ABSTRACT

The project entitled “S-Net – The People’s Network”, is a web application developed for an intra-institute (college) usage. The idea behind, “S-Net”, is to create a network of people; that is, a network of former students, a.k.a. alumni and undergraduates. The aim of this project is to provide and deliver access to anyone who’s authorized, at anyplace and anytime on any device and also provide relevant access to information and helps them to develop and maintain interpersonal relationships. It also provides them with a platform to share their ideas, theories, or any other discoveries that helps them to grow within and without.

Table of Contents

ACKNOWLEDGEMENT……………………………………………………………..v

ABSTRACT……………………………………………………………………..vi

LIST of FIGURES……………………………………………………………….ix

LIST of TABLES……………………………………………………………..xi

1. INTRODUCTION……………………………………………………….....1

    1.1. Motivation……………………………………………………………...1

    1.2. Aim of Software…………………………………………………….....2

    1.3. Document Organization…………………………………………….....2

2. OBJECTIVES…………………………………………………………….3

    2.1. Product Perspective…………………………………………………..3

    2.2. Product Functions…………………………………………………….4

    2.3. User Classes and Characteristics…………………………………….4

    2.4. Constraints…………………………………………………………...5

    2.5. Assumptions and Dependencies……………………………………...5

    2.6. Specific Requirements……………………………………………….5

    2.7. Design Constraints…………………………………………………...7

    2.8. Software System Quality Attribute…………………………………..7

    2.9. Programming Environment…………………………………………..8

3. IMPLEMENTATION……………………………………………………..21

    3.1. Detailed Scope………………………………………………………...21

    3.2. Static Decomposition and Dependency Description……………………..21

    3.3. The Database Design…………………………………………………35

    3.4. Interface Description…………………………………………………..39

4. TESTING…………………………………………………………………..50

    4.1. Methodology…………………………………………………………..50

    4.2 Interface Testing………………………………………………………51

Table of Contents

5. CONCLUSION and FUTURE WORK…………………………………………….55

5.1. Conclusion……………………………………………………………….55

5.2. Future Work…………………………………………………….............56

5.3. Project Summary…………………………………………………………56

BIBLIOGRAPHY………………………………………………………………57

Appendix: Sample Code………………………………………………………...58

LIST of FIGURES

Figure No.	Description	Page No.
2.1	3-Tier Architecture	09
2.2	MVC and 3-Tier Architecture	09
2.3	Logical View of MySQL’s Architecture.	12
2.4 (a)	Synchronous Web Application Model	17
2.4 (b)	Synchronous Web Application Model	18
2.5 (a)	AJAX Web-Application Model	18
2.5 (b)	AJAX Web-Application Model	19
2.6	How Ajax Works?	19
3.1	S-Net Use Case Diagram	22
3.2	S-Net Class Diagram	25
3.3	Sequence Diagram for Login	28
3.4	Sequence Diagram for Signup	29
3.5	Sequence Diagram for Creating a Group	30
3.6	Sequence Diagram for Viewing Alumni Records	31
3.7	Sequence Diagram for Logout	32
3.8	0 Level DFD	34
3.9	1 Level DFD	34
3.10	Entity Relationship Diagram for S-Net	35
3.11	Index Page – S-Net	40
3.12	Signup Page – S-Net	40
3.13	Successful Signup – S-Net	41
3.14	Login Page – S-Net	41
3.15	Profile Page – S-Net	41
3.16	Settings Page – S-Net	42
3.17	Groups Page – S-Net	42
3.18	Create New Group Page – S-Net	43
3.19	Group Created Page – S-Net	43
3.20	Group Join Page – S-Net	44
3.21	Group’s Home Page – S-Net	44

LIST of FIGURES

Figure No.	Description	Page No.
3.22	Alumni/Friend Search Page – S-Net	44
3.23	Search Result for Alumni/Friend – S-Net	45
3.24	Friend Request Option – S-Net	45
3.25	Friend Request Being Accepted – S-Net	45
3.26	Posting on Users Wall – S-Net	46
3.27	Replying Back to User Post – S-Net	46
3.28	Photo Gallery Option – S-Net	47
3.29	Upload Photo Option – S-Net	47
3.30	A Photo Uploaded into the System – S-Net	48
3.31	Enlarged View of the Photo Uploaded– S-Net	48
3.32	Uploaded Photo Being Deleted – S-Net	49
3.33	Uploaded Photo is Removed from the System – S-Net	49

LIST of TABLES

Table No.	Description	Page No.
3.1	Structure of Users Table	36
3.2	Structure of User Options Table	36
3.3	Structure of Status Table	37
3.4	Structure of Photos Table	37
3.5	Structure of Notifications Table	37
3.6	Structure of Groups Table	38
3.7	Structure of Group Post Table	38
3.8	Structure of Group Members Table	38
3.9	Structure of Friends Table	39
3.10	Structure of Blocked User Table	39
4.1	Functional Requirement List	52
4.2	Test Case Table with Description	52
4.3	Test Cases and Their Respective Results.	54

1. INTRODUCTION

Students, past and present, are what any institute should be all about. Faculty and staff are vital as well, but students are unique in that they will forever fill a role – whether they are current students or long graduated. It would be easy to lose track of most graduates after they embark on their various careers, but doing so would be a huge mistake on any institute's part.

Why? They create the institute's reputation, which relies in large part on how successful graduates are in the real world. This process is self-feeding as well. If a school becomes well known for producing graduates that are intelligent, innovative, and effective in their fields, then its reputation will grow. New graduates will have better job prospects because they went to a well-known school, and the process will continue.

At Swami Vivekananda Institute of Technology, the same need has come along since the plans for expansion came into being. This needled to developing the "S-Net – The People's Network", a.k.a. a Community of undergraduates and former students.

1.1. Motivation

Being in such a rapid and fast paced environment requires a Quality Assurance Specialist who takes care of the quality of the records of graduates produced by an institute. Alumni also help sustain the university through donations and volunteerism. It costs a lot of money to keep an institute on its feet, and a big chunk of that comes from alumni donations. Sometimes, large donations allow institutes to provide students with newer technology, better facilities, and a nicer campus—which in turn keeps institutes competitive and appealing.

A key factor in an institute rankings is the amount of alumni participation that an institute has. This isn't surprising, considering an active alumni base generally indicates that past graduates feel like they had a quality educational experience. Attending a school with very active alumni also often encourages current students to remain involved and active after graduation, once more perpetuating a good cycle.

1.2. Aim of Software

This software is developed in order to help everyone at SVIT. S-Net allows the users to view and bridge the gap between former students and undergrads. S-Net, being built for Swami Vivekananda Institute of Technology will have the scope of being extended for additional features in the future as needed.

1.3. Document Organization

The rest of the document is divided into three parts: OBJECTIVE, IMPLEMENTATION, and TESTING. The chapter entitled “Objectives”, lists the need for building the system. It also gives a detailed explanation of each constraint to help with design and implementation, and outlines the constraints regarding the software along with the Environment where the application will be developed. The chapter entitled “Implementation” contains the detailed design of the system, including the Use Case Diagram, Class Diagram, Data Flow Diagram, and Interaction Diagram. The chapter also includes a detailed explanation for each component as well as the interaction of each class and its components with each other when carrying out certain tasks. This chapter also includes the software’s simulated screen shots.

2. OBJECTIVES

This chapter gives the overall description of the S-Net – The People's Network. Then, we address the specific product requirements. The document has details about the Product Function, User Characteristics, Constraints, and Assumptions and Dependencies and Programming Environment for the S-Net.

2.1. Product Perspective

S-Net – The People's Network is a web-based system and can be accessed using: Internet Explorer8.0 and above, Mozilla Firefox 2.0, and Google Chrome.

2.1.1. User Interface

The two interface types in the Community Driven System are as follows:

1. **User Interface**: Users shall be able to view the alumni records, create groups, post or share ideas, seek help from other users of the system.

2. **Community Admin Interface**: The community admin shall be able to perform accept group join requests, remove members who violate the group terms.

2.1.2. Hardware Interface

The Community Driven System shall provide minimum hardware requirements. The following hardware configurations are required for a PC using the Community Driven System a.k.a. S-Net:

- Processor – Intel Pentium 4 or above, or AMD AM3+.
- RAM - 4GB.
- HDD - 500GB.

2.1.3 Software Interface

The proposed software solution, uses Open Software products for software interfaces: PHP, Apache Httpd Server and MySQL Database Server Community Edition.

2.2 Product Functions

The following are the product's features/functionalities:

- User Registration.
- Controlled Access to Resource Information.
- Group Operations.
- Customizability and Extensibility.

2.3 User Classes and Characteristics

The users of the Community Driven System, based on their roles, are Students (Users) and the Group Administrators (Users). These users are identified based on their experience and technical expertise. All the above mentioned users should know the language English, and basic operations of a computer and should be familiar with web browsers.

- **Group Admins:** The Group Admin is the Community's employed Quality Assurance Specialist and head. He or she must have a basic understanding of computers and the internet. He or She should have some prior knowledge in task assignment and standard SVIT operating practices. The Admin would be responsible for maintaining the decorum of the system. The Admin should be able to perform the following functions:
    - Accept/reject group join requests.
    - Add a new administrator to the group.
- **Users:** The users of the system are the Student, Employees (Staff). They must have basic understandings of computers and the internet. They should have some prior knowledge in policy/guideline creation. The users should be able to perform the following functions using this system:
    - Post articles.
    - Develop interpersonal relationships.

2.4. Constraints

- **Hardware Limitations:** The minimum hardware requirement for the system would be 2 GB of RAM and a 250 GB HDD.
- **Regulatory Policies:** For implementing any system in Swami Vivekananda Institute of Technology, proper legal permission has to be obtained from the Board of Directors and the IT department. If approval is not granted, a desired system cannot be implemented.
- **Accessibility:** Initially, the software should be available as a desktop application for a small set of users to test.
- **Others:** The application should be built using Open Source Software and should be accessible through the World Wide Web.

2.5. Assumptions and Dependencies

The assumptions and dependencies are as follows:

- The system is completely dependent on the availability of an Internet connection.
- We assume that the IT department has enough disk space to store all the documents.
- We assume that users of the system adhere to the system's minimum software and hardware requirements.

2.6. Specific Requirements

This section contains details about all the software requirements that will be sufficient for designers to create a system to satisfy those requirements and for testers to test the given requirements. This section contains the interface description of each GUI for the different users involved with the system. This section also gives a description of all the system inputs, all the functions performed by the system, and all the output (responses) from the system.

2.6.1 Functional Requirements

This section contains the requirements for S-Net – The People's Network. The Functional Requirements, as collected from the users, have been categorized to support the types of user interactions the system shall have.

- **Login to the system:** The system shall recognize the user based on the login information (i.e., username and password), and based on the user's role, the system will show a different interface.
    1. **FR 01:** The Community Driven System shall have two types of access/login: a) Users, b) System Administrators.
    2. **FR 02:** The Community Driven System shall only be accessible to specified Users and Admin with a valid username/password.
- **User: Add Friends:** The users shall be able to see their designated page after login and to select the view friends from the tab interface. The user shall be able to view their friend list.
    1. **FR 03:** The users shall be able to view and add/block friends from the list.
- **User: View Groups:** After logging into the system, the users shall be able to see their designated pages. After viewing their profile, the users should be able to view the number of groups/communities available in the system. The users shall be able to view the groups.
    1. **FR 04:** The user shall be able to view the groups added to the system
    2. **FR 05:** The user shall be able to create a new group/community.
    3. **FR 06:** The Group Admin shall be able to accept/reject group join requests.
- **System Admin: Carry out cron jobs**
    1. **FR 07:** The Sys. Admin. shall be able to view all the users registered in the system, and execute cron jobs as intervals.
- **User: Edit Profile:** After logging into the system, the users shall be able to see their designated pages. The system shall allow all the users to edit their personal profiles.
    1. **FR 08:** The system allows the users to edit their personal profiles.

2.6.2 Performance Requirements

This section would list the Performance Requirements expected from the system while in using within the College.

1. **PR 01:** The user shall be able to login to the system in less than 4 seconds.
2. **PR 02:** The navigation between pages shall take less than 5 seconds.

2.7 Design Constraints

This section would list the design requirements to be followed by the system as part of the Software development guidelines.

- **DC 01:** If the logged in user does not perform any operation for more than 30 minutes, the system shall guide the user to the login page to re-login to the system.
- **DC 02:** The User Interface (UI) must have specific fonts and font sizes. The System shall match the fonts and font sizes used in all applications.
- **DC 03:** The UI shall have a help section to guide users (Admin and users).
- **DC 04:** The system shall ask users for the location of the file from which they want to browse the images for upload and the location to which they would like to save the file.

2.8. Software System Quality Attribute

- **Integrity:**
    1. **QA 01:** The authorized user shall be allowed to access the S-Net.
    2. **QA 02:** Based on the type of user, S-Net shall provide a user-specific interface.
- **Correctness:**
    1. **QA 03:** The response should be directed to initiator of the request.
- **Availability:**
    1. **QA 04:** The system shall be made available to the Users year round.
- **Robustness:**
    1. **QA 05:** The system shall be able to save the all the alumni records. If the database server goes down, the system should generate an error message for the users.

2.9 Programming Environment

This section details about the software and architecture that were used to develop the application.

2.9.1 The Three Tire Architecture

S-Net is based on the concepts of MVC Architecture**.** In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client-server architecture in which the presentation, the application processing, and the data management are logically separate processes.

A "tier" can also be referred to as a "layer". A wedding cake is said to have tiers while a chocolate cake is said to have layers, but they mean the same thing.

2.9.1.1 In the software world Tiers/Layers should have some or all of the following characteristics:

- Each tier/layer should be able to be constructed separately, possibly by different teams of people with different skills.
- Several tiers/layers should be able to be joined together to make a whole "something".
- Each tier/layer should contribute something different to the whole. A chocolate layer cake, for example, has layers of chocolate and cake.
- There must also be some sort of boundary between one tier and another. You cannot take a single piece of cake, chop it up into smaller units and call that a layer cake because each unit is indistinguishable from the other units.
- Each tier/layer should not be able to operate independently without interaction with other tiers/layers.
- It should be possible to swap one tier/layer with an alternative component which has similar characteristics so that the whole may continue functioning.

2.9.1.2 3-Tier Architecture, as shown in Figure 2.1 and Figure 2.2.

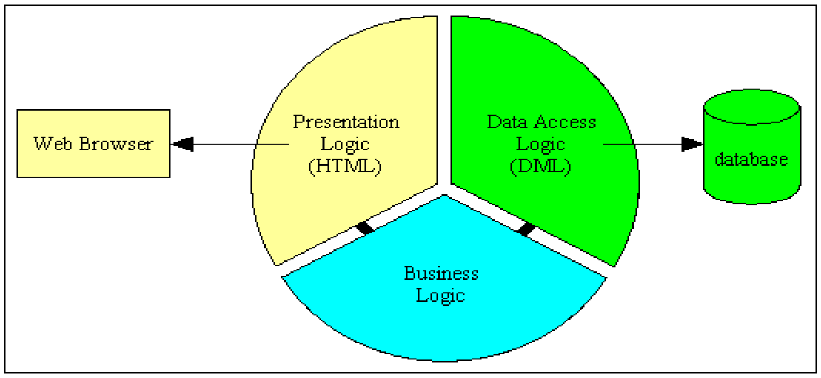


Figure 2.1: 3-Tier Architecture.

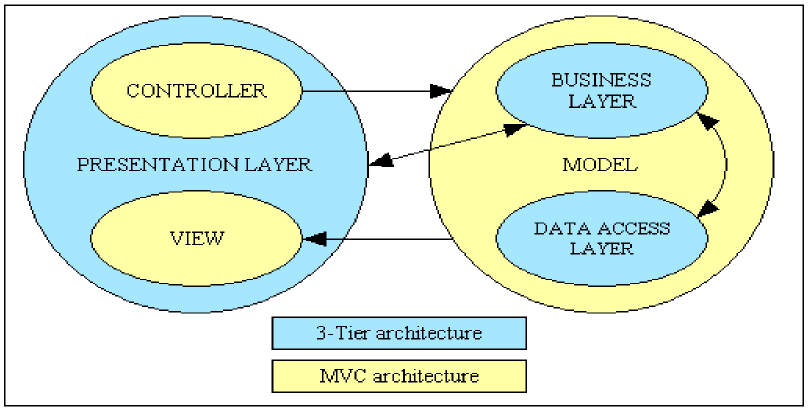


Figure 2.2: MVC and 3-Tier Architecture.

2.9.1.3 The main advantages of the 3 Tier Architecture are often quoted as:

- **Flexibility -** By separating the business logic of an application from its presentation logic, a 3-Tier architecture makes the application much more flexible to changes.
- **Maintainability -** Changes to the components in one layer should have no effect on any others layers. Also, if different layers require different skills (such as HTML/CSS is the presentation layer, PHP/Java in the business layer, SQL in the data access layer) then these can be managed by independent teams with skills in those specific areas.
- **Reusability -** Separating the application into multiple layers makes it easier to implement re-usable components. A single component in the business layer, for example, may be accessed by multiple components in the presentation layer, or even by several different presentation layers (such as desktop and the web) at the same time.

- **Scalability -** A 3-Tier architecture allows distribution of application components across multiple servers thus making the system much more scalable.
- **Reliability -** A 3-Tier architecture, if deployed on multiple servers, makes it easier to increase reliability of a system by implementing multiple levels of redundancy.

2.9.2 PHP

PHP, which stands for "*PHP: Hypertext Preprocessor*" is a widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but you can do much more with PHP.

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was.

PHP can be used on all major operating systems, including Linux, many UNIX variants, Microsoft Windows, Mac OS X, RISC OS, and probably others. PHP has also support for most of the web servers today. This includes Apache, IIS, and many others. PHP works as either a module, or as a CGI processor.

2.9.2.1 There are three main areas where PHP scripts are used.

- **Server-side scripting.** This is the most traditional and main target field for PHP. You need three things to make this work. The PHP parser (CGI or server module), a web server and a web browser. You need to run the web server, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server. All these can run on your home machine if you are just experimenting with PHP programming.
- **Command line scripting.** You can make a PHP script to run it without any server or browser. You only need the PHP parser to use it this way. This type of usage is ideal for scripts regularly executed using cron (on *nix or Linux) or Task Scheduler (on Windows). These scripts can also be used for simple text processing tasks.

- **Writing desktop applications.** PHP is probably not the very best language to create a desktop application with a graphical user interface, but if you know PHP very well, and would like to use some advanced PHP features in your client-side applications you can also use PHP-GTK to write such programs. You also have the ability to write cross-platform applications this way. PHP-GTK is an extension to PHP, not available in the main distribution.

2.9.2.2 Features

- HTTP authentication with PHP.
- Cookies.
- Sessions.
- Dealing with XForms.
- Handling file uploads.
- Using remote files.
- Connection handling.
- Persistent Database Connections.
- Safe Mode.
- Command line usage — Using PHP from the command line.
- Garbage Collection.
- DTrace Dynamic Tracing.

2.9.3 MySQL

MySQL is very different from other database servers, and its architectural characteristics make it useful for a wide range of purposes as well as making it a poor choice for others. MySQL is not perfect, but it is flexible enough to work well in very demanding environments, such as web applications. At the same time, MySQL can power embedded applications, data warehouses, content indexing and delivery software, highly available redundant systems, online transaction processing (OLTP), and much more.

MySQL's most unusual and important feature is its storage-engine architecture, whose design separates query processing and other server tasks from data storage and retrieval. This separation of concerns lets you choose how your data is stored and what performance, features, and other characteristics you want.

2.9.3.1 MySQL's Logical Architecture

A good mental picture of how MySQL's components work together will help you understand the server. Figure 3 shows a logical view of MySQL's architecture.

The topmost layer contains the services that aren't unique to MySQL. They're services most network-based client/server tools or servers need: connection handling, authentication, security, and so forth.

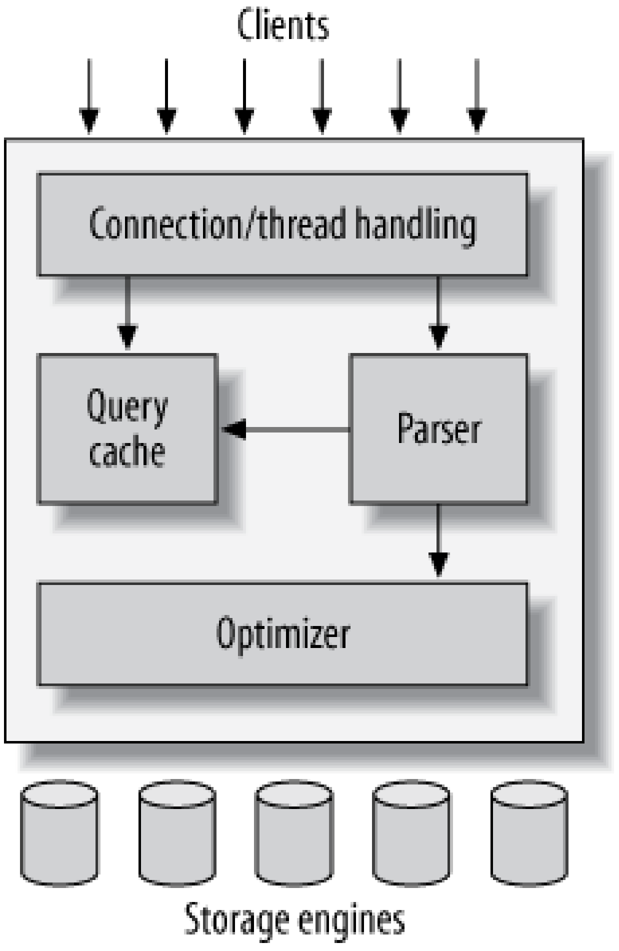


Figure 2.3: Logical View of MySQL's Architecture.

The **second layer** is where things get interesting. Much of MySQL's brains are here, including the code for query parsing, analysis, optimization, caching, and all the built-in functions (e.g., dates, times, math, and encryption). Any functionality provided across storage engines lives at this level: stored procedures, triggers, and views.

The **third layer** contains the storage engines. They are responsible for storing and retrieving all data stored "in" MySQL. Like the various file systems available for GNU/Linux, each storage engine has its own benefits and drawbacks. The server communicates with them through the *storage engine API*. This interface hides differences between storage engines and makes them largely transparent at the query layer. The API contains a couple of dozen low-level functions that perform operations such as "begin a transaction" or "fetch the row that has this primary key." The storage engines don't parse SQL or communicate with each other; they simply respond to requests from the server.

2.9.3.2 Top Reasons to Use MySQL

1. Scalability and Flexibility

The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being supported. And, of course, the open source nature of MySQL allows complete customization for those wanting to add unique requirements to the database server.

2. High Performance

A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results. Whether the intended application is a high-speed transactional processing system or a high-volume web site that services a billion queries a day, MySQL can meet the most demanding performance expectations of any system. With high-speed load utilities, distinctive memory caches, full text indexes, and other performance-enhancing mechanisms, MySQL offers all the right ammunition for today's critical business systems.

3. High Availability

Rock-solid reliability and constant availability are hallmarks of MySQL, with customers relying on MySQL to guarantee around-the-clock uptime. MySQL offers a variety of high-availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.

4. Robust Transactional Support

MySQL offers one of the most powerful transactional database engines on the market. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa. Full data integrity is also assured through server-enforced referential integrity, specialized transaction isolation levels, and instant deadlock detection.

5. Web and Data Warehouse Strengths

MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data insert capability, and strong support for specialized web functions like fast full text searches. These same strengths also apply to data warehousing environments where MySQL scales up into the terabyte range for either single servers or scale-out architectures. Other features like main memory tables, B-tree and hash indexes, and compressed archive tables that reduce storage requirements by up to eighty-percent make MySQL a strong standout for both web and business intelligence applications.

6. Strong Data Protection

MySQL offers exceptional security features that ensure absolute data protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server, with the ability to block users down to the client machine level being possible. SSH and SSL support are also provided to ensure safe and secure connections. A granular object privilege framework is present so that users only see the data they should, and powerful data encryption and decryption functions ensure that sensitive data is protected from unauthorized viewing. Finally, backup and recovery utilities provided through MySQL and third party software vendors allow for complete logical and physical backup as well as full and point-in-time recovery.

7. Comprehensive Application Development

One of the reasons MySQL is the world's most popular open source database is that it provides comprehensive support for every application development need. Within the database, support can be found for stored procedures, triggers, functions, views, cursors, ANSI-standard SQL, and more. For embedded applications, plug-in libraries are available to embed MySQL database support into nearly any application. MySQL also provides connectors and drivers (ODBC, JDBC, etc.) that allow all forms of applications to make use of MySQL as a preferred data management server. It doesn't matter if it's PHP, Perl, Java, Visual Basic, or .NET, MySQL offers application developers everything they need to be successful in building database-driven information systems.

8. Management Ease

MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes. This rule holds true whether the platform is Microsoft Windows, Linux, Macintosh, or UNIX. Once installed, self-management features like automatic space expansion, auto-restart, and dynamic configuration changes take much of the burden off already overworked database administrators. MySQL also provides a complete suite of graphical management and migration tools that allow a DBA to manage, troubleshoot, and control the operation of many MySQL servers from a single workstation. Many third party software vendor tools are also available for MySQL that handle tasks ranging from data design and ETL, to complete database administration, job management, and performance monitoring.

2.9.4 Apache HTTP Server

The **Apache HTTP Server**, colloquially called **Apache**, is the world's most used web server software. Originally based on the NCSA (The **National Center for Supercomputing Applications**) HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP server, and has remained most popular since April 1996. In 2009, it became the first web server software to serve more than 100 million websites.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Most commonly used on a UNIX-like system (usually Linux), the software is available for a variety of operating systems besides UNIX, including Microsoft Windows.

2.9.4.1 HTTP server and proxy features

- Loadable Dynamic Modules.
- Multiple Request Processing modes (MPMs) including Event-based/Async, Threaded and Prefork.
- Highly scalable (easily handles more than 10,000 simultaneous connections).
- Handling of static files, index files, auto-indexing and content negotiation.
- Reverse proxy with caching & Load balancing with in-band health checks WebSocket, FastCGI, SCGI, AJP and uWSGI support with caching with Dynamic configuration.
- TLS/SSL with SNI and OCSP stapling support, via OpenSSL.
- Name- and IP address-based virtual servers & IPv6-compatible.
- HTTP/2 protocol support & Fine-grained authentication and authorization access control.
- gzip compression and decompression; .htaccess support; URL rewriting; Headers and content rewriting; Custom logging with rotation; Concurrent connection limiting; Request processing rate limiting; Bandwidth throttling; Server Side Includes.
- IP address-based geo-location; User and Session tracking; Embedded Perl, PHP and Lua scripting; CGI support; public_html per-user web-pages.
- Generic expression parser; Real-time status views; XML support.

2.9.5 AJAX

AJAX is an acronym for Asynchronous JavaScript and XML. It is a group of inter-related technologies like JavaScript, DOM, XML, HTML, CSS etc. AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

2.9.5.1 Understanding Synchronous vs Asynchronous

Before understanding AJAX, let's understand classic web application model and Ajax web application model first.

- **Synchronous (Classic Web-Application Model):** A synchronous request blocks the client until operation completes i.e. browser is not unresponsive. In such case, JavaScript engine of the browser is blocked.

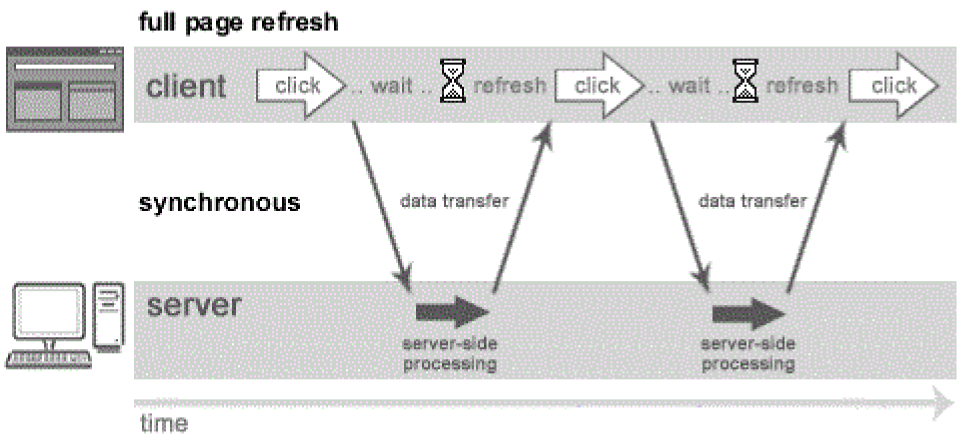


Figure 2.4 (a): Synchronous Web Application Model

As you can see in the above image, full page is refreshed at request time and user is blocked until request completes.

Let's understand it another way.

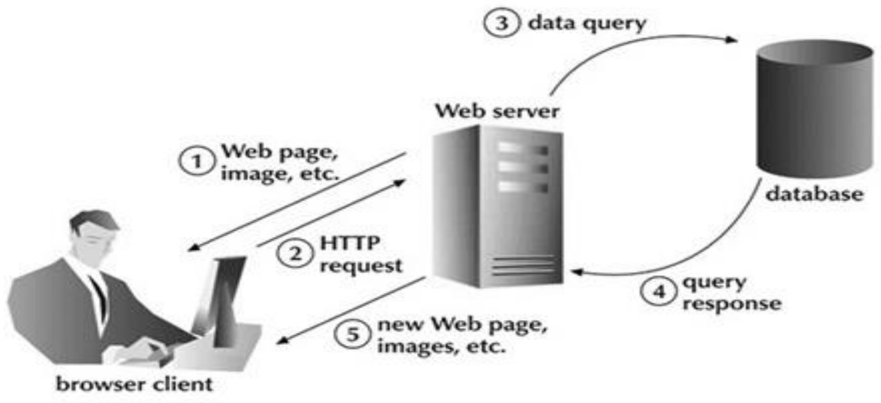


Figure 2.4 (b): Synchronous Web Application Model

- **Asynchronous (AJAX Web-Application Model):** An asynchronous request doesn’t block the client i.e. browser is responsive. At that time, user can perform another operations also. In such case, JavaScript engine of the browser is not blocked.

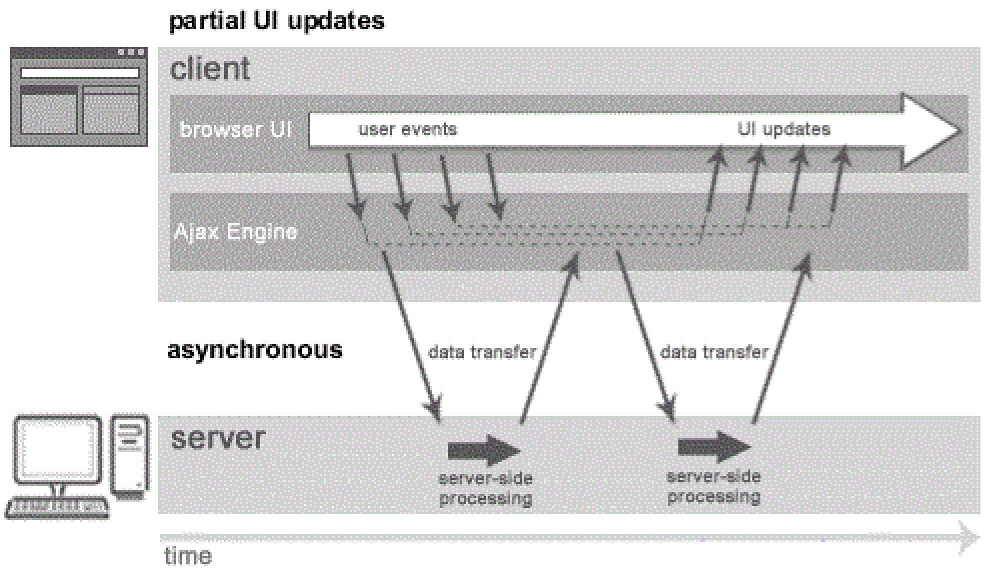


Figure 2.5 (a): AJAX Web-Application Model

As you can see in the above image, full page is not refreshed at request time and user gets response from the Ajax engine.

Let's try to understand asynchronous communication by the image given below.

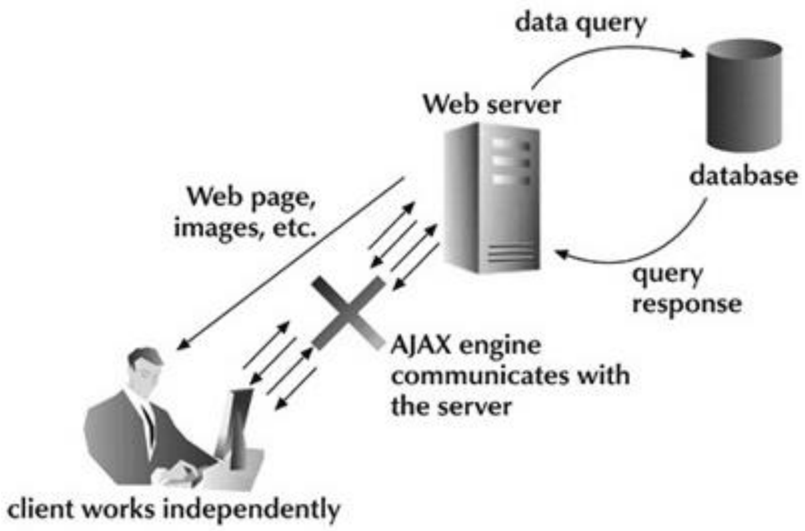

data query
Web server
database
Web page, images, etc.
query response
AJAX engine communicates with the server
client works independently

Figure 2.5 (b): AJAX Web-Application Model

2.9.5.2 How AJAX works?

AJAX communicates with the server using XMLHttpRequest object. Let's try to understand the flow of Ajax or how Ajax works by the image displayed below.

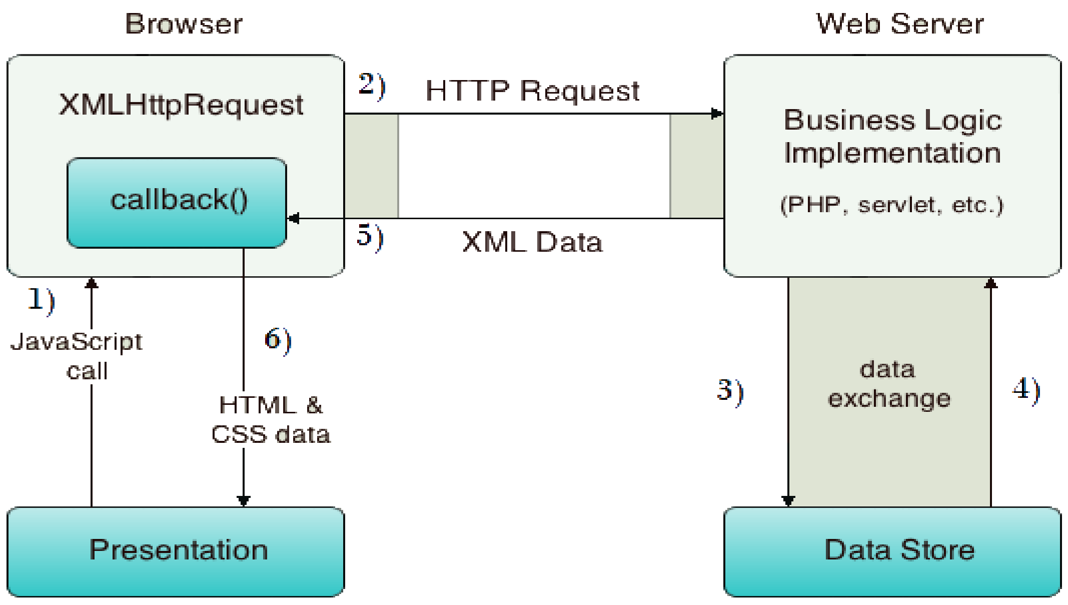


Figure 2.6: How Ajax Works?

As you can see in the above example, XMLHttpRequest object plays an important role:

- User sends a request from the UI and a JavaScript call goes to XMLHttpRequest object.
- HTTP Request is sent to the server by XMLHttpRequest object.
- Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
- Data is retrieved.
- Server sends XML data or JSON data to the XMLHttpRequest callback function.
- HTML and CSS data is displayed on the browser.

3. IMPLEMENTATION

This chapter includes the detailed design used to build the Community Driven System. The system's screen shots are also shown.

3.1. Detailed Scope

This project is supposed to be delivered in two main phases, with each phase being an add-on to the project that makes it more usable and more acceptable.

- In the first delivery, the application must be able to add a user profile and case.
    - User registration.
    - Update user profile.
    - Records Alumni.
- In the second delivery, the application must be able to create groups, enable group administrators to add new admins, manage group join requests.
- After the first two phases of the project have been completed and thoroughly tested, the system would be enhanced into a joint Community Driven and User Training System.

3.2 Static Decomposition and Dependency Description

This section contains the S-Net's design diagrams, which are described to their full extent, so that it helps the coders, end users and other stakeholders to understand the solution proposed.

3.2.1 Use Case Diagram

The System Use Case shows the user a detailed view of the system and how the actors would interact with each other and with the system. The explanation for each use case is then provided below the System Use Case (Figure 3.1) and helps the user understand who the actors areas well as giving the description of each use case along with its pre and post conditions that should be satisfied once the use case is implemented in the software.

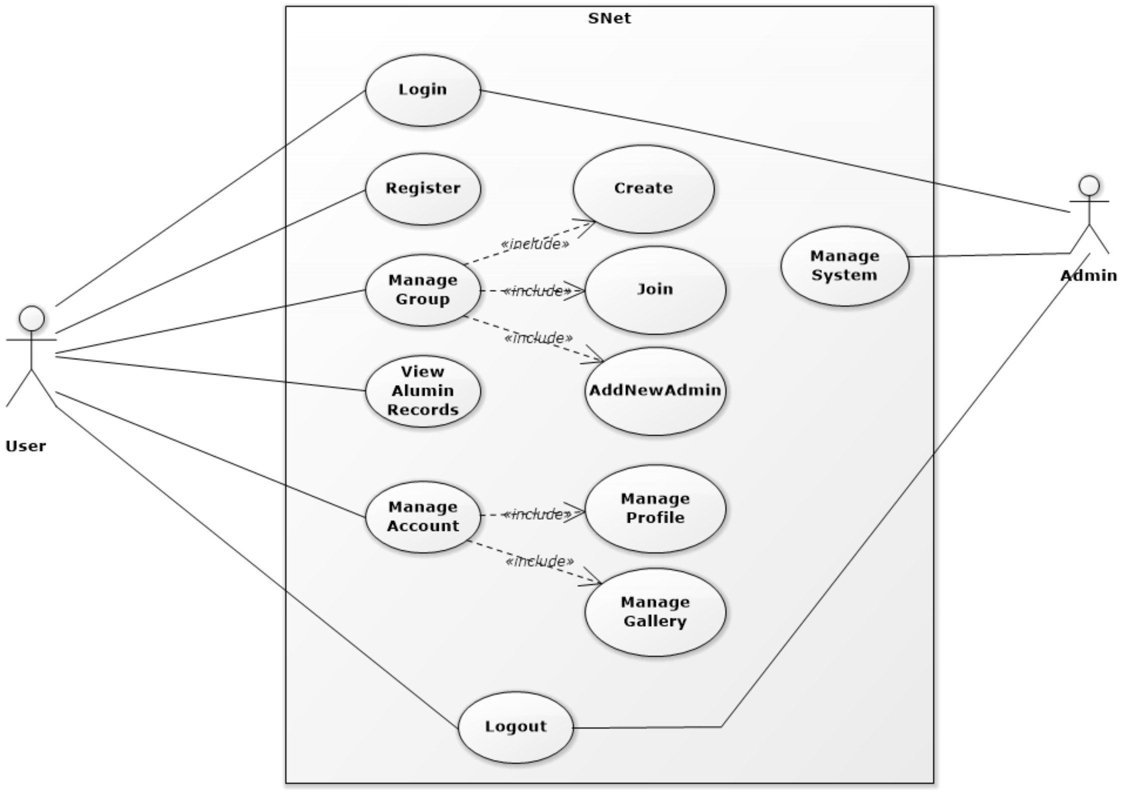


Figure 3.1: S-Net Use Case Diagram

The following is the explanation of the different Use Cases in the System Use case and the Actors associated with each Use Case. The Description is used for a novice user to better understand the working of the System and the pre-conditions that should be satisfied before invoking each use case.

- **Use Case Number:** US-001
    1. **Application:** S-Net.
    2. **Use Case Name:** Login.
    3. **Use Case Description:** This details the login process of the system to access it. The user should login to the system using a username and password provided to him/her.
    4. **Primary Actor:** Admin/User.
    5. **Precondition:** The user is already assigned a username and a password to access the system.
    6. **Post condition:** The user logs into access the system.

- **Use Case Number:** US-002.
    1. **Application:** S-Net.
    2. **Use Case Name:** Register.
    3. **Use Case Description:** This details the user registration process of the system.
    4. **Primary Actor:** Users.
    5. **Precondition:** The user about to register should not have an existing account.
    6. **Post Condition:** The system should maintain its idle state by either reflecting a successful operation, or by a failed operation.
- **Use Case Number:** US-003
    1. **Application:** S-Net.
    2. **Use Case Name:** Manage Group.
    3. **Use Case Description:** This details the group management process of the system.
    4. **Primary Actor:** User/Group Admin.
    5. **Precondition:** The system should maintain its idle state before any operation is performed.
    6. **Post Condition:** The system should return to its idle state after any operation done on the system, irrespective of the result.
- **Use Case Number:** US-004
    1. **Application:** S-Net.
    2. **Use Case Name:** View Alumni Records.
    3. **Use Case Description:** This details the user aspect of viewing alumni records.
    4. **Primary Actor:** User.
    5. **Precondition:** only authorized users, who are required to logon to the system; are allowed to view the records.
    6. **Post Condition:** The system displays the searched record and system should return to its idle state after operation done on the system, irrespective of the result.

- **Use Case Number:** US-005.
    1. **Application:** S-Net.
    2. **Use Case Name:** Manage Account.
    3. **Use Case Description:** This details the options available to a user, such as manage profile and manage personal gallery.
    4. **Primary Actor:** Users.
    5. **Precondition:** The system should store each request that has a sound reason.
    6. **Post Condition:** The request is serviced as submitted by a user.
- **Use Case Number:** US-006
    1. **Application:** S-Net.
    2. **Use Case Name:** Manage System.
    3. **Use Case Description:** This details about the system maintenance functions.
    4. **Primary Actor:** System Administrator.
    5. **Precondition:** The administrator is required to be logged in before performing any action.
    6. **Post Condition:** The system should return to its idle state after any operation done on the system, irrespective of the result
- **Use Case Number:** US-007.
    1. **Application:** S-Net.
    2. **Use Case Name:** Logout.
    3. **Use Case Description:** This details about the logout process of the system.
    4. **Primary Actor:** Admin/User.
    5. **Precondition:** The users are required to be logged in and are accessing the system.
    6. **Post Condition:** The users are redirected to the initial or the homepage of the application.

3.2.2 Class Diagram

A class diagram shows the static structure of classes in the system. The classes represent the "things" that are handled in the system. Classes can be related to each other in a number of ways: They can be associated (connected to each other), dependent (one class depends on or uses another class), specialized (one class is a specialization of another class), or packaged (grouped together as a unit).

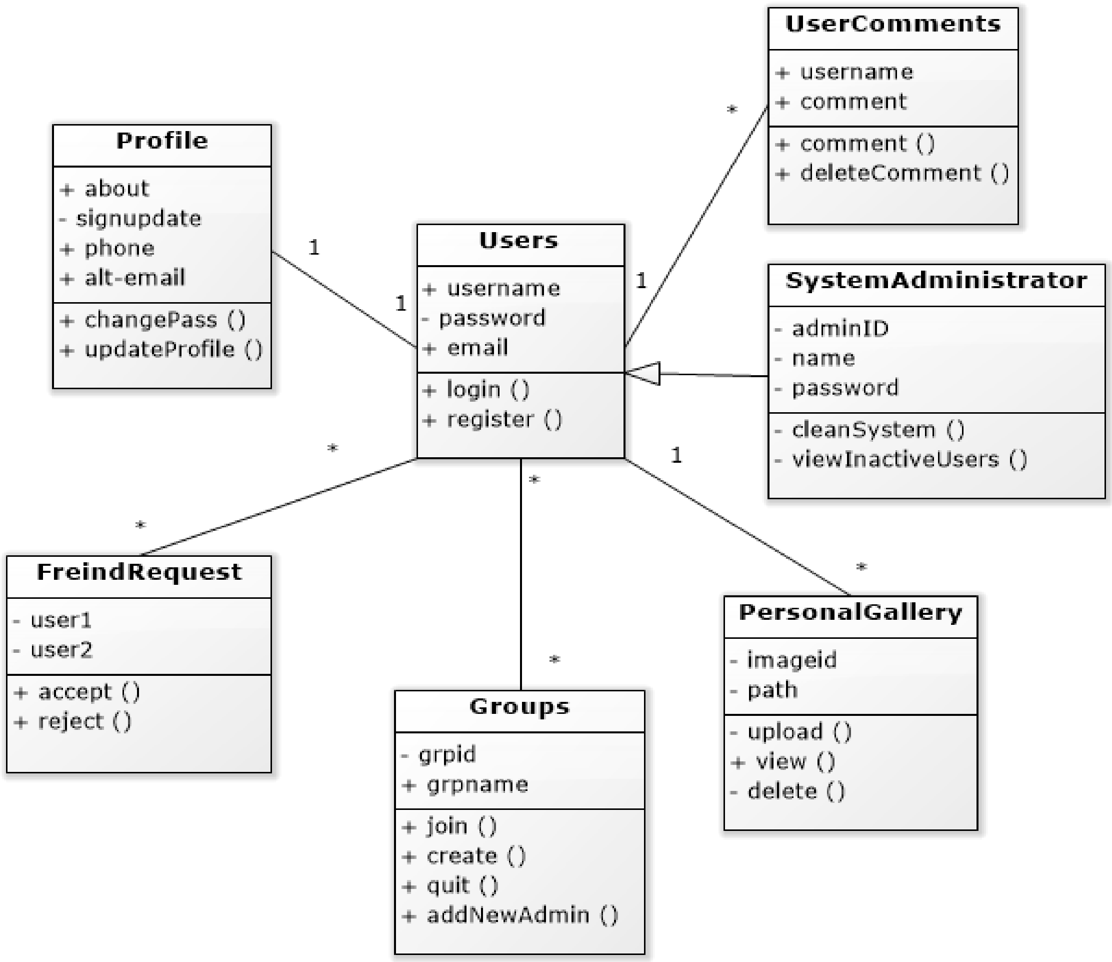


Figure 3.2: S-Net Class Diagram

The above illustration (Figure 3.2) represents the overall structure of the system; that is, it describes the system using the static aspect of the design proposed. The following is the explanation of various classes; that describes which classes are dependent or associated or generalized, and what each class can do for us along with the attributes that are required.

- **Users:** This class is used by the users of the system, and it's an important class, that is responsible for the proper functioning of the system.
    - **Attribute(s):** a) **Username**. b) **Password**. c) **E-mail**.
    - **Method(s):**
        1. **Login ():** This method allows a user to login and gain access to the system. Based on the role, the respective screen is displayed to the user**.**
        2. **Register ():** This method allows a new comer to register himself/herself, such that they can start using the system.
        3. **Logout ():** This method allows a user to log out of the system.
    - A subclass named **System Administrator** is required so as to maintain and manage the system.
- **System Administrator:** This class is responsible for maintaining and managing the system.
    - **Attribute(s):** a) **adminID.** b) **name.** c) **password.**
    - **Method(s):**
        1. **cleanSystem ():** This method allows the system administrator to monitor the system usage and take precautionary measures and also perform cleanup of inactive users.
        2. **ViewInactiveUsers ():** This method lists all the inactive users, so that the cleanSystem() can be used to maintain and manage the system.
- **Groups:** This class is core of the system as it describes the whole system; that is, "Community Driven", where users can come along with common ideas and share them.
    - **Attribute(s):** a) **grpid**. b) **grpname.**
    - **Method(s):**
        1. **Join ():** This method enables users to join groups.
        2. **Create ():** This method enables users to create new groups.
        3. **Quit ():** This method enables users to quit groups.
        4. **addNewAdmin ():** This method enables a group admin to add a new admin to the group.

- **Profile:** This class enables users of the system to change as well as update their details, such that only current professional details get reflected.
  - **Attribute(s):** a) **about**. b) **signupdate**. c) **phone**. d) **alt-e-mail.**
  - **Method(s):**
    1. **changePass ():** This method enables users to change their password.
    2. **updateProfile ():** This method enables users to update their profiles.
- **Friend Requests:** This class is responsible for managing all the user's friends list.
  - **Attribute(s):** a) **user1.** b) **user2.**
  - **Method(s):**
    1. **Accept ():** This method allows a requested user to accept the friendship from the initiator of the request.
    2. **Reject ():** This method allows a requested user to ignore the friendship from the initiator of the request.
- **User Comments:** This class is responsible for storing comments.
  - **Attribute(s):** a) **username.** b) **comment.**
  - **Method(s):**
    1. **Comment ():** This method enables users to comment on other user's posts.
    2. **deleteComment ():** This method allows users to remove comments.
- **Personal Gallery:** This class enables users to upload photos/images to the system.
  - **Attribute(s):** a) **imgid.** b) **path.**
  - **Method(s):**
    1. **Upload ():** This method allows owner of the account upload photos to the system.
    2. **View ():** This method allows users to view other user's gallery.
    3. **Delete ():** This method allows owner of the account to delete photos from the system.

3.2.3 Sequence Diagram

A sequence diagram shows a dynamic collaboration between a numbers of objects. The important aspect of this diagram is that it shows a sequence of messages sent between the objects. Time passes downward in the diagram, and the diagram shows the exchange of messages between the objects as time passes in the sequence or function.

3.2.3.1 Login Sequence

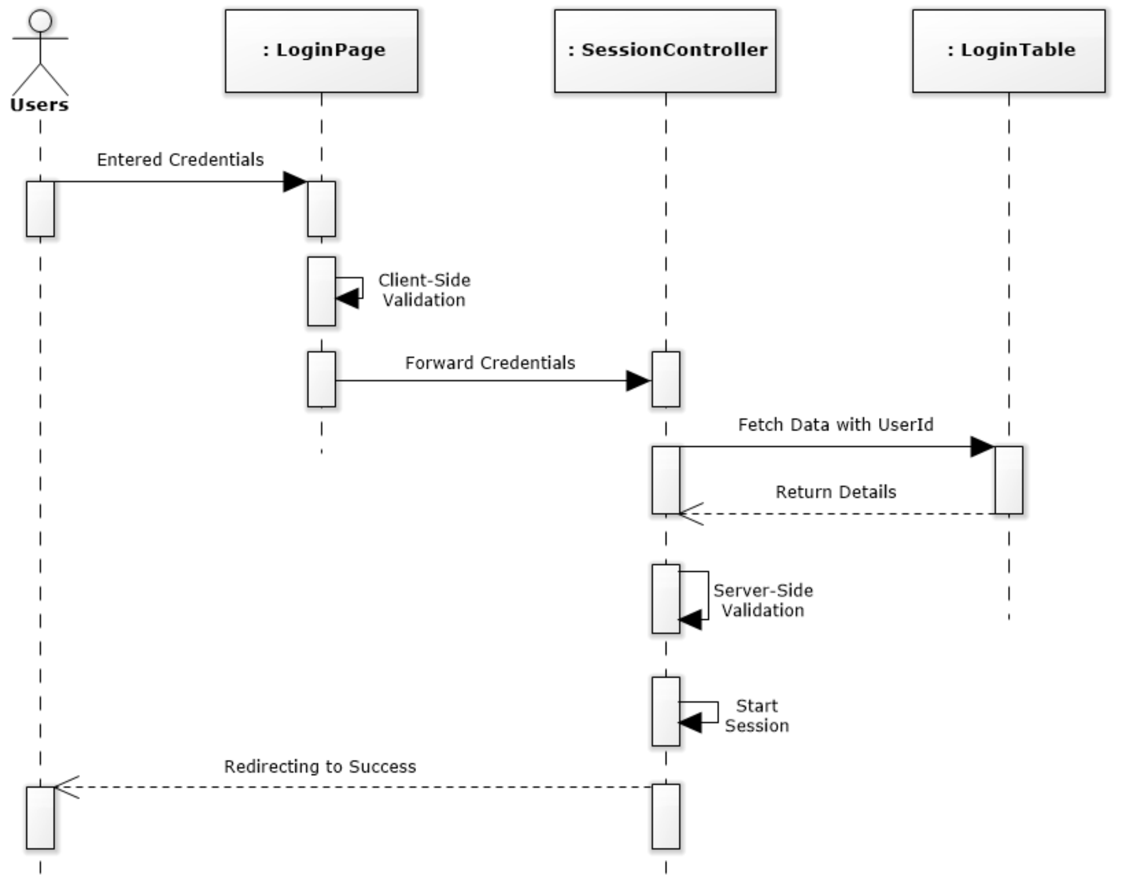


Figure 3.3: Sequence Diagram for Login

The following is the explanation of the Figure 3.3.

- **Primary Actor:** User (Admin/Normal User).
- **Basic Flow:**
  - I. The user enters the login credentials.
  - II. A basic client side validation is done.

III. Credentials are then validated with data from the database, thus performing server side validation.

IV. After are validation the user is redirected to the user homepage.

- **Exceptional Flow:**

I. User (Admin/Normal User) enters the login credentials.

II. A basic client side validation is done, entry fields that are missing are alerted

III. Credentials are then checked with data in the database, on failure, reports an error which is alerted.

IV. Error and the user are redirected to the login page.

3.2.3.2 Register Sequence

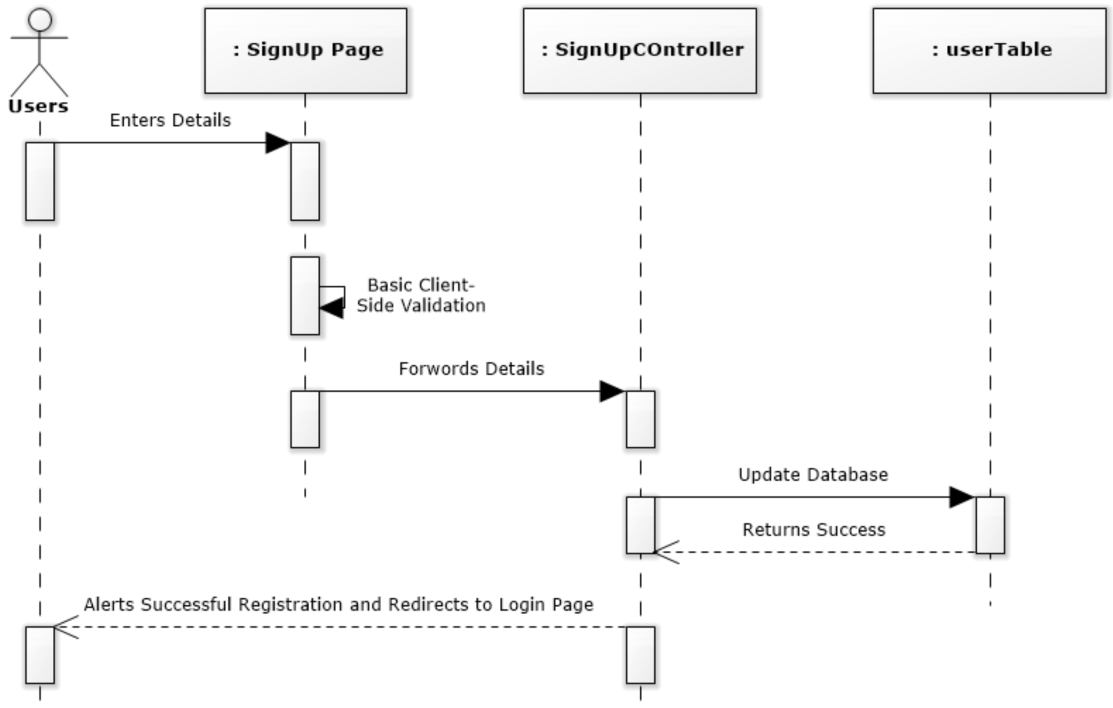


Figure 3.4: Sequence Diagram for Signup

The following is the explanation of the Figure 3.4.

- **Primary Actor:** User.
- **Basic Flow:**

I. The user enters the signup details.

II. A basic client side validation is done.

III. Details are then sent to the server and the details are stored.

IV. After are successful operation the user is redirected to the user login page.

- **Exceptional Flow:**

I. The user enters the signup credentials.

II. A basic client side validation is done, entry fields that are missing are alerted.

III. Credentials are then updated with data into the database, on failure, reports an error which is alerted.

IV. Error and the user are redirected to the signup page.

3.2.3.3 Create Group Sequence

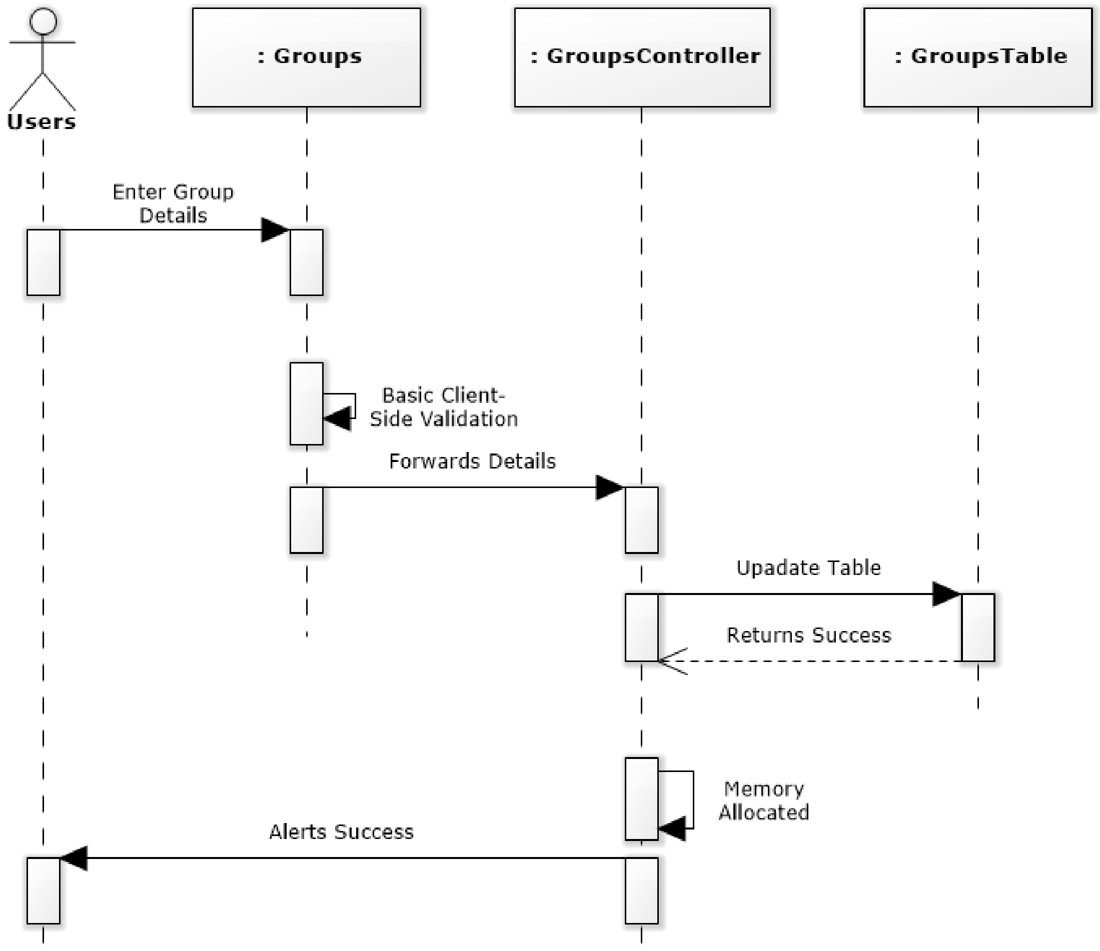


Figure 3.5: Sequence Diagram for Creating a Group

The following is the explanation of the Figure 3.5.

- **Primary Actor:** User.
- **Basic Flow:**
  I. The user login to the system, and select "groups" option.
  II. User enter the group name and type request of joining the group.
  III. A basic client-side validation is done
  IV. After are successful operation the user is redirected to the user group's home page.
- **Exceptional Flow:**
  I. The user enters the login credentials.
  II. A basic client side validation is done, entry fields that are missing are alerted.
  III. Group details are then updated with data into the database, on failure, reports an error which is alerted.
  IV. Error and the user are redirected to the group's main page.

3.2.3.4 View Alumni Records Sequence

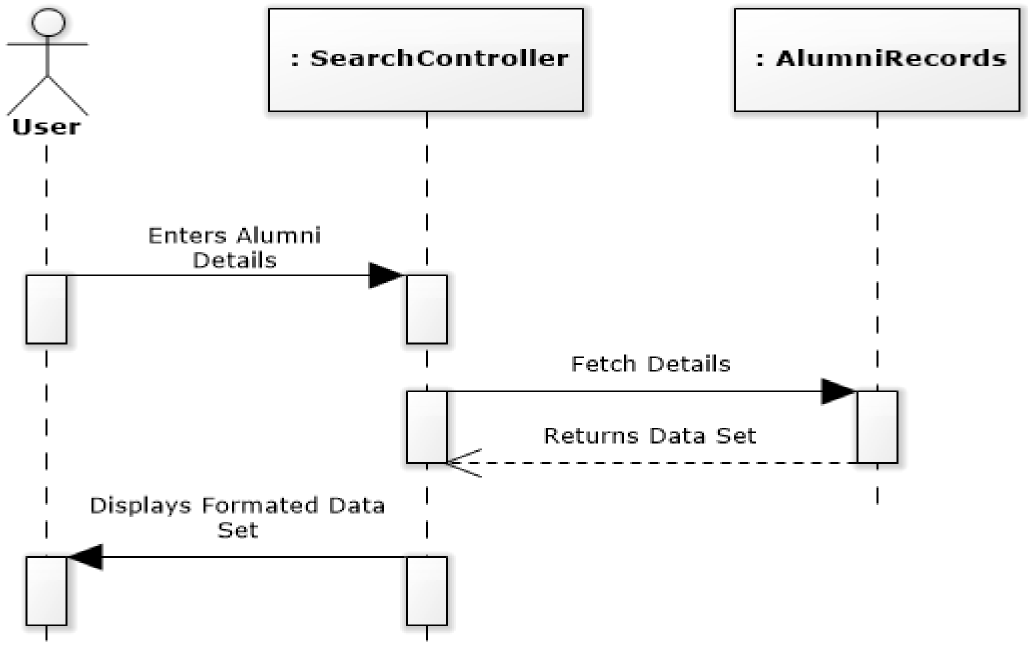


Figure 3.6: Sequence Diagram for Viewing Alumni Records

The following is the explanation of the Figure 3.6.

- **Primary Actor:** User.
- **Basic Flow:**
    I. The user login to the system, and select “Alumni” option.
    II. User enter the alumni name and hits on search.
    III. After are successful operation the user is shown with a list of alumni records.
- **Exceptional Flow:**
    I. The user enters the login credentials.
    II. A basic client side validation is done, entry fields that are missing are alerted.
    III. When searched, on failure, reports an error which is alerted.
    IV. Error and the user are redirected to the alumni’s main page.

3.2.3.5 Logout Sequence

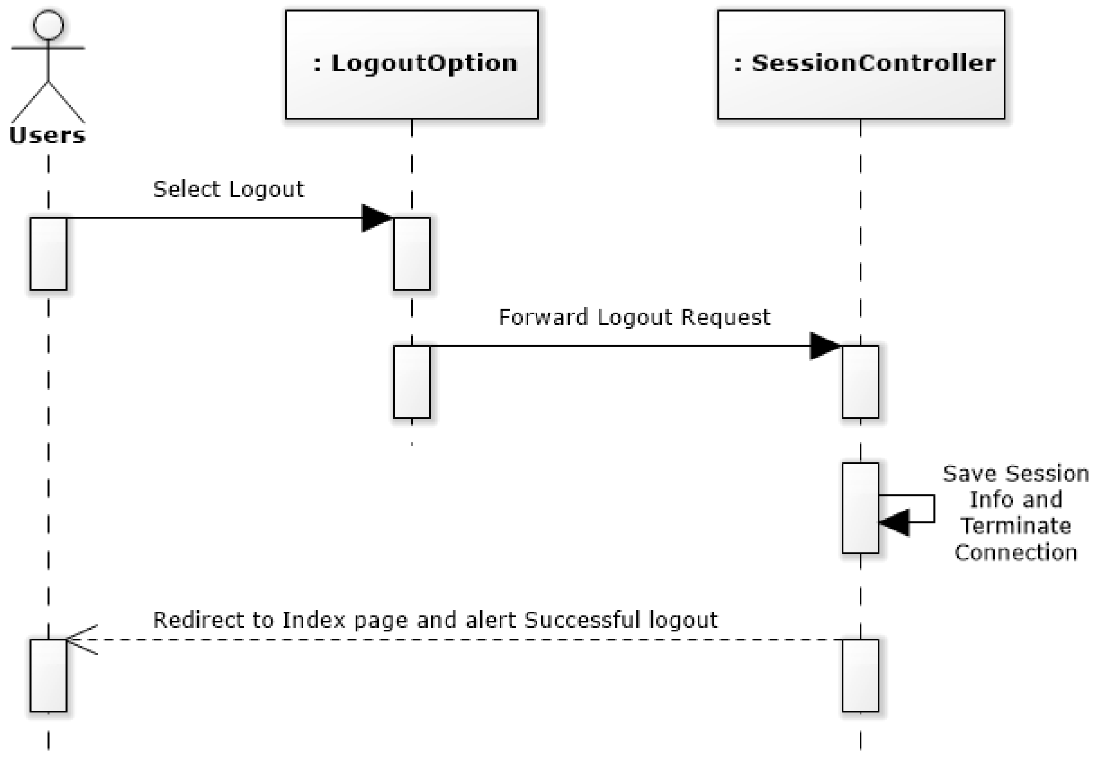

: LogoutOption
: SessionController
Users
Select Logout
Forward Logout Request
Save Session Info and Terminate Connection
Redirect to Index page and alert Successful logout

Figure 3.7: Sequence Diagram for Logout

The following is the explanation of Figure 3.7.

- **Primary Actor:** Users (Admins/Normal Users).
- **Basic Flow:**
  - I. The user select logout option.
  - II. This request is answered by the server and the session is stored.
  - III. The user is then redirected to the index page.
- **Exceptional Flow:**
  - I. The user select logout option.
  - II. This request is answered by the server and on internal errors, a failed error is reported.
  - III. After the error is logged and reported, the user is then redirected to the index page.

3.2.4 Data Flow Diagrams

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel unlike a flowchart which also shows this information.

3.2.4.1 Context Diagram

The highest level data flow diagram is the context diagram

- The context diagram shows the interactions of the system with its environment in terms of data flow
- The context diagram defines the boundary of the system; that’s, the scope of the system.
- Only data flow diagrams which leave the system and the data flow which come from outside the system are show.

0 level DFD: A 0 level DFD, also called as the fundamental system model or context diagram represents the entire software element as a single bubble with input and output indicated by the incoming and outgoing arrows, respectively.

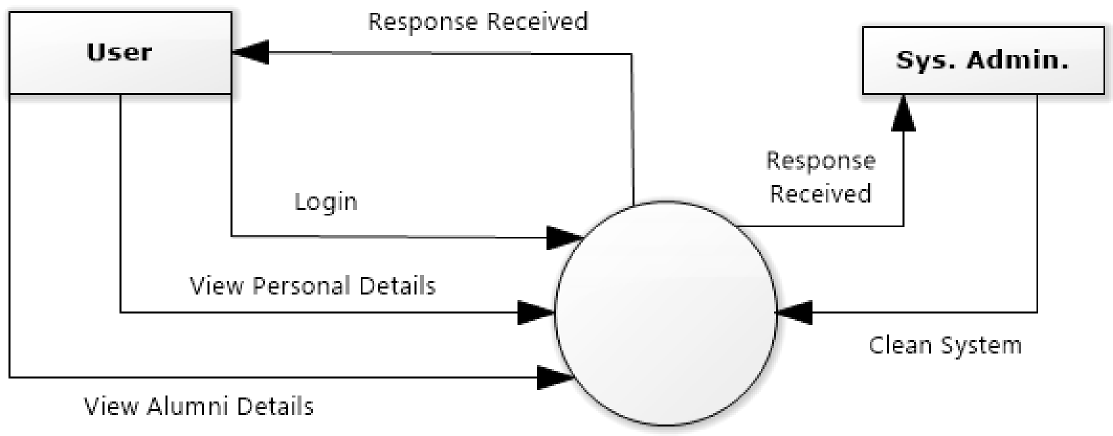


Figure 3.8: 0 Level DFD

1 Level DFD: This level of DFD provides more detailed structure. It provides a detailed view of the requirements and the flow of data from 1 bubble to another

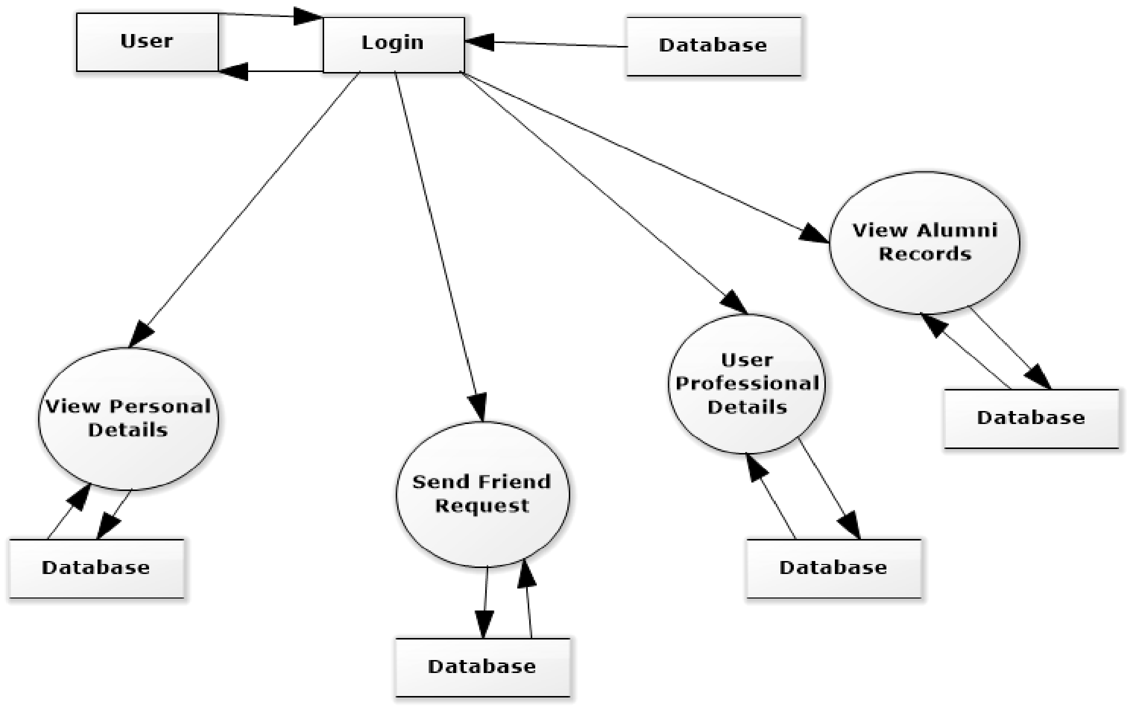


Figure 3.9: 1 Level DFD

3.3 The Database Design

An entity–relationship model (**ER model**) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database.

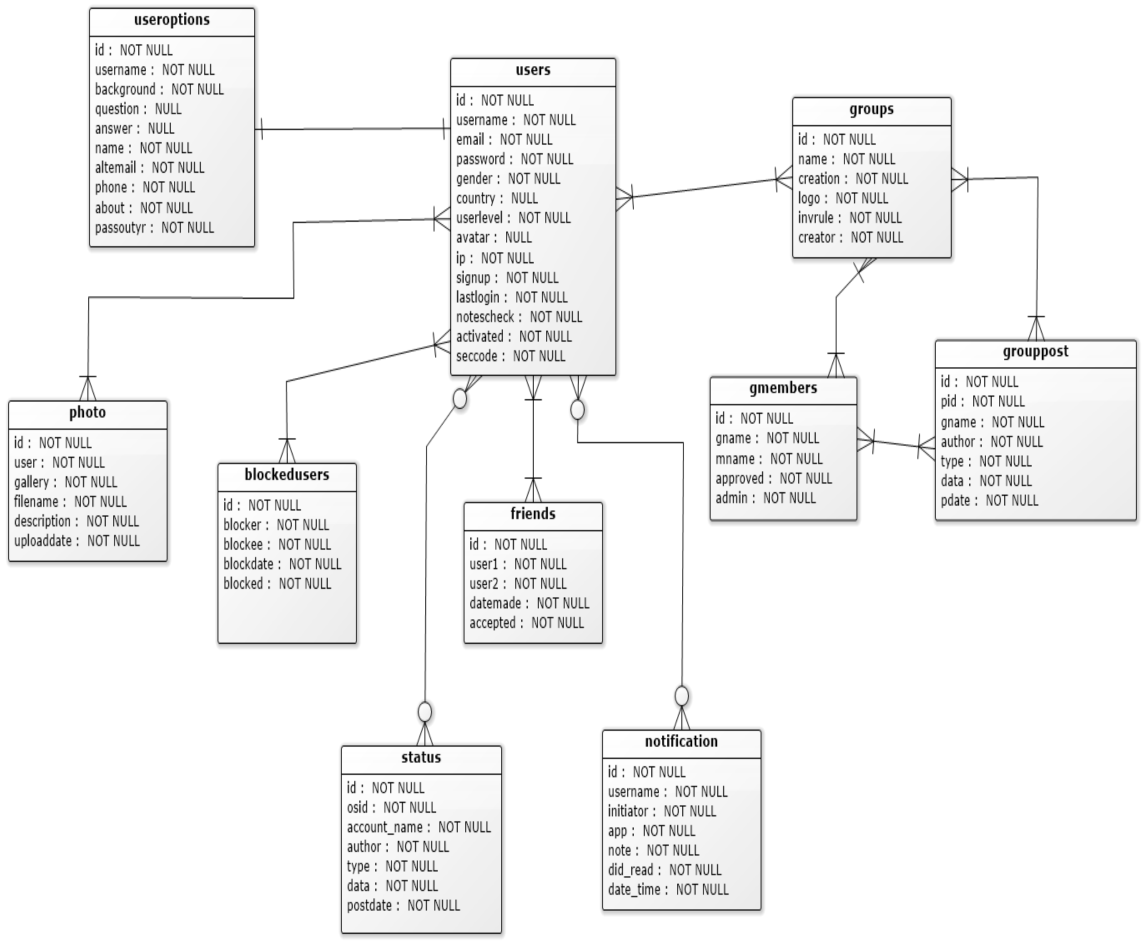


Figure 3.10: Entity Relationship Diagram for S-Net

A database is an organized collection of data. There are many different strategies for organizing data to facilitate easy access and manipulation. A database management system (DBMS) provides mechanisms for storing, organizing, retrieving and modifying data for many users. Database management systems allow for the access and storage of data without concern for the internal representation of data. The following is the explanation of the Figure 3.10.

- **Users Table: user**
    - **Description:** This table is responsible for storing all user related data.

Attribute	Type	Null	Default
id	int(11)	No	
username	varchar(16)	No	
email	varchar(255)	No	
password	varchar(255)	No	
gender	enum('m', 'f')	No	
country	varchar(255)	Yes	NULL
userlevel	enum('A', 'B', 'C', 'D')	No	A
avatar	varchar(255)	Yes	NULL
ip	varchar(255)	No	
signup	datetime	No	
lastlogin	datetime	No	
notescheck	datetime	No	
activated	enum('0', '1')	No	1
seccode	int(5)	No	

Table 3.1: Structure of Users Table

- **User Options Table: useroptions**
    - **Description:** This Table is responsible for storing all user related data for viewing profile

Attribute	**Type**	**Null**	**Default**
id	int(11)	No	
username	varchar(16)	No	
background	varchar(255)	No	
question	varchar(255)	Yes	NULL
answer	varchar(255)	Yes	NULL
name	varchar(255)	No	
altemail	varchar(255)	No	
phone	int(10)	No	
about	text	No	
passoutyr	date	No	

Table 3.2: Structure of User Options Table

- **Status Table: status**
    - **Description:** This table is responsible for holding group post data

Attribute	Type	Null	Default
id	int(11)	No	
osid	int(11)	No	
account_name	varchar(16)	No	
author	varchar(16)	No	
type	enum('a', 'b', 'c')	No	a
data	text	No	
postdate	datetime	No	

Table 3.3: Structure of Status Table

- **Photos Table: photos**
    - **Description:** This table is responsible for holding all the user uploaded photos; that is, the user's gallery.

Attribute	Type	Null	Default
id	int(11)	No	
user	varchar(16)	No	
gallery	varchar(16)	No	
filename	varchar(255)	No	
description	varchar(255)	Yes	NULL
uploaddate	datetime	No	

Table 3.4: Structure of Photos Table

- **Notifications Table: notifications**
    - **Description:** This table is responsible for holding all the unread threads, posts by users.

Attribute	Type	Null	Default
id	int(11)	No	
username	varchar(16)	No	
initiator	varchar(16)	No	
app	varchar(255)	No	
note	varchar(255)	No	
did_read	enum('0', '1')	No	0
date_time	datetime	No	

Table 3.5: Structure of Notifications Table

- **Groups Table: groups**
    - **Description:** This table is responsible for holding all the groups details.

Attribute	Type	Null	Default
id	int(11)	No	
name	varchar(100)	No	
creation	datetime	No	
logo	varchar(255)	No	
invrule	enum('0', '1')	No	
creator	varchar(16)	No	

Table 3.6: Structure of Groups Table

- **Group Posts Table: grouppost**
    - **Description:** This table is responsible for holding all the posts that a user posts in a group.

Attribute	Type	Null	Default
id	int(11)	No	
pid	varchar(16)	No	
gname	varchar(100)	No	
author	varchar(16)	No	
type	enum('0', '1')	No	
data	text	No	
pdate	datetime	No	

Table 3.7: Structure of Group Post Table

- **Group Members Table: gmembers**
    - **Description:** This table is responsible for holding all the member details of a group.

Attribute	Type	Null	Default
id	int(11)	No	
gname	varchar(100)	No	
mname	varchar(16)	No	
approved	enum('0', '1')	No	
admin	enum('0', '1')	No	

Table 3.8: Structure of Group Members Table

- **Friends Table: friends**
  - **Description:** This table is responsible for holding all the friend requested made by the users of the system.

Attribute	Type	Null	Default
id	int(11)	No	
user1	varchar(16)	No	
user2	varchar(16)	No	
datemade	datetime	No	
accepted	enum('0', '1')	No	0

Table 3.9: Structure of Friends Table

- **Blocked Users Table: blockedusers**
  - **Description:** This table is responsible for holding all the blocked users details; that's between blocker and blockee.

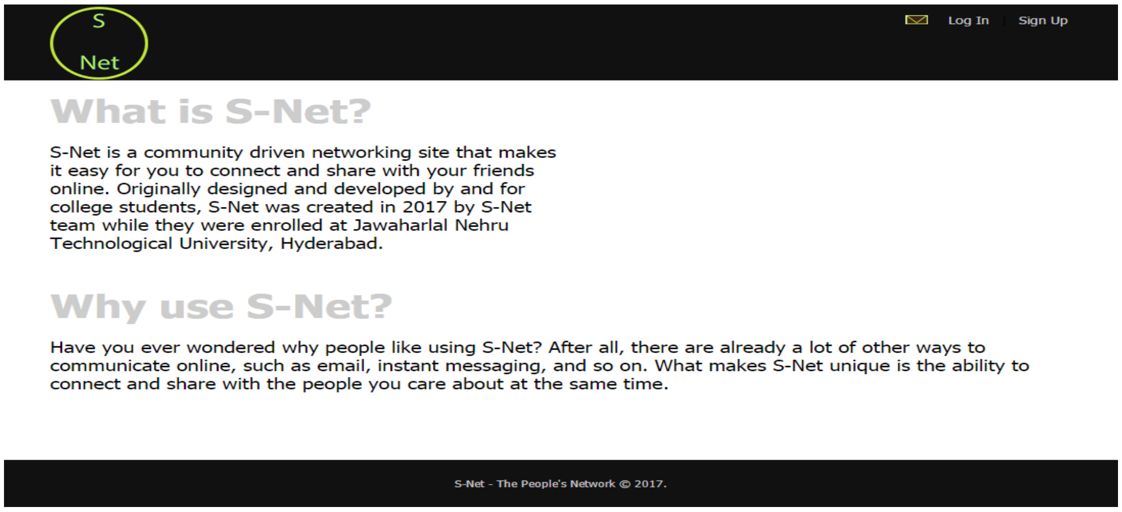
Attribute	Type	Null	Default
id	int(11)	No	
blocker	varchar(16)	No	
blockee	varchar(16)	No	
blockdate	datetime	No	
blocked	enum('0', '1')	No	1

Table 3.10: Structure of Blocked User Table

3.4 Interface Description

This section details about various interfaces of the system. Where each interface is explained with a screen shot of the Interface.

- **Main Interface:** Figure 3.11 describes about the main interface, which displays the Index Page of S-Net, which is common to all. It's the primary navigation point, where Users (Users/Admins) decide their destination.



What is S-Net?

S-Net is a community driven networking site that makes it easy for you to connect and share with your friends online. Originally designed and developed by and for college students, S-Net was created in 2017 by S-Net team while they were enrolled at Jawaharlal Nehru Technological University, Hyderabad.

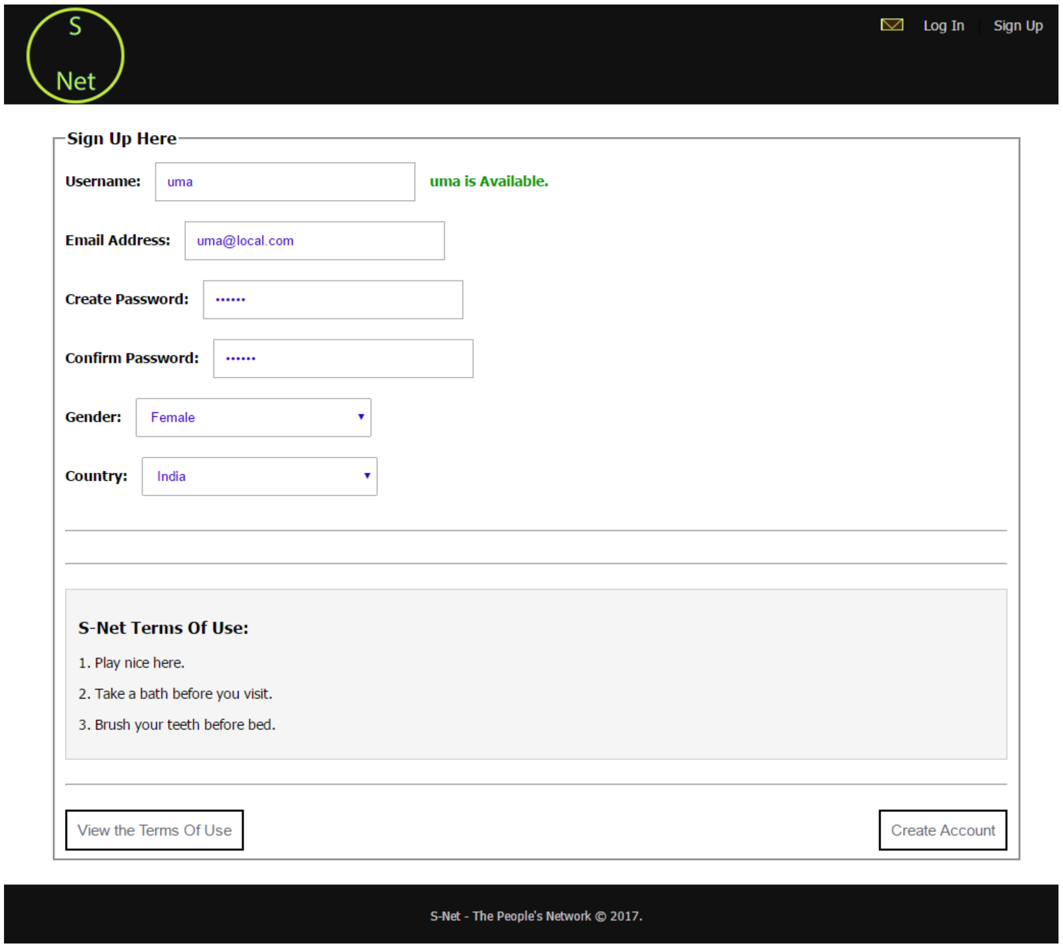
Why use S-Net?

Have you ever wondered why people like using S-Net? After all, there are already a lot of other ways to communicate online, such as email, instant messaging, and so on. What makes S-Net unique is the ability to connect and share with the people you care about at the same time.

S-Net - The People's Network © 2017.

Figure 3.11: Index Page – S-Net

- **Signup Interface:** Figure 3.12, depicts signup interface that enables new users, to register and to start using the system.



Sign Up Here

Username: uma — uma is Available.

Email Address: uma@local.com

Create Password: ••••••

Confirm Password: ••••••

Gender: Female

Country: India

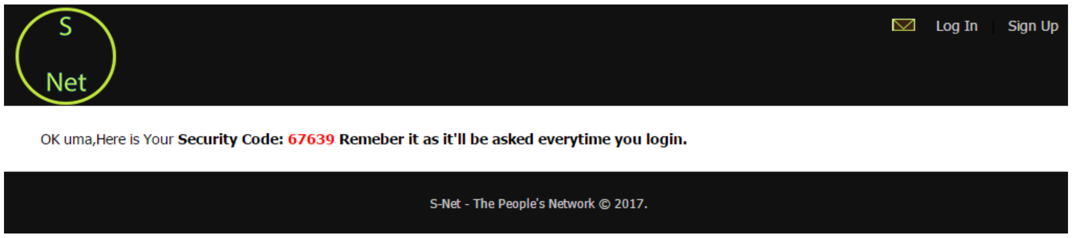
S-Net Terms Of Use:

1. Play nice here.
2. Take a bath before you visit.
3. Brush your teeth before bed.

S-Net - The People's Network © 2017.

Figure 3.12: Signup Page – S-Net

- **Successful Signup Snapshot:** Figure 3.13, depicts a successful signup page. Note that the secret code is confidential and should not be shared with others.

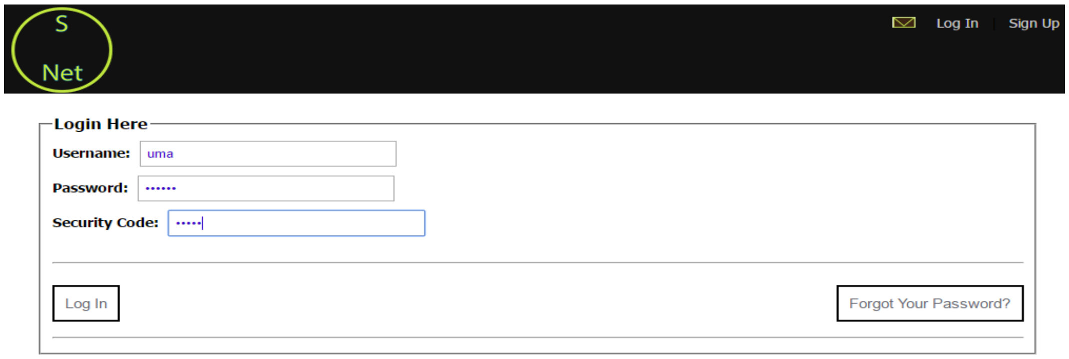


OK uma,Here is Your **Security Code: 67639 Remeber it as it'll be asked everytime you login.**

S-Net - The People's Network © 2017.

Figure 3.13: Successful Signup – S-Net

- **Login Interface:** The Figure 3.14, shows the common login page for users, where the login credentials are entered and then the user is redirected to the profile page.

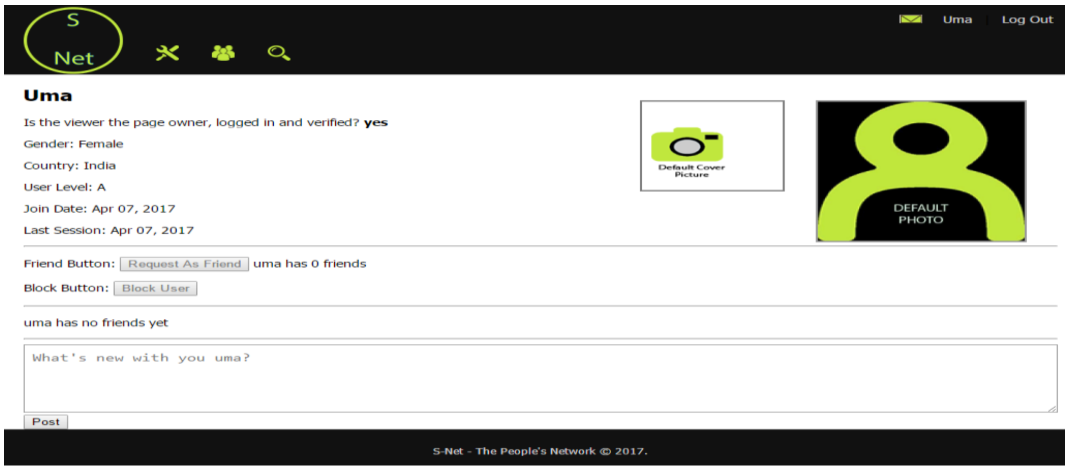


Login Here

Username:	uma
Password:	••••••
Security Code:	•••••

Figure 3.14: Login Page – S-Net

- **Profile Page Interface:** Figure 3.15, depicts the common profile page for every user.



Uma

Is the viewer the page owner, logged in and verified? **yes**
Gender: Female
Country: India
User Level: A
Join Date: Apr 07, 2017
Last Session: Apr 07, 2017

Friend Button: Request As Friend uma has 0 friends
Block Button: Block User

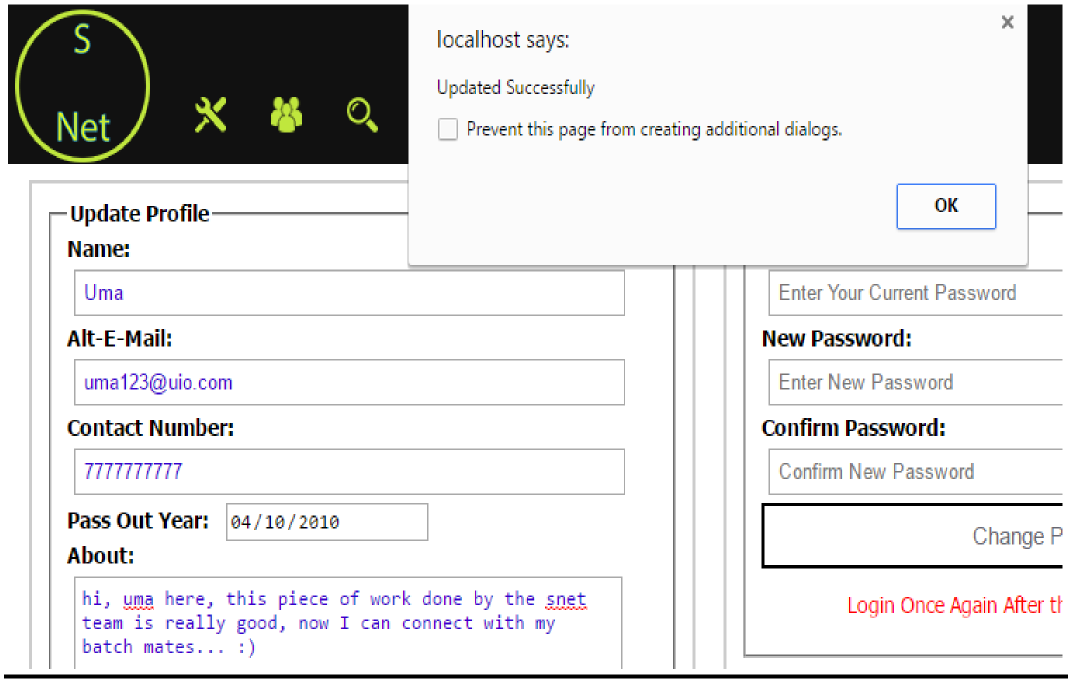
uma has no friends yet

What's new with you uma?

S-Net - The People's Network © 2017.

Figure 3.15: Profile Page – S-Net

- **Settings Interface:** Figure 3.16, shows a basic account settings page, where the user needs to fill out a form, so that their profile is complete.



localhost says:

Updated Successfully

Prevent this page from creating additional dialogs.

Update Profile

Name: Uma

Alt-E-Mail: uma123@uio.com

Contact Number: 7777777777

Pass Out Year: 04/10/2010

About: hi, uma here, this piece of work done by the snet team is really good, now I can connect with my batch mates... :)

Enter Your Current Password

New Password: Enter New Password

Confirm Password: Confirm New Password

Login Once Again After th

Figure 3.16: Settings Page – S-Net

- **Groups Page Interface:** Figure 3.17, shows the option available to users to create a community/group, where they can share common ideas and work on them.

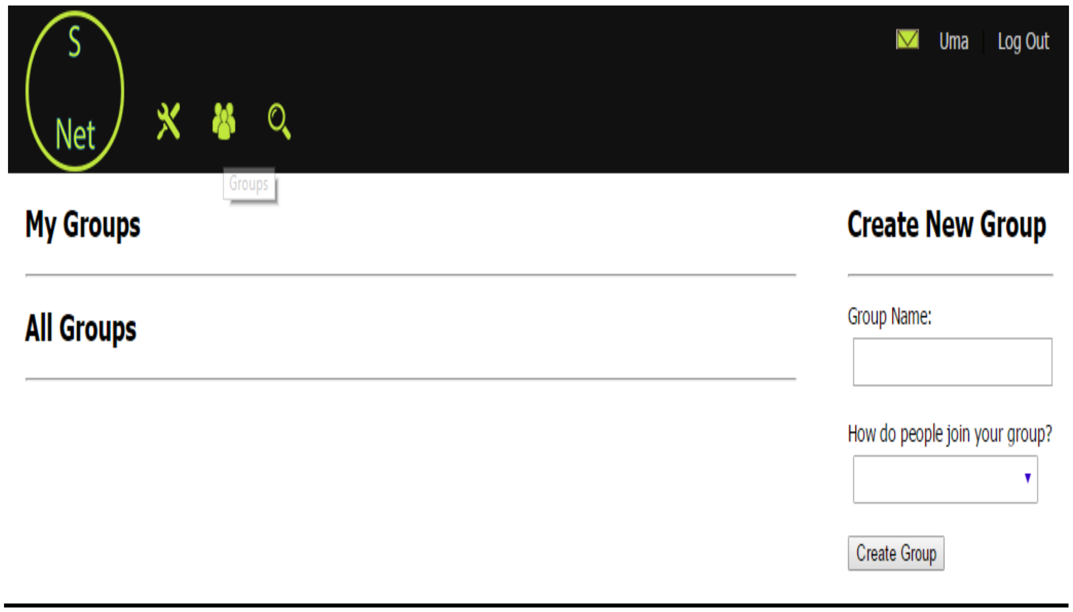
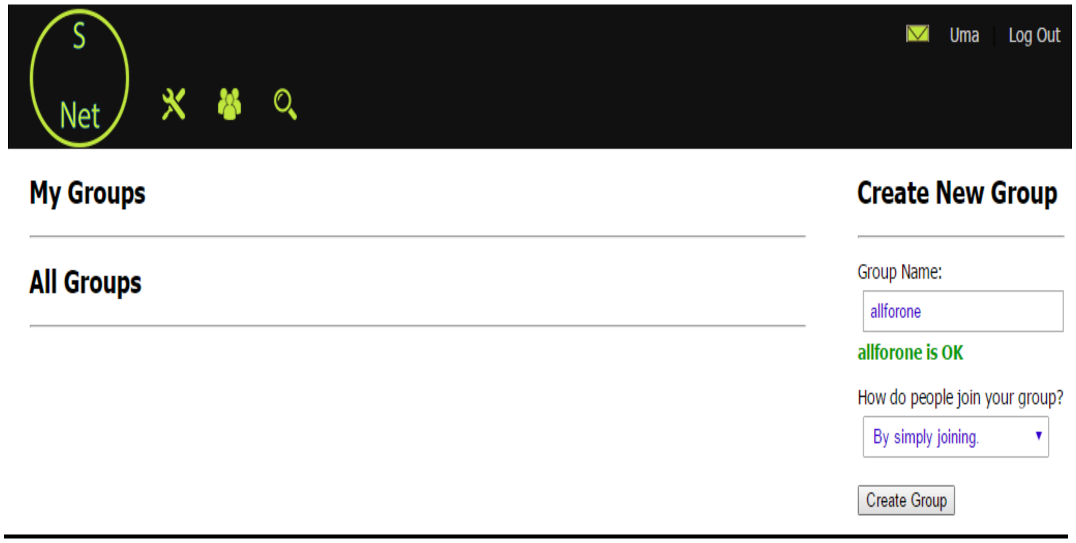


Figure 3.17: Groups Page – S-Net

- Figures 3.18, 3.19, show how a group can be created and Figure 3.20, 3.21 shows a user named **“madmax”** joining the group named **“allforone”**.



My Groups

All Groups

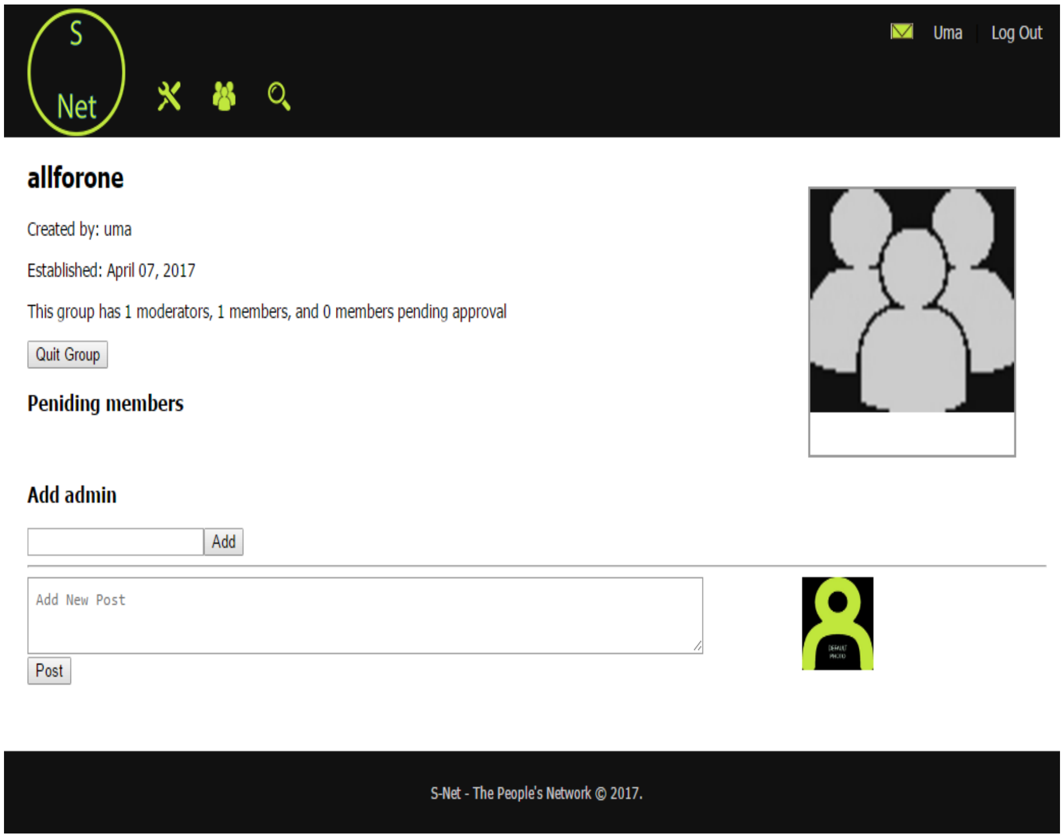
Create New Group

Group Name: allforone

allforone is OK

How do people join your group? By simply joining.

Figure 3.18: Create New Group Page – S-Net



allforone

Created by: uma

Established: April 07, 2017

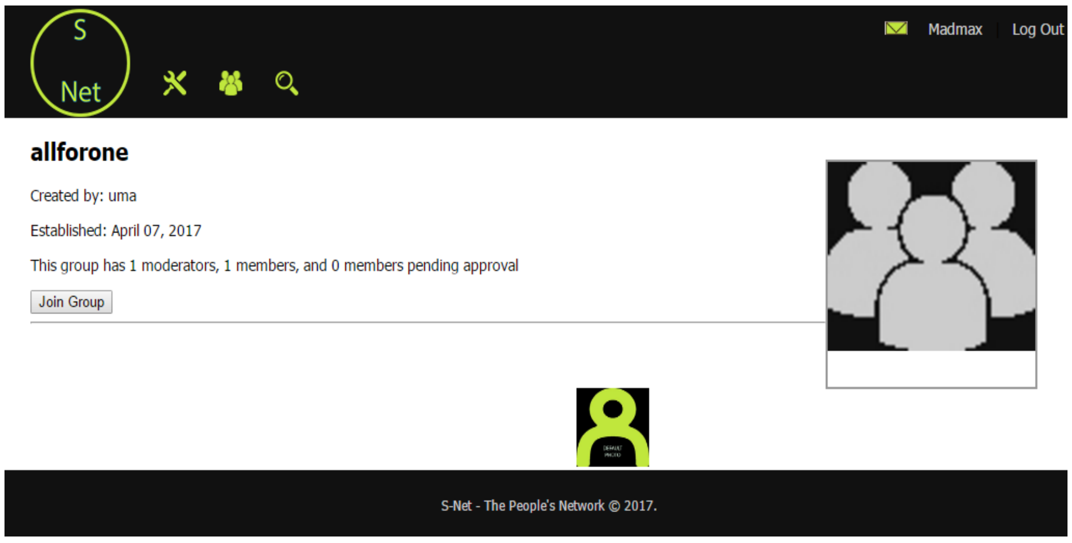
This group has 1 moderators, 1 members, and 0 members pending approval

Peniding members

Add admin

S-Net - The People's Network © 2017.

Figure 3.19: Group Created Page – S-Net



allforone

Created by: uma

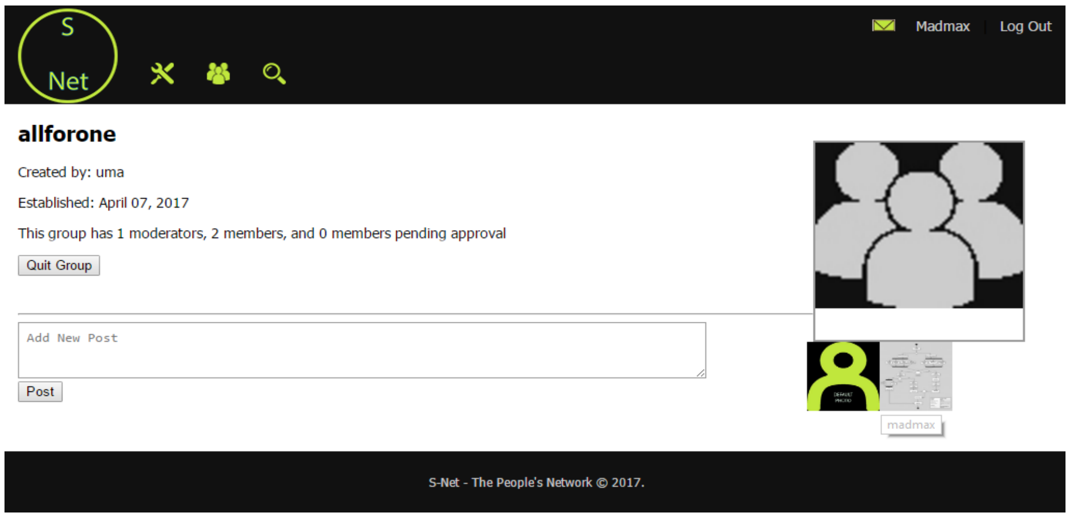
Established: April 07, 2017

This group has 1 moderators, 1 members, and 0 members pending approval

Join Group

S-Net - The People's Network © 2017.

Figure 3.20: Group Join Page – S-Net



allforone

Created by: uma

Established: April 07, 2017

This group has 1 moderators, 2 members, and 0 members pending approval

Quit Group

Add New Post

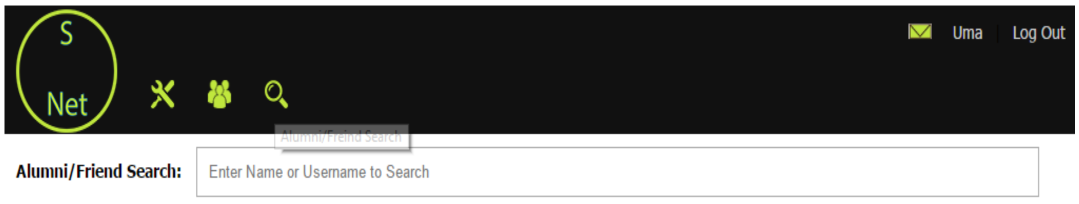
Post

madmax

S-Net - The People's Network © 2017.

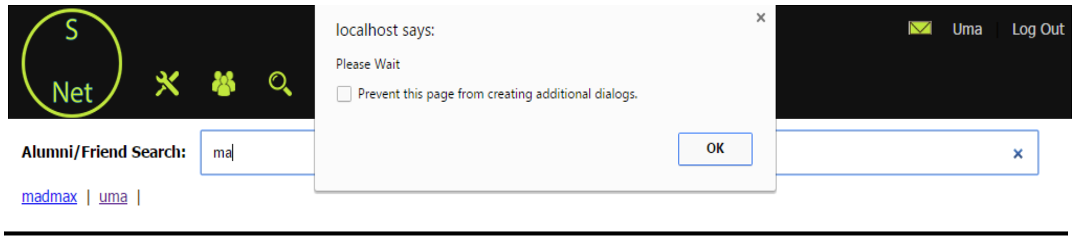
Figure 3.21: Group's Home Page – S-Net

- **Alumni/Friend Search Interface:** Figure 3.22, shows the interface required to search for an alumni/friend in the system and Figure 3.23 shows an example.



Alumni/Friend Search: Enter Name or Username to Search

Figure 3.22: Alumni/Friend Search Page – S-Net



localhost says:

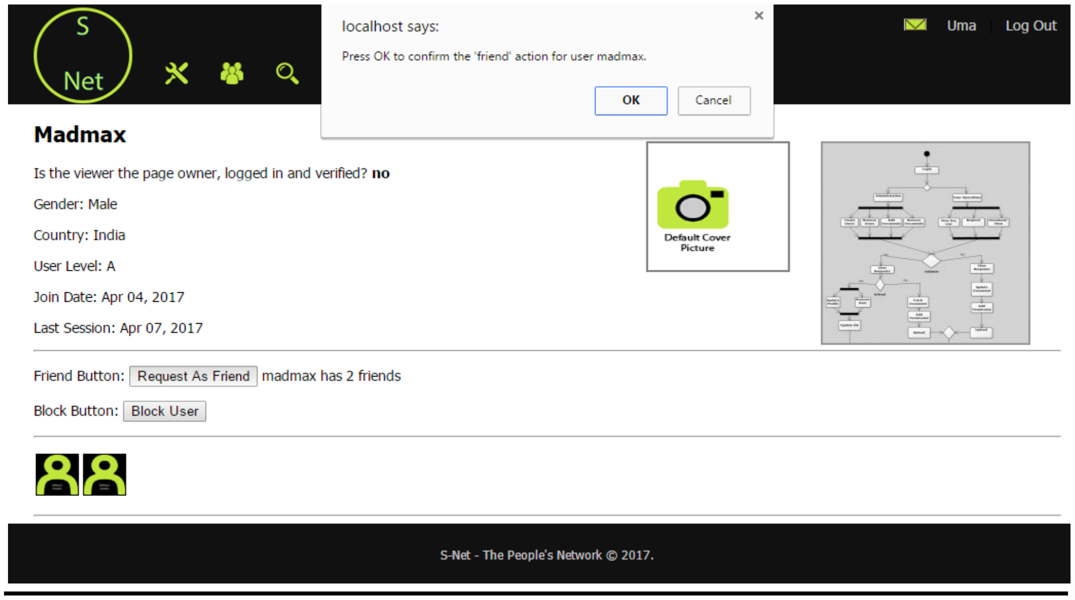
Please Wait

Prevent this page from creating additional dialogs.

OK

Figure 3.23: Search Result for Alumni/Friend – S-Net

- **Friend Request Interface:** Figure 3.24, shows how to send a friend request.



localhost says:

Press OK to confirm the 'friend' action for user madmax.

Madmax

Is the viewer the page owner, logged in and verified? **no**

Gender: Male

Country: India

User Level: A

Join Date: Apr 04, 2017

Last Session: Apr 07, 2017

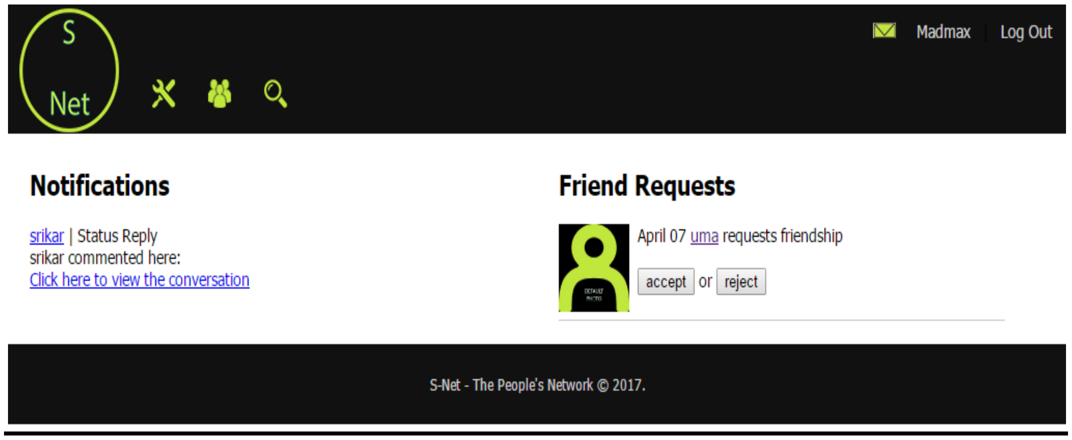
Friend Button: Request As Friend madmax has 2 friends

Block Button: Block User

S-Net - The People's Network © 2017.

Figure 3.24: Friend Request Option – S-Net

- Figure 3.25 shows user "**madmax**" accepting friend request sent in by user "**uma**".



Notifications

srikar | Status Reply
srikar commented here:
Click here to view the conversation

Friend Requests

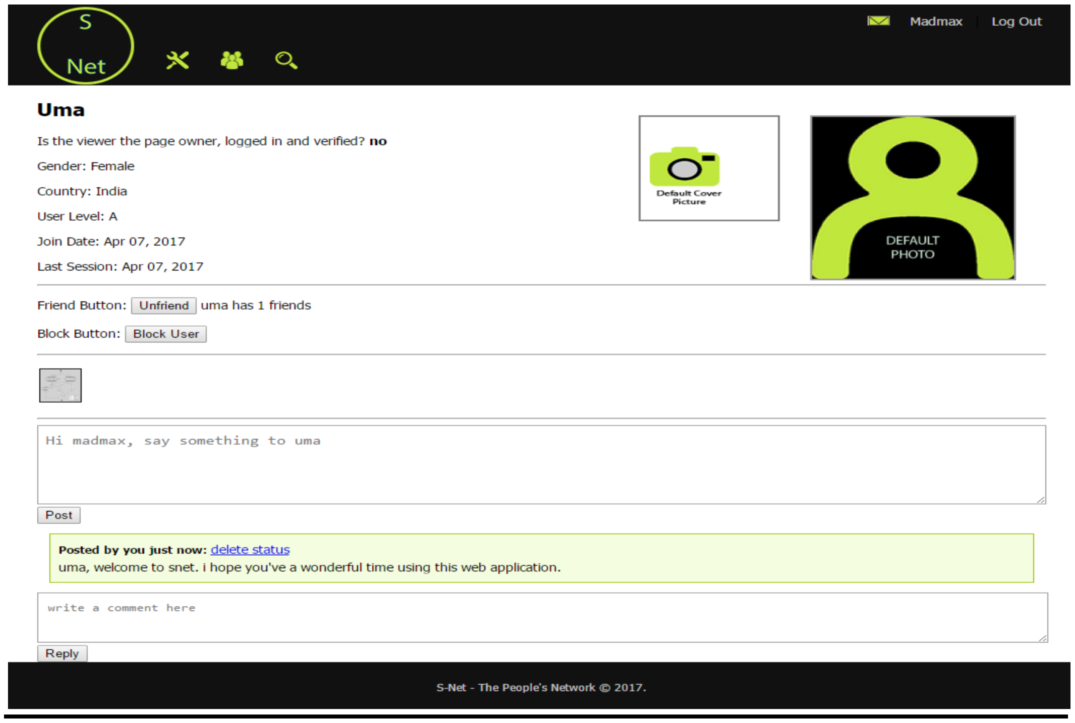
April 07 uma requests friendship

accept or reject

S-Net - The People's Network © 2017.

Figure 3.25: Friend Request Being Accepted – S-Net

- Figure 3.26, shows user "**madmax**", writing content on user "**uma's**" wall. And Figure 3.27, shows user "**uma**" replying back.



Uma

Is the viewer the page owner, logged in and verified? **no**

Gender: Female

Country: India

User Level: A

Join Date: Apr 07, 2017

Last Session: Apr 07, 2017

Friend Button: Unfriend uma has 1 friends

Block Button: Block User

Hi madmax, say something to uma

Post

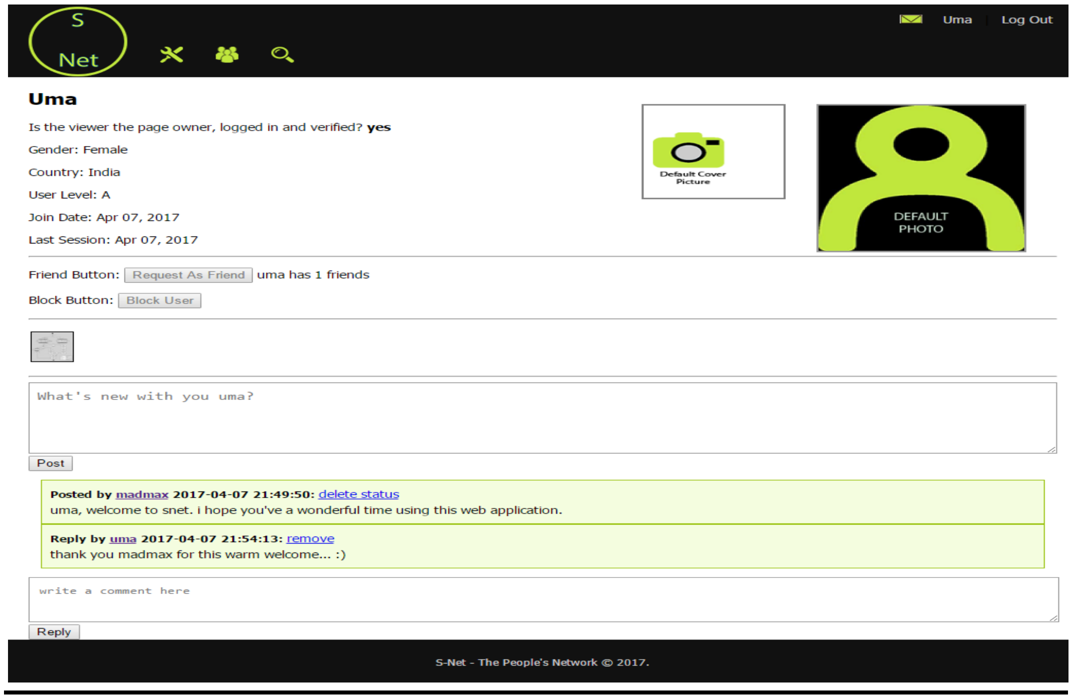
Posted by you just now: delete status
uma, welcome to snet. i hope you've a wonderful time using this web application.

write a comment here

Reply

S-Net - The People's Network © 2017.

Figure 3.26: Posting on Users Wall – S-Net



Uma

Is the viewer the page owner, logged in and verified? **yes**

Gender: Female

Country: India

User Level: A

Join Date: Apr 07, 2017

Last Session: Apr 07, 2017

Friend Button: Request As Friend uma has 1 friends

Block Button: Block User

What's new with you uma?

Post

Posted by madmax 2017-04-07 21:49:50: delete status
uma, welcome to snet. i hope you've a wonderful time using this web application.

Reply by uma 2017-04-07 21:54:13: remove
thank you madmax for this warm welcome... :)

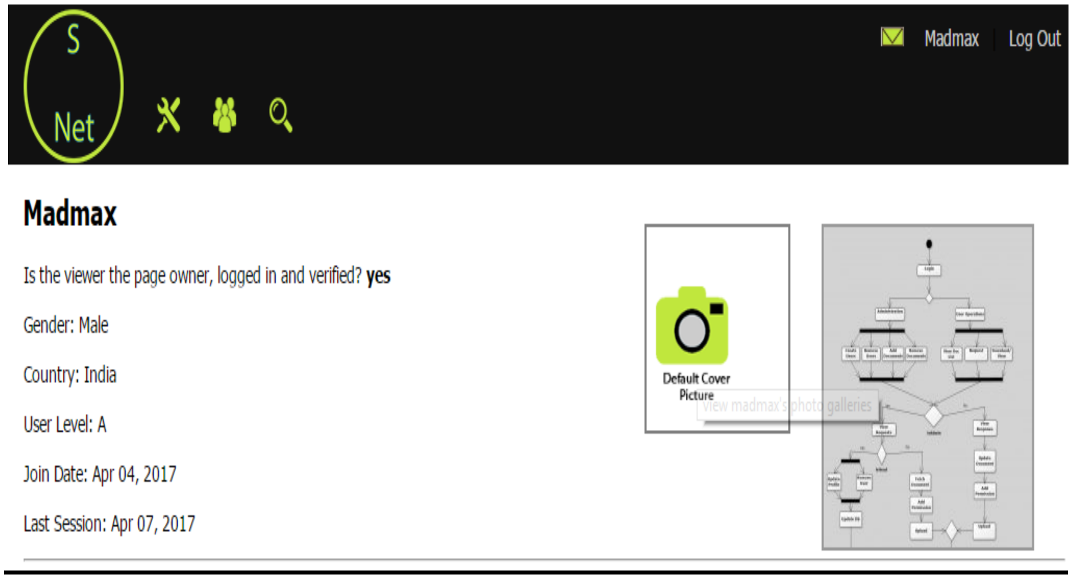
write a comment here

Reply

S-Net - The People's Network © 2017.

Figure 3.27: Replying Back to User Post – S-Net

- **Photo Gallery Interface:** Figure 3.28, shows a personal user gallery system that enables users to store photos.



Madmax

Is the viewer the page owner, logged in and verified? **yes**

Gender: Male

Country: India

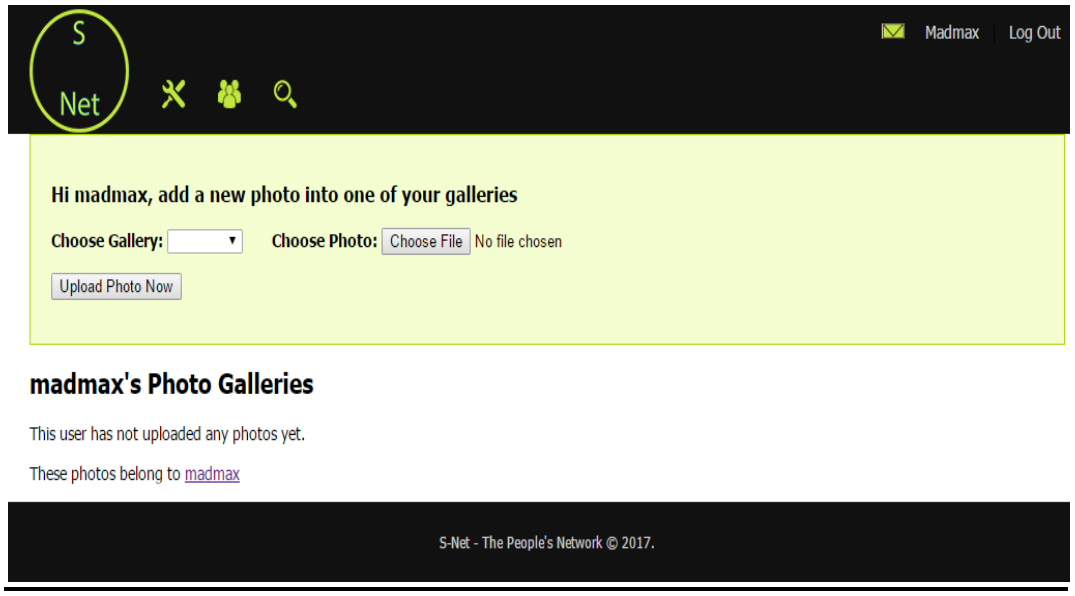
User Level: A

Join Date: Apr 04, 2017

Last Session: Apr 07, 2017

Figure 3.28: Photo Gallery Option – S-Net

- **Upload Photos Interface:** Figure 3.29, shows the interface that is responsible for uploading photos onto the system.



Hi madmax, add a new photo into one of your galleries

madmax's Photo Galleries

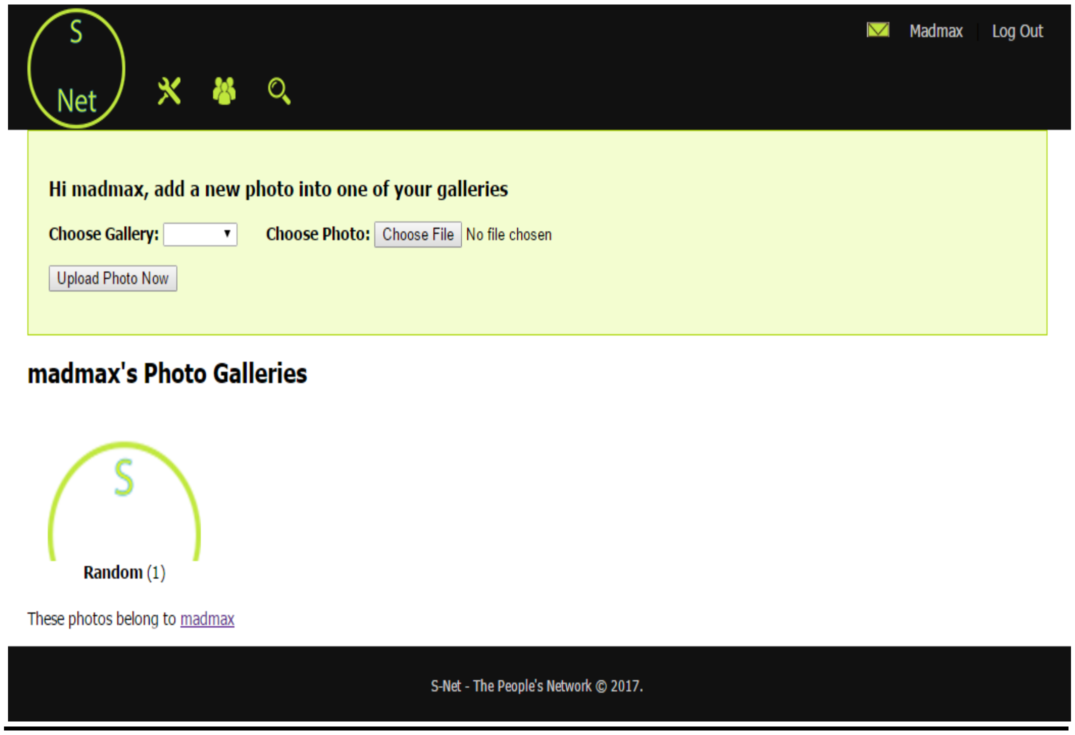
This user has not uploaded any photos yet.

These photos belong to madmax

S-Net - The People's Network © 2017.

Figure 3.29: Upload Photo Option – S-Net

- The following series of snapshots of the web application show how a photo is uploaded and removed from the system.


Madmax Log Out

Hi madmax, add a new photo into one of your galleries

Choose Gallery: Choose Photo: Choose File No file chosen

Upload Photo Now

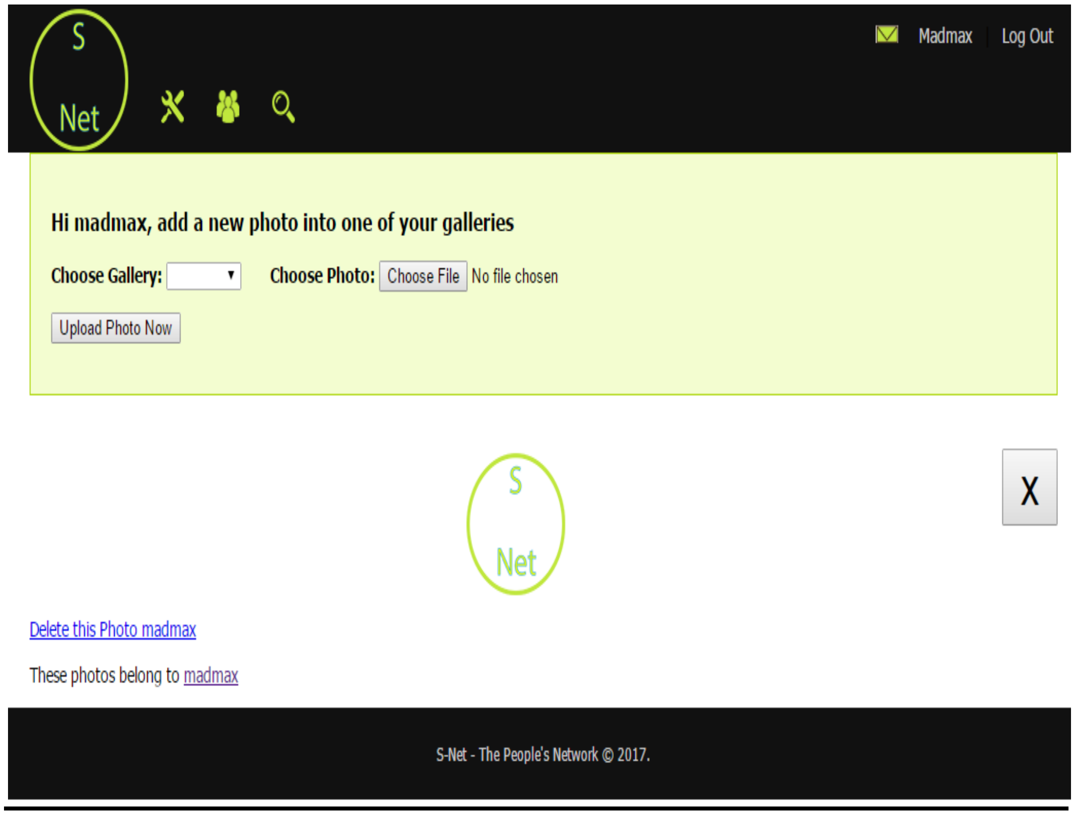
madmax's Photo Galleries

Random (1)

These photos belong to madmax

S-Net - The People's Network © 2017.

Figure 3.30: A Photo Uploaded into the System – S-Net


Madmax Log Out

Hi madmax, add a new photo into one of your galleries

Choose Gallery: Choose Photo: Choose File No file chosen

Upload Photo Now

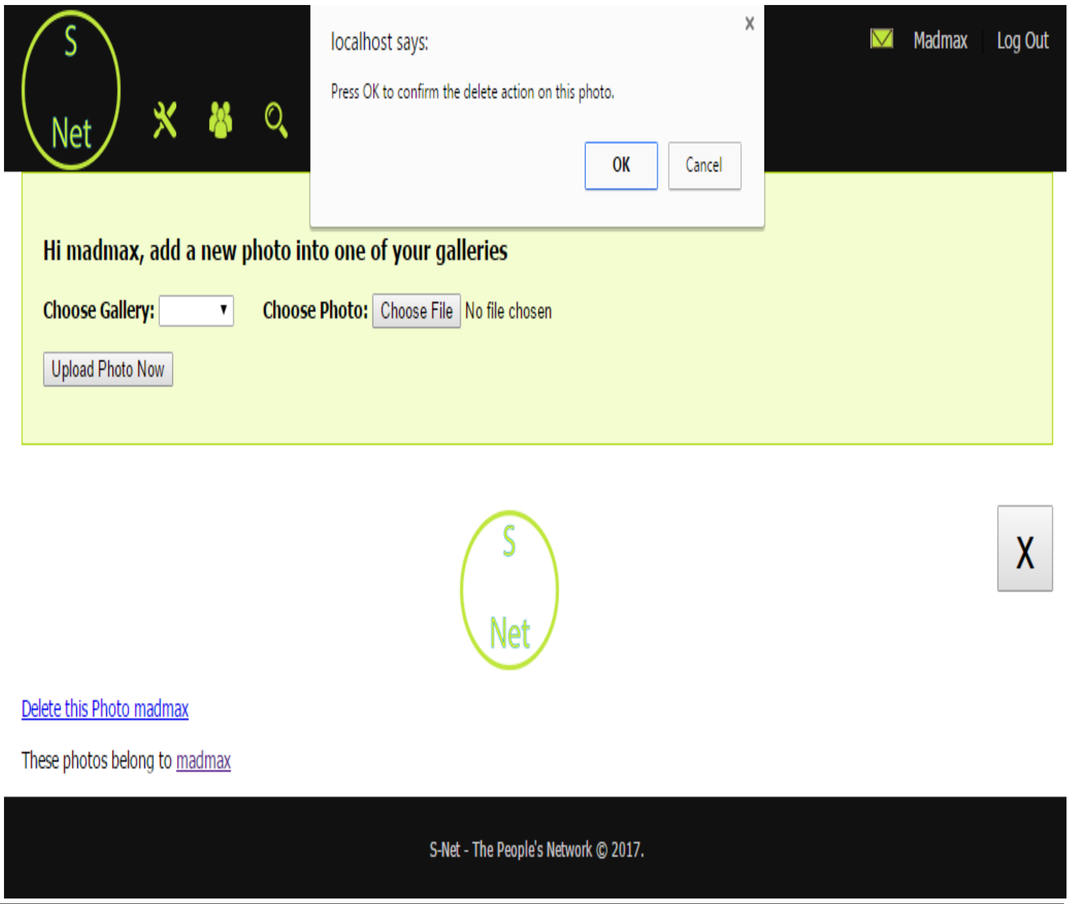
X

Delete this Photo madmax

These photos belong to madmax

S-Net - The People's Network © 2017.

Figure 3.31: Enlarged View of the Photo Uploaded– S-Net


localhost says:

Press OK to confirm the delete action on this photo.

Hi madmax, add a new photo into one of your galleries

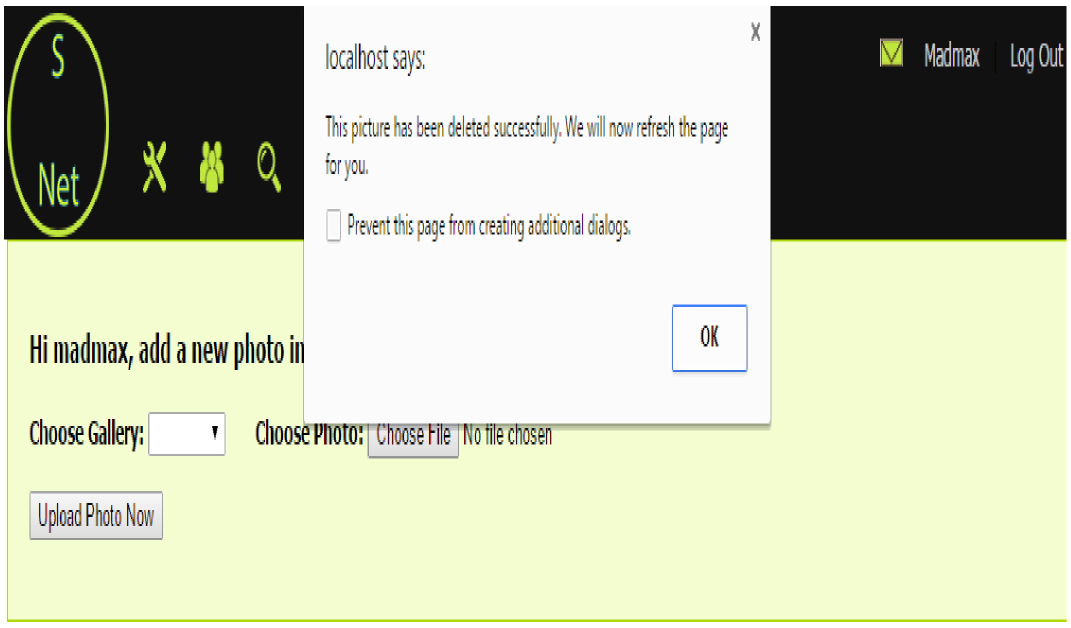
Choose Gallery: Choose Photo: Choose File No file chosen

Delete this Photo madmax

These photos belong to madmax

S-Net - The People's Network © 2017.

Figure 3.32: Uploaded Photo Being Deleted – S-Net


localhost says:

This picture has been deleted successfully. We will now refresh the page for you.

Prevent this page from creating additional dialogs.

Hi madmax, add a new photo in

Choose Gallery: Choose Photo: Choose File No file chosen

Figure 3.33: Uploaded Photo is removed from the System – S-Net

4. TESTING

This chapter follows up with the testing methods that were used during the validation and verification of the system.

4.1 Methodology

The method used while testing this software was different than the conventional testing route followed in the software industry. This testing approach was valuable for the software and was easier because the company was familiar with the methodology.

In this approach, as the specs were ready for a prototype to be shown, the tester started writing his or her code and saw if he or she could obtain the same results as the specs mentioned. This way, the specs were tested on each prototype, and continuous testing was applied. This also helped in minimizing the testing that would have to be implemented at the end of the software lifecycle. In the process, all aspects of the software were tested.

4.1.1 Steps to follow while implementing the methodology

1) Start with a base functionality that you want to implement.
2) Create a document with the detailed requirement definition, an activity diagram with a description of the flow, database tables that would be used and component diagram and description of each component with precondition and tables that would be affected by the component.
3) Give the document to the tester, and work with the tester while he or she writes the code to check if the steps in the document can be implemented and if the result of each use case can be achieved.
4) If the tester finds a step difficult to implement or thinks he or she is missing additional information to implement the functionality, then go to step 2; otherwise, go to step 3.
5) Ask the tester to log all the errors and difficulties he or she encountered while working on the prototype implementation.

6) Once the prototype is done and the results match between the developer's prototype and tester's prototype, work on the other requirement, and expand the prototype to final software.

When the testing approach was implemented the following pros and cons regarding the testing approach were realized.

4.1.2 Pros of using the methodology

- Helps give a better understanding about the requirements.
- Better design at the end of the cycle.
- Reduced testing to be performed at the end of the cycle.
- Documents produced would be of higher quality.

4.1.3 Cons of using the methodology

- The person working on the document should be experienced.
- There are increased time and money involved in testing.
- Different viewpoints for the same problem can lead to different results.

4.2 Interface Testing

This section lists the functional requirements that were used for creating the test cases table, the test cases that were used for verifying the Interface table and the results for the test cases table.

The Table 4.1 below list the Functional requirements for the Interface built for the S-Net – The People's Network, along with a short description of each requirement.

The Table 4.2 below shows the Functional requirements that were used to write the test cases along with the test case numbers for each test case and a short description of the test cases.

FR No.	Short Description
FR 01	S-Net shall have two types of access/login: User/Admin.
FR 02	S-Net shall only be accessible to specified users and Admin with a valid username/password.
FR 03	The users shall be able to view and add/block friends.
FR 06	The Group Admin shall be able to accept/reject group join requests.
FR 07	The Sys. Admin shall be able to view all the users registered in the system and execute cron jobs.
FR 08	The users shall be able to edit their respective profiles.

Table 4.1: Functional Requirement List

FR No.	Test Case No.	Short Description
FR 01	TC 01	To test the Login Interface for Admin and To test the Login Interface for User.
FR 02	TC 02	To test the validation of a user
FR 03	TC 03	To test whether users can view and add/block friends.
FR 06	TC 04	To test Group Admin can accept group join requests.
FR 07	TC 05	To test Sys. Admin can view all the users registered in the system and execute cron jobs.
FR 08	TC 06	To test User, to update profile.

Table 4.2: Test Case Table with Description

The following list include the steps that should be taken by the user, the conditions that should be met for the successful execution of the test case and the end result that should be met for the test cases to pass.

- **Test Case Number: TC 01 and TC 02**
    - **Description:** To test the Login Interface for Admin and to test the Login Interface for User and their validation.
    - **Input:** Username/AdminID and Password.
    - **Valid Range:** Username/AdminID – String/Integer; Password – Alphanumeric.
    - **End Message/Result:**
        - If (**User** == **Valid User**), a message should be displayed Welcome, <**User Name**> (This shall be displayed on the respective screen.).
        - If (**User** != **Valid User**), an error message shall be displayed on the Login interface.
- **Test Case Number: TC 03**
    - **Description:** To test whether users can view and add/block friends.
    - **Input:** User selects/searches for another user from a list.
    - **Output:** User blocked or friend request sent.
    - **End Message/Result:**
        - If (**selection** == **username** and **username** == **exist** and **friend** == **yes**), the user will be notified of the request sent.
        - If (**selection** == **username** and **username** == **exist** and **block** == **yes**), the user will block the user with username.
        - If (**selection** == **username** and **username** == **self**) then user cannot either block/friend herself/himself.
- **Test Case Number: TC 04**
    - **Description:** To test Group Admin can accept group join requests.
    - **Input:** Username, Password.
    - **Output:** Join Requests are either accepted or rejected.
    - **End Message/Result:**
        - If (**login type** == "**user.gadmin**" & **groupbutton**.**clicked** == '**true'** and **joinreq** == **existing** and **accept** == **false**) then display "Group Join Requests Rejected".
        - If (**login type** == "**user.gadmin**" & **groupbutton**.**clicked** == '**true'** and **joinreq** == **existing** and **accept** == **true**) then display "Group Join Requests Accepted".
        - If (**login type** != "**user.gadmin**" & **groupbutton**.**clicked** = '**true'** and **joinreq** == **existing** and **accept** == **true**) then display "Only Group Admin can either accept or reject group join requests".
- **Test Case Number: TC 05**
    - **Description:** To test Sys. Admin can view all the users registered in the system and execute cron jobs.
    - **Input:** AdminId; Password; User-List Selection.
    - **Output:** User List.
    - **End Message/Result:**

        - If (**login type** == "**Sys Admin**" & **UserManagement.clicked** == '**true'** and **list.clicked** == **true** and **userlist.exists == true**), then display users.
        - If (**login type** == "**Sys Admin**" & **UserManagement.clicked** == '**true'** and **list.clicked** == **true** and **userlist**.**exists** == **false**), then display message "No users Exist".
        - If (**login type** == "**Sys Admin**" & **UserManagement.clicked** == '**true'** and **list.clicked** == **true** and **userlist.exists** == **true** & **count (inactive users/groups)** > **0**), then display message "Inactive users/groups Exist, execute cron_job".

- **Test Case Number: TC 06**
    - **Description:** To test User, to update profile.
    - **Input:** Username, Password.
    - **Output:** Message/Alert.
    - **End Message/Result:**
        - If (**login type** == "**user**" & **settings.Clicked** == '**true**' and **form.Details** == **filled** and **update.Clicked** == **true**), then display message "Profile Successfully Updated".
        - If (**login type** == "**user**" & **settings.Clicked** == '**true**' and **form.Details** != **filled** and **update.Clicked** == **true**), then display message "Fill in Form Data".

4.2.1 Results

This section lists the results that were produced by running the test cases. Table 4.3 list out the results that were obtained after executing each test case with valid inputs.

Test Case No.	Expected Results	Actual Result
TC 01	Pass	Pass
TC 02	Pass	Pass
TC 03	Pass	Pass
TC 04	Pass	Pass
TC 05	Pass	Pass
TC 06	Pass	Pass

Table 4.3: Test Cases and Their Respective Results.

5. CONCLUSION AND FUTURE WORK

This Chapter would include the Conclusion reached after creating the current version of the Software in regards to the Objectives of the System.

5.1 Conclusion

The following results have been achieved after the completion of the System which relates back to the Objective for the System.

- ✓ **Allow users to view and either add or block users:** This is achieved when the user gives the permission to a certain user to either be friended or blocked. The selected user then appear to be in the user’s friend list or is blocked completely.
- ✓ **Upload photos to the system so other users can view:** The users can upload the images/photos through an internal operation and save them to their respective folders.
- ✓ **Allow the System administrator to view the users/groups who are inactive:** This is achieved when a System Admin performs a maintenance work, such that system remains in an ideal state.
- ✓ **Approve/Reject the group join requests:** This is achieved when a user who has created a group, a.k.a The Group Administrator sets the group’s join setting as “request group join”, then the group admin has the right to either accept or reject requests.
- ✓ **Allow users to search other users:** This is achieved when a user searches for another user
- ✓ **Allow users to upload profile pictures:** This is achieved when a user uploads a photo/images as his/her profile picture.

5.2 Future Work

The following section discusses the future work that would be implemented in the future releases of the Software.

- ✓ **A Personal Messaging/Instant Messaging:** After much discussion with the QA another need for the system has been realized. A Personal Messaging/Instant Messaging has been proposed that would be included in the next release of the system. It'd allow users to have private conversations.
- ✓ **A User/Friend Suggestion Option:** Another requirement has been realized after the reviewing the current system, that an auto user suggestion option, should be made available to users, to reduce search overhead for the user, and would be included in the next release.
- ✓ **Enable users to add images to user posts:** Another requirement has been realized, and would be included in the next release. This feature enables users to add images to posts.
- ✓ **Enables a group to upload documents:** This requirement is a number one priority, and will be included in the next release of the system. This requirement will helps groups to share documents.

5.3 Project Summary

S-Net – The People's Network is a web-based application for primarily providing a solution to meet organizational needs for having to maintain a strong Alumni.

This application software has been computed successfully and was also tested successfully by taking "test cases". It is user friendly, and has required options, which can be utilized by the user of the system. The goals that are achieved by the software are:

- ✓ Instant access and improved productivity.
- ✓ Optimum utilization of resources.
- ✓ Efficient management of records.
- ✓ Simplification of the operations.
- ✓ Less processing time and getting required information.

BIBLIOGRAPHY

- DeAndrea, D.C., Ellison, N., LaRose, R., Steinfield, C., and Fiore, A. Serious social media: On the use of social media for improving students' adjustment to college.
- Apache HTTPd Project @ https://www.apache.org
- MySQL Open Community Server Project @ https://www.mysql.com
- PHP Manual @ https://www.php.net
- Steam @ https://store.steampowered.com
- Stack Overflow @ https://stackoverflow.com/
- Facebook @ https://www.facebook.com
- Twitter @ https://www.twitter.com
- SNS @ https://en.wikipedia.org/wiki/Social_networking_service

Appendix: Sample Code

Index.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>S-Net - The People's Network</title>
<link rel="icon" href="favicon.ico" type="image/x-icon">
<link rel="stylesheet" href="style/style.css">
</head>
<body>
<?php
include_once("php_includes/check_login_status.php");
include_once("template_pageTop.php"); ?>
<div id="pageMiddle">
	<div style="height: 486px;">
		<p style="width: 500px;"><span style="font-weight: bold; font-size:
42px; color: #CCC">What is S-Net?</span><br/><br/>
			<span style="font-size: 20px">S-Net is a community driven
networking site that makes it easy for you to connect and share with your friends online.
Originally designed and developed by and for college students, S-Net was created in
2017 by S-Net team while they were enrolled at Jawaharlal Nehru Technological
University, Hyderabad.</span>
		</p><br/>
		<p><span style="font-weight: bold; font-size: 42px; color: #CCC">Why
use S-Net?</span><br/></br/>
			<span style="font-size: 20px">Have you ever wondered why
people like using S-Net? After all, there are already a lot of other ways to communicate
online, such as email, instant messaging, and so on. What makes S-Net unique is the
ability to connect and share with the people you care about at the same time.
```

```
                    </span></p>
        </div>
</div>
<?php include_once("template_pageBottom.php"); ?>
</body>
</html>
```

User.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> <?php  echo ucwords($u); ?>'s Profile</title>
<link rel="icon" href="favicon.ico" type="image/x-icon">
<link rel="stylesheet" href="style/style.css">
</head>
<body>
<?php include_once("template_pageTop.php"); ?>
<?php if(isset($_SESSION['username'])){ ?>
<div id="pageMiddle">
  <div id="profile_pic_box" ><?php echo $profile_pic_btn; ?><?php echo $avatar_form;
?><?php echo $profile_pic; ?></div>
  <div id="photo_showcase" onclick="window.location = 'photos.php?u=<?php echo $u;
?>';" title="view <?php echo $u; ?>&#39;s photo galleries">
    <?php echo $coverpic; ?>
  </div>
  <h2><?php echo ucwords($u); ?> <!-- &lt;script&gt; --></h2>
  <p>Is the viewer the page owner, logged in and verified? <b><?php echo $isOwner;
?></b></p>
  <p>Gender: <?php echo $sex; ?></p>
  <p>Country: <?php echo $country; ?></p>
  <p>User Level: <?php echo $userlevel; ?></p>
  <p>Join Date: <?php echo $joindate; ?></p>
  <p>Last Session: <?php echo $lastsession; ?></p>
  <hr />
  <p>Friend Button: <span id="friendBtn"><?php echo $friend_button; ?></span> <?php
echo $u." has ".$friend_count." friends"; ?> <?php echo $friends_view_all_link; ?></p>
  <p>Block Button: <span id="blockBtn"><?php echo $block_button; ?></span></p>
  <hr />
  <p><?php echo $friendsHTML; ?></p>
  <hr />
  <?php include_once("template_status.php"); ?>
</div>
<?php include_once("template_pageBottom.php"); ?>
<?php } ?>
</body>
</html>
```

ChangePassword.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sign Up</title>
<link rel="icon" href="favicon.ico" type="image/x-icon">
<link rel="stylesheet" href="style/style.css">
<script type="text/javascript" src="js/main.js"></script>
<script type="text/javascript" src="js/ajax.js"></script>
</head>
<body>
<?php include_once("template_pageTop.php"); ?>
<div id="pageMiddle" style="height: 476px;">
```

```
  <form name="changepass" id="changepass" onsubmit="return false;">
          <fieldset>
          <legend><h3>Forgot Password?</h3></legend>
    <div>
              <label>Username: </label>
              <input id="username" type="text" onfocus="emptyElement('status')">
    </div>
    <div>
          <label>New Password: </label>
          <input      id="pass1"      type="password"      onfocus="emptyElement('status')"
maxlength="16">
    </div>
    <div>
          <label>Confirm Password: </label>
          <input      id="pass2"      type="password"      onfocus="emptyElement('status')"
maxlength="16">
    </div>
    <div>
          <label>Security Code: </label>
          <input     id="seccode"     type="password"     onfocus="emptyElement('status')"
maxlength="5">
    </div>
    <br/><hr/>
    <p id="status"></p>
    <br /><hr /><br />
    <div>
          <button    id="chgbtn"    class="btn"    style="float:    left;    width:100%;"
onclick="chgpass()">
                      Change Password
          </button>
    </div>
    </fieldset>
  </form>
</div>
<?php include_once("template_pageBottom.php"); ?>
</body>
</html>
```

Settings.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8"/>
<title> <?php  echo ucwords($u); ?>'s Settings</title>
<link rel="icon" href="favicon.ico" type="image/x-icon" />
<link rel="stylesheet" href="style/style.css" />
<script src="js/main.js"></script>
<script src="js/ajax.js"></script>
</head>
<body>
<?php include_once("template_pageTop.php"); ?>
<div id="pageMiddle" style="height: 500px;">
<div style="width:450px; border: 2px #CCC solid; float: left; margin: 10px; padding:
10px; display:inline-block;">
		<form onSubmit="return false" id="updateform">
			<fieldset>
				<legend>Update Profile</legend>
				<label>Name:</label>
				<input type="text"  placeholder="Enter Your Name" id="name"
				value="<?php echo $n;?>"/>
				<label>Alt-E-Mail:</label>
```

```
                        <input id="altemail" type="text" value="<?php echo $alt;?>"
placeholder="Enter Alternate E-Mail Address"/>
                        <label>Contact Number:</label>
                        <input id="phone" type="tel" value="<?php echo $ph;?>"
                        maxlength="10"  placeholder="Enter Contact Number"/>
                        <label>Pass Out Year:&nbsp;</label>&nbsp;<span id="yr"><?php
echo $passout;?></span>
                        <input type="date" id="passout" /><br/>
                        <label>About:</label>
                        <textarea id="about" maxlength="450" placeholder="Write About
Yourself" rows="8"><?php echo $ab;?></textarea>
                        <button id="update" type="button" class="btn"
onClick="updateProfile('update','<?php echo $u;?>');">Update</button>
                        <a href="#" id="edit" class="btn" onClick="edit()" style="text-
decoration: none; text-align: center;" >Edit Profile</a>
                        <button id="reset" type="reset" class="btn" style="float:
right">Reset</button>
                </fieldset>
        </form>
</div>
<div style="width:450px; border: 2px #CCC solid; float: left; margin: 10px; padding:
10px; display:inline-block;">
        <form onSubmit="return false" id="changepassform">
                <fieldset>
                        <legend>Update Profile</legend>
                        <label>Current Password:</label>
                        <input type="password"  placeholder="Enter Your Current
Password" id="curpass"/>
                        <label>New Password:</label>
                        <input id="newpass" type="password" placeholder="Enter New
Password"/>
```

```
                              <label>Confirm Password:</label>
                              <input id="confpass" type="password" placeholder="Confirm New
Password"/>
                              <button id="update" type="button" class="btn" style="width:
100%;" onClick="changePass('changepass','<?php echo $u;?>');">Change
Password</button>
                              <p style="color: red;" align="center">Login Once Again After the
Password Is Changed</p>
                    </fieldset>
          </form>
</div>
</div>
<?php include_once("template_pageBottom.php"); ?>
</body>
</html>
```

Login.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Log In</title>
<link rel="icon" href="favicon.ico" type="image/x-icon">
<link rel="stylesheet" href="style/style.css">
</head>
<body>
<?php include_once("template_pageTop.php"); ?>
<div id="pageMiddle" style="height: 476px;">
  <!-- LOGIN FORM -->
  <form id="loginform" onsubmit="return false;">
  <fieldset>
```

```
            <legend><h3>Login Here</h3></legend>
            <div>
                  <label>Username:</label>
            <input type="text" id="username" onfocus="emptyElement('status')"
maxlength="88">
            </div>
  <div>
            <label>Password:</label>
            <input type="password" id="password" onfocus="emptyElement('status')"
maxlength="100">
  </div>
  <div>
            <label>Security Code:</label>
            <input type="password" id="seccode" onfocus="emptyElement('status')"
maxlength="5">
  </div>
  <br /><hr /><br/>
  <div>
            <button id="loginbtn" class="btn"
                  style="float: left;"onclick="login()">Log In</button>
            <a href="chgpass.php" style="float: right;" class="btn">Forgot Your
Password?</a>
  </div>
  <br /><br /><br /><hr/>
  <p id="status"></p>
  </fieldset>
 </form>
 <!-- LOGIN FORM -->
</div>
<?php include_once("template_pageBottom.php"); ?>
</body>
```

```
</html>
```

SignUp.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sign Up</title>
<link rel="icon" href="favicon.ico" type="image/x-icon">
<link rel="stylesheet" href="style/style.css">
<script type="text/javascript" src="js/main.js"></script>
<script type="text/javascript" src="js/ajax.js"></script>
</head>
<body>
<?php include_once("template_pageTop.php"); ?>
<div id="pageMiddle">
  <form name="signupform" id="signupform" onsubmit="return false;">
        <fieldset>
        <legend><h3>Sign Up Here</h3></legend>
    <div>
            <label>Username: </label>
            <input id="username" type="text" onblur="checkusername()"
onkeyup="restrict('username')" maxlength="16">
            <span id="unamestatus"></span>
    </div>
    <div>
        <label>Email Address: </label>
        <input id="email" type="text" onfocus="emptyElement('status')"
onkeyup="restrict('email')" maxlength="88">
```

```
  </div>

  <div>

      <label>Create Password: </label>

      <input id="pass1" type="password" onfocus="emptyElement('status')"
maxlength="16">

  </div>

  <div>

      <label>Confirm Password: </label>

      <input id="pass2" type="password" onfocus="emptyElement('status')"
maxlength="16">

  </div>

  <div>

      <label>Gender: </label>

      <select id="gender" onfocus="emptyElement('status')">

            <option value=""></option>

              <option value="m">Male</option>

              <option value="f">Female</option>

      </select>

  </div>

  <div>

      <label>Country: </label>

      <select id="country" onfocus="emptyElement('status')">

            <?php include_once("template_country_list.php"); ?>

      </select>

  </div>

  <br/><hr/>

  <p id="status"></p>

  <br /><hr /><br />

  <div id="terms" style="display:none;">
```

```
                <h3>S-Net Terms Of Use:</h3>

                <p>1. Play nice here.</p>

                <p>2. Take a bath before you visit.</p>

                <p>3. Brush your teeth before bed.</p>

    </div>

    <br /><hr /><br />

    <div>

            <button id="signupbtn" class="btn" style="float: right;" onclick="signup()">

                        Create Account

            </button>

            <a href="#" class="btn" style=" float: left;"onclick="return false"
onmousedown="openTerms()">

            View the Terms Of Use

            </a>

    </div>

    </fieldset>

  </form>

</div>

<?php include_once("template_pageBottom.php"); ?>

</body>

</html>
```