Edit Event |
|  |  |
| Expected Input | eventName, photoName, date |
|  |  |
| Expected flow of Events | Admin change or update the event and then send to the user |
|  |  |
| Alternate flow of Events | Admin Quits to edit the event |
|  |  |
| Pre-Condition | Event exits in order to edit it |
|  |  |
| Post Condition | Changed or updated event is send to the user |
|  |  |
| Actor | Admin, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Add Member**

## **6.13.28 Test Case 28: Delete Event**

|  |  |
| --- | --- |
| ID | 28 |
|  |  |
| Name | Delete Event |
|  |  |
| Brief Description | Used to Delete Event |
|  |  |
| Expected Input | eventName, photoName, date |
|  |  |
| Expected flow of Events | Admin delete the event after sending it to the user |
|  |  |
| Alternate flow of Events | Admin Quits to delete an event after sending it to the user |
|  |  |
| Pre-Condition | Event exits |
|  |  |
| Post Condition | Event deleted |
|  |  |
| Actor | Admin, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete Event**

## **6.13.29 Test Case 29: Show Event**

|  |  |
| --- | --- |
| ID | 29 |
|  |  |
| Name | Show Event |
|  |  |
| Brief Description | Used to Show Event |
|  |  |
| Expected Input | Admin change or update the event and then send to the user |
|  |  |
| Expected flow of Events | Event successfully shown to the user after sending from the admin |
|  |  |
| Alternate flow of Events | Events are not shown to user due to some fault |
|  |  |
| Pre-Condition | Event must exits |
|  |  |
| Post Condition | Event shown |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Show Event**

## **6.13.30 Test Case 30: Like**

|  |  |
| --- | --- |
| ID | 30 |
|  |  |
| Name | Like |
|  |  |
| Brief Description | Used to Like the status. |
|  |  |
| Expected Input | likerId likerName, likes |
|  |  |
| Expected flow of Events | like status |
|  |  |
| Alternate flow of Events | Leave the status button without liking |
|  |  |
| Pre-Condition | Status Exits |
|  |  |
| Post Condition | Status liked |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Like**

## **6.13.31 Test Case 31: Unlike**

|  |  |
| --- | --- |
| ID | 31 |
|  |  |
| Name | Unlike |
|  |  |
| Brief Description | Used to unlike the Status. |
|  |  |
| Expected Input | likerId likerName, likes |
|  |  |
| Expected flow of Events | Unlike the Status |
|  |  |
| Alternate flow of Events | Leave the status button without unliking |
|  |  |
| Pre-Condition | Status Exists |
|  |  |
| Post Condition | Status Unlike |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Unlike**

## **6.13.32 Test Case 32: Like Show**

|  |  |
| --- | --- |
| ID | 32 |
|  |  |
| Name | Like show |
|  |  |
| Brief Description | Used to Like show |
|  |  |
| Expected Input | likerId likerName, likes |
|  |  |
| Expected flow of Events | The like that is done by the user is shown |
|  |  |
| Alternate flow of Events | Like is not shown to user |
|  |  |
| Pre-Condition | Status Exists |
|  |  |
| Post Condition | Like is shown to the user |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Like Show**

## **6.13.33 Test Case 33: Like Count**

|  |  |
| --- | --- |
| ID | 33 |
|  |  |
| Name | Like Count |
|  |  |
| Brief Description | Used to Like Count |
|  |  |
| Expected Input | likerId likerName, likes |
|  |  |
| Expected flow of Events | The likes that are done by the user are counted |
|  |  |
| Alternate flow of Events | Likes are not counted |
|  |  |
| Pre-Condition | Status Exists |
|  |  |
| Post Condition | Likes are counted successfully are shown to the user |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Like Count**

## **6.13.34 Test Case 34: Insert Message**

|  |  |
| --- | --- |
| ID | 34 |
|  |  |
| Name | Insert Message |
|  |  |
| Brief Description | Used to Insert Message |
|  |  |
| Expected Input | Message, senderId, senderName, recieverId, status, date |
|  |  |
| Expected flow of Events | The message is inserted by the user in order to send it to other user weather they are online or offine |
|  |  |
| Alternate flow of Events | Message is not inserted by the user |
|  |  |
| Pre-Condition | User Exists in order to insert message |
|  |  |
| Post Condition | Message inserted |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Insert Message**

## **6.13.35 Test Case 35: Delete Message**

|  |  |
| --- | --- |
| ID | 35 |
|  |  |
| Name | Delete Message |
|  |  |
| Brief Description | Used to Delete the message |
|  |  |
| Expected Input | Message, senderId, senderName, recieverId, status, date |
|  |  |
| Expected flow of Events | Message is deleted by the user which is send to the other user. The message is deleted from the side of the user who delete it not delete from both sides. |
|  |  |
| Alternate flow of Events | Message delete |
|  |  |
| Pre-Condition | messageExists |
|  |  |
| Post Condition | Deleted Message is not shown to the user who delete it |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete Message**

## **6.13.36 Test Case 36: Show Message**

|  |  |
| --- | --- |
| ID | 36 |
|  |  |
| Name | Show Message |
|  |  |
| Brief Description | Used to Show the Message |
|  |  |
| Expected Input | Message, senderId, senderName, recieverId, status, date |
|  |  |
| Expected flow of Events | The message is show to the user that is send by the other user. |
|  |  |
| Alternate flow of Events | Message is not show to the user |
|  |  |
| Pre-Condition | Message Exists |
|  |  |
| Post Condition | Message is shown to the user |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Show Message**

## **6.13.37 Test Case 37: Check Message Status**

|  |  |
| --- | --- |
| ID | 37 |
|  |  |
| Name | Check Message status |
|  |  |
| Brief Description | Used to Check Message status |
|  |  |
| Expected Input | Message, senderId, senderName, recieverId, status, date |
|  |  |
| Expected flow of Events | Check the status of the message whether the message is seen by the user or not |
|  |  |
| Alternate flow of Events | Message status is not checked |
|  |  |
| Pre-Condition | message Exists |
|  |  |
| Post Condition | Message of the status is checked |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Check Message Status**

## **6.13.38 Test Case 38: Add Comment**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 38 |  |  |
|  |  |  |  |
| Name | Add Comment |  |  |
|  |  |  |  |
| Brief Description | Used to Add Comment |  |  |
|  |  |  |  |
| Expected Input | comment, commenterId, commenterName, date |  |  |
|  |  |  |  |
| Expected flow of Events | Comment on Status |  |  |
|  |  |  |  |
| Alternate flow of Events | Not Comment on Status |  |  |
|  |  |  |  |
| Pre-Condition | Status exists |  |  |
|  |  |  |  |
| Post Condition | Comment On Status |  |  |
|  |  |  |  |
| Actor | Admin, Student, Teacher, Staff |  |  |
|  |  |  |  |
| Status | Pass |  |  |
|  |  |  |  |

**Test Case Name: Add Comment**

## **6.13.39 Test Case 39: Edit Comment**

|  |  |  |
| --- | --- | --- |
| ID | 39 |  |
|  |  |  |
| Name | Edit Comment |  |
|  |  | |
| Brief Description | Used to Edit the Comment | |
|  |  | |
| Expected Input | comment, commenterId, commenterName, date | |
| Expected flow of Events | 1.Edit the Comment | 2. Add new comment. |
|  |  |  |
| Alternate flow of Events | User Quits |  |
|  |  | |
| Pre-Condition | Status Exits | |
|  |  | |
| Post Condition | Update comment | |
|  |  |  |
| Actor | Admin, Student, Teacher, Staff |  |
|  |  |  |
| Status | Pass |  |
|  |  |  |

**Test Case Name: Edit Comment**

## **6.13.40 Test Case 40: Delete Comment**

|  |  |
| --- | --- |
| ID | 40 |
|  |  |
| Name | Delete Comment |
|  |  |
| Brief Description | Used to Delete the comment. |
|  |  |
| Expected Input | comment, commenterId, commenterName, date |
|  |  |
| Expected flow of Events | Delete comment |
|  |  |
| Alternate flow of Events | User Quits/Don’t delete the comment |
|  |  |
| Pre-Condition | Status Exits |
|  |  |
| Post Condition | Comment is Deleted |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Delete Comment**

## **6.13.41 Test Case 41: Show Comment**

|  |  |
| --- | --- |
| ID | 41 |
|  |  |
| Name | Show Comment |
|  |  |
| Brief Description | Used to Show Comment |
|  |  |
| Expected Input | comment, commenterId, commenterName, date |
|  |  |
| Expected flow of Events | Comment show to the user and other users who join the network |
|  |  |
| Alternate flow of Events | Comment is not shown |
|  |  |
| Pre-Condition | Comment must exits |
|  |  |
| Post Condition | The comment on status is shown to the user |
|  |  |
| Actor | Admin, Student, Teacher, Staff |
|  |  |
| Status | Pass |
|  |  |

**Test Case Name: Show Comment**

**Chapter 07**

# CONCULISION & FUTURE WORK

## **7.1 Conclusion**

It has been a great pleasure for us to work on this exciting and challenging project. During our four years of university life in MUST we feel that there is lack of communication among teachers and students. There is no effective alert service which alert all the student and other members of university in time. Using this departments of MUST can alert their students about exams.

This project proved good for us as it provided practical knowledge in PHP, AJAX, JQUERY, and JAVASRIPT. We learn how to use PHP and Ajax together to provide an environment like online chat where window load data without getting refresh. JQuery provide us different effects like slide down, slide up, hide and show a window. Also about all handling procedure related with “MUST social network’.

Development of this software also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently. We also came to know that how some websites uses different languages to facilitate their target audience.

Social Networking website is a revolutionary idea with a very bright future with further scope for advancements. The opportunities provided from this medium are immense and many organizations are making use of this medium to better their practices. Online communities and Blogs are becoming very popular and moreover since the advancement of embedded systems people can use them “on the go” with the help of handheld devices like cell phones or palmtops. The world is getting closer every day and everyone wants to be connected. Static blogs and websites are losing popularity. World is moving more towards "information streams". The information comes to users rather than users have to make effort to get the information.

The proposed system is built successfully and it performs according to the requirements specified in the requirements specification document. System is satisfactorily tested with example data, it provides intended results. It is found to be bug free as per the testing standards that are implemented. We hope that it will be helpful for MUST in future. University has not a platform where student can discuss their problems. The alert system will be very useful in alerting students about different events.

## **7.2 Future Improvements**

Enhancement and changes can be made to the software, files or procedures to meet the emerging requirements. Since organization systems and the business environment undergo continual changes, the information system should keep place.

As the system has a database so in future, it can easily be extended to include a management system that may contain all records. This extension will help the administration to let go the paper work and do their work with simple clicks.

Photo upload in status can be added in future according to requirements of audience. Friend Tag future can be also added. One friend can tag other friend can tag other friends in his post or in a photo

In chat typing feature can also be added by using this feature user can know that the other user is typing something or not.

In event gallery modifications can also be made. Slide effects can be added. User can view photo using different effects.

## **7.3 Limitations**

* 1. The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity.
  2. We are not professional software engineers and developers. So there may be some minor mistakes.
  3. There may be flexibility needed in some cases.
  4. Only MUST staff, teachers & students can join this network

## **8.1 REFERENCES**

We take help from the following books and web Sites for the software engineering and development of this project.

**[i]** Michal Bah .Object Oriented Modeling with UML 2.0. Publisher: PEARSON EDUCATION, 2007

**[ii]** Summerville, Ian (2011). Software engineering. Boston: Pearson. ISBN 0-13-705346-0

**FOR DEPLOYMENT** [Access Date: Sep-2017 ]

<https://www.youtube.com/watch?v=Nm5n9MMVOuI>

<http://www.w3schools.com/bootstrap/bootstrap_buttons.asp>

http://www.w3schools.com/bootstrap/bootstrap\_forms.asp

<http://stackoverflow.com/questions/33197179/ajax-delete-multiple-records-with-checkboxes>

<http://api.jquery.com/slidetoggle/>

http://asethome.org/soft/soft\_Process2.html