Table of Contents

1. Introduction:

Today Social networking sites (SNSs) have gained the tremendous popularity among the youth causing a lot of attention of researcher both for academic and industrial purpose towards this. This paper starts with the introduction of SNS and previous work. We have also presented a survey report for existing SNS. We have developed our own SNS (Quipmate) with a lot of innovative and creative ideas that try to simulate the offline social community to online. This explores all the features of this SNS and also addresses the challenges that we have to resolve for the global operation of this site. .Along with this we also compare the features of Quipmate with other existing SNSs. This study is intended for researcher to serve as a reference for making them educated about SNSs and their features.

2. Quipmate Feature:

Quipmate is the new initiative taken by us. The term "Quipmate" comprises of two word, i.e. Quip + Mate. According to Cambridge Advanced Learner Dictionary, quip means "a clever and amusing remark". Thus Quipmate means making clever and amusing remarks on your mates i.e. your friends. The main theme of this SNS is to put the offline structure online and to create and maintain the online identity of a person by creating their diary and do the best to describe the feeling of person on the Internet.

This SNS incorporates many new concepts that are direct taken from our day-to-day life experiences. The new concept that we incorporates in this SNS are ---

Diary:

We create the online diary of the members of Quipmate. This concept is simulations of the diaries that people are used to write. They maintain all of their activities and experiences. Thus in this era of the Internet, we provide them online diary structure to do the same and provide a lot extra facilities to manage them. In the dairy, all the activity of members is recorded and the owner of diary can control the visibility and privacy of the post from the friends. There are two types of the diary we have implemented. First one is My Diary is the default diary of the members in which all the activities of the member is recorded along with the friends activity on that member. Second is the Interest Diary that is the diary any one can create on the basis of their interest/hobby. The members of the Quipmate can be the fan of diary on the basis of that diary privacy. Interest Diary privacy is Open Diary/Close Diary.

Status Song & Song Dedication:

We are trying to do our best to put the feeling of person online. Music is the best way to express the feeling. With this whole idea, we have launched the feature named "Status Song". In this feature, there is an interface via which we can search the songs and put it as a status. Friends can play these songs and listen to them. We also know that there are a lot of songs which describe the same as your feelings, thus it will be easy to share our feeling via songs. This feature will be a huge hit because on average (figures) % of the [Statistics on Music Lovers] person love music. Along with Status Song, we can also dedicate a song to our friends. Unlike Status Song, which expresses what you are feeling, Song Dedication expresses what you are feeling for your friends i.e. you can sing a song for your friends.

Miss-U:

This is another exciting feature that we have implemented to say your friend that you are missing them. They can miss you back in response to your miss you. To maintain the privacy, we don't reveal the identity of the person who is missing their friends.

Crush-Match:

On moving forward to simulate the online social structure, we can't avoid to explore the relationship. Basically this feature is for lovers. Crush match is another very unique feature created for people who cannot express their love to their crush. Using crush match they can add their love to their crush list. In that case his/her crush receives an anonymous email from Quipmate telling someone added her/him into crush list. Crush match happens when his/her crush adds back that person into his/her crush list. Meanwhile using personal messaging both of them can talk keeping the initiators identity hidden. To add your crush to crush list, enter your crush's email in the provide box. After you add your crush into your crush list, your crush will receive an anonymous email from Quipmate saying someone added her/him as crush at Quipmate. Person's identity will not be revealed in this email. This email will also contain a link which your crush will use to communicate with you using Quipmate's personal messaging system. You will find your crush's messages inside your inbox at Quipmate. You can use this thread to communicate with your crush. Meanwhile if your crush gets some hint about you, she/he will attempt adding you as crush at Quipmate. If she/he also adds you as crush at Quipmate that means she/he too has crush on you. It's a crush match. Now we reveal your identity now. Yours as well as your crush's friend will get an update about successful crush match.

Pin Board-A Photo Sharing tool:

It is said that a picture is worth of thousand words. Any SNS will look dead if it doesn't provide the image sharing. Pin Board is a photo sharing tool that displays the photos of people collectively that is shared with the member. It is another exciting way of representing the collection of photos. This tool also shows the comments made on those pictures and can also put comments on that. Members can also view the image of member individually by visiting their profiles.

Mood Sharing:

Mood sharing is the pictorial way of representing the way a member feels at particular time. In mood sharing, there's a list of emoticons that is used to represent the various expressions of human. Friends can also response to these shared posts.

Tagline:

Tagline is a short message that describes what drives a member for a particular span of time. It displays along with the each post of the members. Members can set their tagline by visiting their profiles.

Gift:

This is also a very exciting feature that simulates the trend of sending gifts to friends in a particular occasion. There is a set of gift i.e. is a virtual gifts or pictorial gifts members can send to their friends. This sharing is open i.e. this is visible to friends of sender as well as receiver.

Responses to the Post and Other Features:

Members can respond to the post posted by their friends by following using following three response types. These are:

1. Exciting
2. Sad
3. New-Pinch .

Exciting and Sad are the two responses for the each shared post/image posted by friends. These terms are self-explanatory. We choose these words because these words are only things we feel about any posts. We can also respond to any comment made on post/shared images by using "Exciting".

New-Pinch response is associated with the "profile picture change" action. Whenever any member changes his/her profile picture, friends of that member can respond to this action by new-pinching them because changing profile picture is analogous to changing clothes in offline life. New-Pinch is jargon among college friends that is used to give complements to the friends on their appearance.

There is also other feature that we have included in this SNS. One of the entertaining features of this SNS is "Playing Games". In this, two member of the Quipmate can compete with each other in the game. "Online Chat" is also part of any SNS. "Chat Rooms" is also incorporated in which members can create their own chat room and invite their friends to the room. Thus this provides the facility of chatting with more than one person. There are various applications in this SNS. The "Friend Finder Tool" let you search your friend in Quipmate world. If members have made mistakes then they can also delete their post, images. Search tool is also there to assist member to find the friends and Interests Diaries, school and colleges. We provide an extensive privacy on the information provided by the members of the Quipmate so that they can control their personal data.

3. Technologies Used in Quipmate:

3.1 Server Side Technologies: LAMP stack.

Linux Server:

Quipmate Server Architecture

Server 1:

Load-Balancer

50.57.190.112(Public)

Quipmate.com

Load Balancing Algorithm:

Weighted Round Robin Algorithm

Quipmatelead Server

10.5.12.32(Private)
PHP processing only
256 M RAM
10 GB disk space

Quipmate Server
Php + Database Server
512 M RAM
20 GB disc space
10.2.42.12

3.1.1 Linux Platform: Fedora 15.

3.2 Client Side Technologies: jQuery, JavaScript ,CSS, XHTML.

XHTML:
The doctype used in all are web document:-
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
This doctype supports fixed positioning in IE, a major problem in internet explorer.
Also the xml doctype which uses validation to validate the document, ensures that the document is flexible and deploys on a larger platforms.

CSS

1. Fixed positing in IE

Used
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> To get the fixed positioning supported in IE.

2. Negative Z-index issues:

When placing an element absolutely in between the document z-index issue props up. All the browsers stack elements positively that means higher the z-index of an element that element will be placed near to the user, but internet explorer does it in exactly reverse order.

3. Browser spreadsheet

Each browser has its own spreadsheet which means that it appends some margin and padding to the element from its own. Also they add border to the elements from their own.

3.3 Other Technologies Used: JSON(Java Script Object Notation), AJAX.

JavaScript Object Notation: These days JSON has proved to be very convenient way of transmitting data between the client and the server.

How we have used it at Quipmate?

1. Call any PHP script using a AJAX call from the jQuery script using the getJSON function and pass the parameters that the page will need to perform the manipulation.
2. At the server, get the required parameters into suitable variables, get the data from the database by instantiating the database object, perform the required manipulation and output the data using creating a proper format using json_encode.
3. The data obtained by the client script has a certain format, which it works upon by deploying it at the browser page, so it acts as a form of irrelevance to the fact that how the server is implemented and works fine if the data be output in that format.

3.3.3 Content Delivery Network:

A content delivery network (CDN) is a collection of web servers distributed across multiple locations to deliver content more efficiently to users. The server selected for delivering content to a specific user is typically based on a measure of network proximity. For example, the server with the fewest network hops or the server with the quickest response time is chosen. Creating several DNS pointing to the same IP helps browser create parallel connection with the server which implies even faster download of static content from the . Like script0.qm.net, script1.qmcdn.net, script2.qmcdn.net all pointing to single IP where all the script files of Quipmate are put. Quipmate uses the following CDNs:
script.qmcdn.net for downloading jQuery files.
style.qmcdn.net for downloading CSS files.
icon.qmcdn.net for downloading icons for the site.
song.qmcdn.net for directly streaming songs from the servers.

We can create containers, insert cloud files to containers, fetch objects, delete them programmatically using cloud files API(Application Programming Interface).

5. Configuration of Quipmate Platform:

5.1. Apache Web Server

General Information

Version:	
Executable name	/usr/sbin/httpd
Configuration File Location	/etc/httpd/conf/httpd.conf
ServerRoot	/etc/httpd

User	apache
Group	apache
ServerAdmin	root@localhost

Configurations

Configuration file:

ServerRoot	/etc/httpd
KeepAlive	On
<IfModule worker.c> MaxClients 300 <IfModule/>	This is used to describe the number of the simultaneous client connection.
Listen	80, this can be change to whatever port onto http request can be made.
LoadModule Module Location: /etc/httpd/modules	Load the Dynamic Shared Objects mod_deflate.so : The mod_deflate module provides the DEFLATE output filter that allows output from your server to be compressed before being sent to the client over the network. Compress content before it is delivered to the client AddOutputFilterByType DEFLATE text/html text/plain text/xml **mod_expires.so: To control browser caching** his module controls the setting of the Expires HTTP header and the max-age directive of the Cache-Control HTTP header in server responses. The expiration date can set to be relative to either the time the source file was last modified, or to the time of the client access. Generation of Expires and Cache-Control HTTP headers according to user-specified criteria ExpiresDefault "<base> [plus] {<num> <type>}*" ExpiresByType type/encoding "<base> [plus] {<num> <type>}*" For example, any of the following directives can be used to make documents expire 1 month after being accessed, by default: ExpireActive : -Enables generation of Expires headers ExpiresDefault "access plus 1 month" ExpiresDefault "access plus 4 weeks" ExpiresDefault "access plus 30 days"

	The expiry time can be fine-tuned by adding several '<num> <type>' clauses: ExpiresByType text/html "access plus 1 month 15 days 2 hours" ExpiresByType image/gif "modification plus 5 hours 3 minutes"
DocumentRoot	/var/www/html
Direcytory	/var/www/html

5.2 PHP Configuration:

display_errors: false
Output buffering : 4096
Max_ececution_time : 60
Memory_limit : 128 M
File_upload = on
Upload_max_filesize = 8M

Fopen wrappers:
Allow_url_fopen: on
Allow_url_include:off

Date
Date.timezone = UTC

Session
Session.use_cookies= 1
Session.name = PHPSESSID
Session.auto_start = 0
Session.cookie_lifetime = 0
Session.gc_probability = 1
Session.gc_divisor : 1000
Session.gc_maxlifetime=1440

5.3 Postfix Mail Server Configuration:

Mail Server: Postfix
Configuration:
Edit the file: /etc/postfix/main.cf
And append the following code
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = static:username:password
smtp_sasl_security_options = noanonymous
smtp_tls_security_level = may
header_size_limit = 4096000
relayhost = [smtp.sendgrid.net]:587

We use “service postfix start” command to start the email server and “service postfix stop” to stop the service.

We configured postfix for sending emails. We used the service of “sendgrid” which enabled us to send up to 40,000 emails per month. We have around 30 PHP email scripts which we used to send emails. The emails send the Quipmate users included the emails for:

1. Friend request
2. Email confirmation
3. Invitation email
4. Friendship confirmations
5. Comment on a post by a user
6. Any response on a post by a particular user
7. Any message to a user from any other user on the site
8. If anyone added you as your crush
9. If anyone missed you at Quipmate
10. Dedicated a song to you and many other things

5.4 Firewall configurations

1. Allowing outside connection from any remote client only on the following ports
   1.1 ports 80: for http requests
   1.2 port 22: SSH for configuring the sever using putty
   1.3 port 25: (smtp) postfix server for sending emails
   1.4 port 3306: remote connection to MYSQL server
2. Iptable command to enable remote access on port 3306

```
/sbin/iptables -A INPUT -i eth0 -s <serverip> -p tcp --destination-port
3306 -j ACCEPT
```

5.5 Remote Access to MySQL server:

root@quipmate:# vi /etc/my.cnf

```
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
language        = /usr/share/mysql/English
bind-address    = 50.57.190.112
# skip-networking
```

root@quipmate:# service mysqld restart

Allow remote access on the port 3306 used by mysql server using the iptable command listed above.

6. Quipmate Architecture:

6.1. Web Directory Architecture

6.1.1 Blocking and non-blocking web server:

Under normal circumstances apache acts as a blocking server i.e its keeps blocking a particular bowser from sending other request to the same web-server until the previous request is being full-filled. This is a great disadvantage when polling is needed to be implemented.

Polling:

Polling a mechanism of fetching the newly created data from the server based on certain conditions.

There are two kinds of polling mechanisms that are being used.

1. Short Polling:

In short-polling the client explicitly requests the server after a certain period of time for the new data, if the data is available at the server, the server responds with that data, otherwise the server script terminates and client again polls the server for the new data.

1.1 Short polling consumes a lot of resources and is an overhead.

1.2 It's not very-scalable.

Code for short polling:

```
<script>
    $.getJSON('server-script-url',{param1:param1,param2:param2,...},function(data){
});
</script>

<?php
call function();  <-  checkfor new data
?>
```

2. Long polling:

Long polling helps create server-side event, which ensures that whenever new data is available at the server the data will be fetched to the client, without client explicitly requesting it. The client just needs to subscribe to the server for the new data once. The script on the server which is requested by the client then just checks for the new data, if it is available it JSON encodes it and sends them to the client else it sleeps itself for a certain period of time. Again after that period the server script starts and repeats the whole steps.

2.1 Long polling is very-scalable.

2.2 It consumes minimum amount of resources.

Long polling is not possible under normal apache configurations as it blocks the current running thread. An implementation of tornado server or node.js is required to accomplish this.

6.1.2 Web-server Directives

6.1.2.1 Handling various HTTP ERROR CODES:

ErrorDocument 400 http://www.quipmate.com/welcome.php

ErrorDocument 401 /welcome.php
ErrorDocument 402 http://www.quipmate.com/welcome.php
ErrorDocument 403 http://www.quipmate.com/welcome.php
ErrorDocument 404 http://www.quipmate.com/welcome.php
ErrorDocument 500 http://www.quipmate.com/welcome.php

6.1.2.2 Preventing directory Listing

IndexIgnore *
Options -Indexes

6.1.2.3 Permanent Redirections(301)

3.1 Non-www to www redirection:

RewriteCond %{HTTP_HOST} !^www\.
RewriteRule ^(.*)$ http://www.%{HTTP_HOST}/$1 [R=301,L]

6.1.2.4 Domain redirection permanent

3.3.RewriteCond %{HTTP_Host} ^(www\.)?quipmate\.co\.in$ [NC]
RewriteRule ^(.*)$ http://www.quipmate.com/$1 [L,R=301]

RewriteCond %{HTTP_Host} ^(www\.)?quipmate\.in$ [NC]
RewriteRule ^(.*)$ http://www.quipmate.com/$1 [L,R=301]

RewriteCond %{HTTP_Host} ^(www\.)?allrumors\.in$ [NC]
RewriteRule ^(.*)$ http://www.quipmate.com/$1 [L,R=301]

3.4 I.P to domain redirections:

RewriteCond %{HTTP_HOST} ^50\.57\.190\.112
RewriteRule (.*) http://www.quipmate.com/$1 [R=301,L]

6.1.2.4 Enable caching mechanism with the help of htaccess

```
<IfModule mod_expires.c>
  ExpiresActive On
  ExpiresByType text/html M36000
  ExpiresByType text/css M36000
  ExpiresByType application/x-javascript M36000
  ExpiresByType image/bmp M36000
  ExpiresByType image/gif M36000
  ExpiresByType image/x-icon M36000
  ExpiresByType image/jpeg M36000
</IfModule>
```

6.1.2.5 Enable Gunzip compression

```
<IfModule mod_deflate.c>
  AddOutputFilterByType DEFLATE text/html text/css application/x-javascript text/plain
text/xml image/x-icon
</IfModule>
```

6.1.3. Robots.txt

About:

As site owners, what little control we have over what robots are allowed to do when they visit our sites exist in a magical little file called "robots.txt.

Web site owners use the /robots.txt file to give instructions about their site to web robots; this is called The Robots Exclusion Protocol.

It works like this: a robot wants to visits a Web site URL, say http://www.example.com/welcome.html. Before it does so, it firsts checks for http://www.example.com/robots.txt, and finds:

User-agent: *
Disallow: /

The "User-agent: *" means this section applies to all robots.
The "Disallow: /" tells the robot that it should not visit any pages on the site.

There are two important considerations when using /robots.txt:

- Robots can ignore your /robots.txt. Especially malware robots that scan the web for security vulnerabilities, and email address harvesters used by spammers will pay no attention.
- The /robots.txt file is a publicly available file. Anyone can see what sections of your server you don't want robots to use.

So don't try to use /robots.txt to hide information. Only goods spider use robots.txt

Where to put the robots.txt:

The short answer: in the top-level directory of your web server. (File name: robots.txt)

Example:

Let's get a little more discriminatory now. While every webmaster loves Google, you may not want Google's Image bot crawling your site's images and making them searchable online, if just to save bandwidth. The below declaration will do the trick:

User-agent: Googlebot-Image
Disallow: /

The Quipmate robots.txt:

User-agent: *
Disallow: /AJAX/
Disallow: /style/
Disallow: /script/
Disallow: /image/

6.2 Programming Architecture:

6.2.1 Incremental AJAX Loading

Most of the pages at Quipmate do not load all the content at once, rather it loads in the incremental order that means initially only some data is fetched from the database and later on based on the following two events the data is loaded into the page

1. Scroll down event:

Whenever user requests for the new data, i.e. when the scrolls down we assume that user is interested in more date so we make a call to the server script and fetch the data, this process repeats whenever the web-page scrolls to 60% of its size.

2. Explicitly request by the user for more data:

We also provide a button saying load more data which the user can use to load more content from the database.

In both the case we make request to the same server script just passing the current state of web-page so that further information can be brought from the server.

Universal loader:

Since most of the pages required require this mechanism of incrementally loading the data, we devised a single loading script which can request the server script based on the client state and the web-page which the client is viewing. It is combination of all the loading scripting which passes the server script as parameter and also the parameters required by that server script to perform the computations and fetch the result to the client.

Code:

```
$(function(){
var profileid = $('#id').val();
var myprofileid = $('#myprofileid_hidden').attr('value');
var page = $('#page_hidden').attr('value');
var load = false;
var param = {};
var url = '';
var increment = 0;
param.start = 0;
if(page == 'news_json')
{
	url = 'AJAX/news_json.php';
	increment = 10;
	$('#center').append('<div id="news_poll"></div>');
	$('#center').append('<div id="prev"></div>');
	$('#center').append('<input type="submit" id="load_more" style="heigth:30px;width:150px;margin-left:200px;display:none;" value="Show More Updates" />');
}
else if(page == 'quip')
{
	url = 'AJAX/news_json.php';
	increment = 10;
	param.quip = 'quip';
	$('#center').append('<div id="prev"></div>');
```

```
	$('#center').append('<input type="submit" id="load_more" style="heigth:30px;width:150px;margin-left:200px;display:none;" value="Show More Updates" />');
}
else if(page == 'notice_json')
{
	$('#links').remove();
	url = 'AJAX/news_json.php';
	var myprofileid = $('#myprofileid_hidden').attr('value');
	param.myprofileid = myprofileid;
	param.notice = 'notice';
	increment = 10;
	$('#center').append('<div id="prev"><h1 style="margin:1em 0em .5em 10em;font-size:1.4em;">All Notifications.</h1></div>');
	$('#center').append('<input type="submit" id="load_more" style="heigth:30px;width:150px;margin-left:200px;display:none;" value="Show More Notices" />');
}
else if(page == 'action')
{
	$('#links').remove();
	url = 'AJAX/news_json.php';
	var actionid = $('#actionid_hidden').attr('value');
	var life_is_fun = $('#life_is_fun_hidden').attr('value');
	param.actionid = actionid;
	param.life_is_fun = life_is_fun;
	increment = 10;
	$('#center').append('<div id="prev"></div>');
}
else if(page == 'profile_json')
{
	url = 'AJAX/news_json.php';
	var profileid = $('#profileid_hidden').attr('value');
	param.profileid = profileid;
	increment = 10;
	$('#center').append('<div id="prev"></div>');
	$('#center').append('<input type="submit" id="load_more" style="heigth:30px;width:150px;margin-left:200px;display:none;" value="Show More Diary Entry" />');
}
else if(page == 'album')
{
	url = 'AJAX/news_json.php';
	param.myprofileid = myprofileid;
	param.album = 'album';
	increment = 10;
```

```
	var profile_name = $('#myprofilename_hidden').attr('value');
	$('#center').remove();
	$('#left').remove();
	$('#right').remove();
	$('body').css('background','#E7EBF2');
	$('body').append('<div style="position:absolute;left:2em;top:5em;width:21em" id="tr0"></div><div style="position:absolute;left:24em;top:5em;width:21em" id="tr1"></div><div style="position:absolute;left:46em;top:5em;width:21em" id="tr2"></div><div style="position:absolute;left:68em;top:5em;width:21em" id="tr3"></div><div style="position:absolute;left:90em;top:5em;width:21em" id="tr4"></div>');
}
else if(page == 'photo')
{
	url = 'AJAX/photo_json.php';
	param.myprofileid = myprofileid;
	increment = 25;
	var profile_name = $('#myprofilename_hidden').attr('value');
	$('#center').append('<div id="prev"></div>');
	$('#center').append('<input type="submit" id="load_more" style="heigth:30px;width:150px;margin-left:200px;display:none;" value="Show More Photo" />');
	$('#prev').append('<h1 style="font-size:16pt;">'+profile_name+' -> Friends -> Photo</h1>');
	$('#prev').append('<table></table>');
}
else if(page == 'pphoto')
{
	url = 'AJAX/photo_json.php';
	param.profileid = profileid;
	increment = 25;
	var profile_name = $('#profilename_hidden').attr('value');
	$('#center').append('<div id="prev"></div>');
	$('#center').append('<input type="submit" id="load_more" style="heigth:30px;width:150px;margin-left:200px;display:none;" value="Show More Photo" />');
	$('#prev').append('<h1 style="font-size:16pt;">'+profile_name+' -> Photo</h1>');
	$('#prev').append('<table></table>');
}
else if(page == 'college_mate')
{
	url = 'AJAX/people_json.php';
	param.college = 'college';
	increment = 10;
	$('body').append('<div id="prev" style="position:relative;top:2em;"></div>');
}
else if(page == 'new_user')
```

```
{
	url = 'AJAX/people_json.php';
	param.new_user = 'new_user';
	increment = 10;
	$('body').append('<div id="prev" style="position:relative;top:2em;"></div>');
}
else if(page == 'friend')
{
	url = 'AJAX/friend_json.php';
	param.profileid = profileid;
	param.myprofileid = myprofileid;
	increment = 50;
	var profile_name = $('#profilename_hidden').attr('value');
	$('#center').append('<div id="prev"></div>');
	$('#center').append('<input type="submit" id="load_more" style="heigth:30px;width:150px;margin-left:200px;display:none;" value="Show More Friends" />');
	$('#prev').append('<h1 style="font-size:16pt;">'+profile_name+' - Friends</h1>');
	$('#prev').append('<table></table>');
}
else if(page == 'inbox')
{
	var profileid = $('#profileid_hidden').attr('value');
	var myprofileid = $('#myprofileid_hidden').attr('value');
	var profileid_name = $('#profilename_hidden').attr('value');
	var myprofileid = $('#myprofileid_hidden').attr('value');
	var myprofileid_name = $('#myprofilename_hidden').attr('value');
	var myprofileid_image = $('#myprofileimage_hidden').attr('value');
	url = 'AJAX/message_json.php';
	param.profileid = profileid;
	increment = 10;
	$('#center').append('<div id="prev"></div>');
	if(myprofileid != profileid)
	{
	$('#center').prepend('<h1 style="margin-bottom:-35px;">All Messages between '+profileid_name+' and you</h1>');
	$('#center').prepend('<div id="message_container"><h1>Drop a Message For '+profileid_name+'</h1><div style="position:relative;left:5em;"><textarea id= "message_textarea" ></textarea><input id="drop_button" type="submit" value="Drop"></div><div style = "color:gray;margin-top:15px;margin-bottom:-10px;">( The message will also be sent to '+profileid_name+' \'s email )</div></div>');
	}
	else
	{
	$('#center').prepend('<h1 style="margin-bottom:-35px;">All Messages</h1>');
	}
```

```
}
$.getJSON(url,param,function(data){
  load = true;
        switch(page)
        {
                case  'news_json':  $('#max_actionid_hidden').attr('value',data.action[0].actionid);
news_deploy(data,'#prev'); break;
                case 'quip': news_deploy(data,'#prev'); break;
                case 'notice_json': notice_deploy(data,'#prev'); break;
                case 'action': news_deploy(data,'#prev'); break;
                case 'profile_json': news_deploy(data,'#prev'); break;
                case 'album': album_deploy(data); break;
                case 'photo': photo_deploy(data); break;
                case 'pphoto': photo_deploy(data); break;
                case 'college_mate':people_deploy(data); break;
                case 'new_user':people_deploy(data); break;
                case 'friend':friend_deploy(data); break;
                case 'inbox':message_deploy(data); break;
                default: $("#prev").html(data); break;
        }
var oldh = $("#prev").height();
oldh = parseInt(oldh) + 300;
$("#center").height(oldh);
param.start += increment;
$('#load_more').show();
});

$(window).scroll(function(){
var scroll_size = $(document).height() - $(window).height();
scroll_size = scroll_size * 0.6;
if(($(window).scrollTop() > scroll_size) && load)
{
  load = false;
$('#load_more').hide();
$("#prev").append('<p align="center"><img src="png/loading.gif" id="loading"></p>');
$.getJSON(url,param,function(data){
if($.trim(data)!='')
{
 load = true;
$("#loading").remove();

        switch(page)
        {
                case 'news_json': news_deploy(data,'#prev'); break;
                case 'quip': news_deploy(data,'#prev'); break;
                case 'profile_json': news_deploy(data,'#prev'); break;
```

```
                case 'notice_json': notice_deploy(data,'#prev'); break;
                case 'action': news_deploy(data,'#prev'); break;
                case 'album': album_deploy(data); break;
                case 'photo': photo_deploy(data); break;
                case 'pphoto': photo_deploy(data); break;
                case 'college_mate':people_deploy(data); break;
                case 'new_user':people_deploy(data); break;
                case 'friend':friend_deploy(data); break;
                case 'inbox':message_deploy(data); break;
                default: $("#prev").append(data); break;
        }
var oldh = $("#prev").height();
oldh = parseInt(oldh) + 300;
$("#center").height(oldh);
param.start += increment;
$('#load_more').show();
}
else
{
        load = false;
        $('#loading').remove();
        $('#prev').append('<p align="center">No More Updates From Your Friends</p>');
}
});
}
});

$('#load_more').click(function(){
$('#load_more').hide();
if(load)
{
   load = false;
$("#prev").append('<p align="center"><img src="png/loading.gif" id="loading"></p>');
$.getJSON(url,param,function(data){
if($.trim(data)!='')
{
  load = true;
$("#loading").remove();
var oldh = $("#center").height();
        switch(page)
        {
                case 'news_json': news_deploy(data,'#prev'); break;
                case 'quip': news_deploy(data,'#prev'); break;
                case 'profile_json': news_deploy(data,'#prev'); break;
                case 'notice_json': notice_deploy(data,'#prev'); break;
                case 'action': news_deploy(data,'#prev'); break;
```

```
                case 'album': album_deploy(data); break;
                case 'photo': photo_deploy(data); break;
                case 'pphoto': photo_deploy(data); break;
                case 'college_mate':people_deploy(data); break;
                case 'new_user':people_deploy(data); break;
                case 'friend':friend_deploy(data); break;
                case 'inbox':message_deploy(data); break;
                default: $("#prev").append(data); break;
        }
        var oldh = $("#prev").height();
        oldh = parseInt(oldh) + 300;
        $("#center").height(oldh);
        param.start +=increment;
        $('#load_more').show();
}
else
{
        load = false;
        $('#loading').remove();
        $('#prev').append('<p align="center">No More Updates From Your Friends</p>');
}
});
}
});
```

6.3. Storing session data into database

By default PHP stores session variables in files. But this can be overridden and session data can be stored in database instead of files. There are six - functions used by the session management module of PHP. The following functions are written below and these are overridden in the following way:-

Top reasons for storing data into the database:

1. To remove server affinity i.e. to be able to use more than one server. Store the session data into a database connected centrally to all the servers, so that user has session even if he is redirected to the other server.
2. Enables the concept of load balancer to be implemented.
3. Storing session in database is more secure than storing them in files.

We have done this using the following codes:

Code:

```
1. CREATE TABLE sessions (
2. id varchar(32) NOT NULL,
3. access int(10) unsigned,
4. data text,
```

5. PRIMARY KEY (id)
6.);

```
mysql> DESCRIBE sessions;
+--------+-----------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+-----------------+------+-----+---------+-------+
| SESSIONID | varchar(32) | | PRI | | |
| TIME | int(10) unsigned | YES | | NULL | |
| DATA | text | YES | | NULL | |
+--------+-----------------+------+-----+---------+-------+
```

```
<?php
session_set_save_handler('_open','_close','_read','_write','_destroy','_clean');
session_start();
//require_once('../include/initialize_session_variables.php');
echo $_SESSION['NAME'];
$_SESSION['NAME'] ='00000000';
function _open(){}
function _close(){}
function _read($sessionid)
{
        $con = new mysqli('ip','user','password','database');
        $result = $con->query("SELECT DATA from session where SESSIONID =
'$sessionid'");
        $result =  $result->fetch_array();
        return $result['DATA'];
}
function _write($sessionid,$data)
{
        $con = new mysqli('ip','user','password','database');
        $time = time();
        $result = $con->query("REPLACE INTO session(SESSIONID,TIME,DATA)
        VALUES('$sessionid','$time','$data')");
        return $result;
}
function _destroy($sessionid)
{
        $con = new mysqli('ip','user','password','database');
        $result = $con->query("delete from session where SESSIONID = '$sessionid' ");
        return $result;
}
function _clean($time)
{
        $con = new mysqli('ip','user','password','database');
        $result = $con->query("delete from session where TIME < '$time' ");
```

```
        return $result;
}
?>
```

6.3 .2 Screen Scrapping

Screen scrapping involves getting the various tags of the webpage and selecting the required tag out of those. But a direct request to that file cannot be made to such a web-page as browsers do not permit that for security reasons, so a call need to be made to the Quipmate server passing as parameter the file to be screen scrapped and then performing the screen scrapping in PHP. The result obtained can then be JSON encoded and sent back to the client and output displayed to the user. The data is then provided to the user which he/she can edit and then again those parameters can be passed to the server and stored in database.

The PHP code for screen scrapping is as follows:

```
        if(isset($_GET['link']))
        {
                $link = $_GET['link'];
                $http_link = explode('://',$link);
                if($http_link[0] != 'http' && $http_link[0] != 'https')
                {
                        $link = 'http://'.$link;
                }
                $dom = new DOMDocument();
                $data = array();
                $src = array();
                if(@$dom->loadHTMLFile($link))
                {
                        $list = $dom->getElementsByTagName("title");
                        if ($list->length > 0)
                        {
                                $data['title'] = $list->item(0)->textContent;
                        }
                        $metas = $dom->getElementsByTagName("meta");
                        foreach ($metas as $meta)
                        {
                                if($meta->getAttribute('name') == 'description')
                                {
                                        $data['meta'] = $meta->getAttribute('content');
                                }
```

```
                }
                $parse = parse_url($link);
                $data['host'] = $host = 'http://'.$parse['host'].'/';
                if($host == 'http://youtu.be/')
                {
                        $data['video'] = 1;
                        $path = explode($host,$link);
                        $data['path'] = $path[1];
                }
                else if($host == 'http://www.youtube.com/')
                {
                        $data['video'] = 1;
                        $path = explode('v=',$link);
                        $path = explode('&',$path[1]);
                        $data['path'] = $path[0];
                }
                else
                {
                        $data['video'] = 0;
                        $tags = $dom->getElementsByTagName("img");
                        $valid_formats = array("jpg","png","gif","bmp","jpeg","cms");
                        foreach ($tags as $tag)
                        {
                                $img = $tag->getAttribute('src');
                                $ext = pathinfo($img,PATHINFO_EXTENSION);
                                if(in_array($ext,$valid_formats))
                                {
                                        if(strpos($img,'http://') === false)
                                        {
                                                $img = $host.$img;
                                        }
                                                $src[] = $img;
                                }
                        }
                        $data['src'] = $src;
                }

                echo json_encode($data);
        }
}
```

7. Security Aspects:

7.1 Use of HTACCESS Files:

Direct Calls to AJAX Files:

This can be a source of security venerability point in a site which uses AJAX for fetching the data. The AJAX files which are called by the jQuery code can be easily seen by viewing the source code of the web-page on the browser.

To prevent this we implemented a mechanism in which we grouped the files of a certain kind together and put them in single folder. So all the AJAX files were put in one folder and using htaceess we prevent any direct call to any file in that particular folder.

HTACCESS code:

```
SetEnvIfNoCase X-Requested-With XMLHttpRequest ajax
Order Deny,Allow
Deny from all
Allow from env=ajax
```

Direct Calls to include files

We can use php file including including the same files on various doc-root pages. This is very helpful in a way, but if the included files are all located in the document root only or with a sub-folder of the document root that can be called directly by the user which can expose potential errors displaying the information about the scripts.

7.2 SQL-Injection:

1. It can be easily handled by escaping the parameters before they are actually passed to the SQL query.

Code:

2. And PHP strip slashes to output the data after fetching from the database.

Code:

7.3 Cross-site Scripting:

1. Cross site scripting uses cookies to transfer data from one site to the the other.
2. Cross site can be easily used to attack websites which uses cookies to store user data.

How we took care of it:

1. Do not store any sensitive user-information in the cookie like password.
2. Use PHP strip-tags to strip-off the tags like script which makes CSS possible.

8. Scalability

1. Number of Database connections:

The number of connection by the web-server to the database server is most important issue which concerns scalability. So to make application scalable the first thing to keep in mind is to have lower number of connections to the database server.

In a social network people are the real world entity around which everything evolves, the two most used information are the name and profile picture of people, generally those in the friend-list.

1.1 Name of a user

1.2 Profile-image of the user

So the first step to reduce the number of database connection is to develop some kind of caching mechanisms and store both name and profile picture in that.

How we solve the problem:

8.1 Session storage:

1. Store the name and profile-picture of a particular user in the session of each user.
2. Use associative array to store the information into the session.

For example, the profileid of a particular user is: 1000000122 then his name is available at
$_SESSION['name_json']['1000000122'] and his profile image is available at
$_SESSION['pimage_json']['1000000122'] and his tagline is available at
$_SESSION['tag_json']['1000000122']

3. Whenever these information are required by a PHP script first look for those information in the session variables, if they are available fetch these information from the session variables else fetch them from the database and store them in the session variables. This mechanism has proved to be of a significant use in making Quipmate faster.

8.2. Perform heavy MYSQL operations like join in PHP:

Php works much faster when compared to MySQL over any amount of data manipulation.
So heavy amount of data-manipulation can be performed in PHP rather in MySQL when the size of table grow over lakhs of tuples and more than one tables need to worked upon.
We developed a mechanism to perform the MySQL join-operation in PHP and fetch the result in the associative array.
For example, the procedure to perform join of friend table with the action table to obtain the actions performed the friends of the person in whose session is running.

STEP 1: select all the friends of that person using a database call

```
$r = $database->friend_select($profileid);
$i=0;
while($row = $r->fetch_array())
{
$friendid[$i] = $row['FRIENDID'];
$i++;
```

```
}
```

STEP 2: select all the actions from the action table using a database call

```
$r = $database->action_select($profileid);
$j=0;
while($row = $r->fetch_array())
{
$actionid[$j] = $row['ACTIONID'];
$profieid[$j] = $row['PROFILEID'];
$actionby[$j] = $row['ACTIONBY'];
$j++;
}
```

STEP 3: Check for the condition of equality and put them in a new array

```
while($k)
{
while()
{
If($friendid[$p] ==$actionby[$q] || $friendid[$p] ==$profileid[$q])
{
        $join['actionid'][$r] = $actionid[$q];
        $join['actionby'][$r] = $actionby[$q];
        $join['profileid'][$r] = $profileid[$q];
}
}
}
```

9. Conclusion:

The project Quipmate (http://www.quipmate.com/) is huge in terms of the number of lines of codes written. This technical report only tried to sum up the background technologies and tools used by us during the development process.

The project has currently 146 running PHP scripts, 40 javascripts and 15 stylesheets. Working on it since a period of 1.3 years we have released three versions of Quipmate optimizing the codes in each next version and redesigning and modifying the base architectures each time to make it easy to extent and scale. Every day while working at Quipmate we have tried to innovate and differentiate not only at the social fronts but as well as on the technical and code fronts. Our core ambition throughout the process has been to use computer science in the best possible way to develop social features.

10. References:

Internet through Google search.

11. Quipmate Developer Team

Kunal Kumar Singh
Brijesh Kushwaha
Dheraj Nigam
Prashant Agarwal

B.Tech Computer Science at MMMEC - Gorakhpur