

ABSTRACT

The project entitled “Smart Share – The Repository of Information”, is a web application developed for managing, creating/storing, versioning, reporting, and archiving of e-files. The idea behind, “Smart Share – The Repository of Information”, is the “DOCUMENT MANAGEMENT SYSTEM”, that is efficient, time saving, and an easy way to report, view and control the version of a file. 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 information to access. Thus, a DMS at college level can improve efficiency and effectiveness of research and also can improve knowledge sharing among students and get new idea from other researches. Students need a well-organized knowledge repository in order to reuse information throughout their research, which creates new opportunities for collaboration, coordination, and information exchange among students that work on a construction project and consequently reduce costs and response times. The entire service is delivered through Internet, browse/search in any location and access documents. Thus “Smart Share – The Repository of Information”, will automate away most of the drudgery involved in keeping an annotated history of files and avoid modification conflicts.

Table of Contents

ACKNOWLEDGMENTS.....	v
ABSTRACT.....	vi
LIST OF FIGURES.....	ix
LIST OF TABLES.....	xi
1. INTRODUCTION.....	1
1.1. Motivation.....	1
1.2. Aim of Software.....	1
1.3. Document Organization.....	2
2. OBJECTIVES.....	3
2.1. Product Perspectives.....	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.....	6
2.7. Design Constraints.....	7
2.8. Software System Quality Attributes.....	8
2.9. Programming Environment.....	8
3. IMPLEMENTATION.....	17
3.1. Detailed Scope.....	17
3.2. Static Decomposition and Dependency Description.....	17
3.3. The Database Design.....	32
3.4. Interface Description.....	35
4. TESTING.....	48
4.1. Methodology.....	48
4.2 Interface Testing.....	49

Table of Contents

5. CONCLUSION and FUTURE WORK.....	54
5.1. Conclusion.....	54
5.2. Comparison.....	54
5.3. Future Work.....	55
BIBLIOGRAPHY.....	57
Appendix: Sample Code.....	58

LIST of FIGURES

Figure No.	Description	Page No.
2.1	3-Tier Architecture	9
2.2	MVC and 3-Tier Arch.	9
2.3	Logical View of MySQL's Architecture	12
3.1	Use Case of Smart Share	18
3.2	Class Diagram of Smart Share	21
3.3	Sequence Diagram for Login	25
3.4 (a)	Sequence Diagram for User Management	26
3.4 (b)	Sequence Diagram for User Management	27
3.5	Sequence Diagram for User Operations	28
3.6	Sequence Diagram for Document Request	29
3.7	Sequence Diagram for Document Processing	30
3.8	Sequence Diagram for Logout	31
3.9	Entity Relationship Diagram of Smart Share	32
3.10	Index Page of Smart Share	36
3.11	Administrator Login	36
3.12	Head Administrator's Operations	37
3.12.1	System Status Tab	37
3.12.2	User Request Tab	37
3.12.3	Create User(s) Tab	38
3.12.4	Remove User(s) Tab	38
3.13	Document Administrator's Operations	39
3.13.1	Document Status Tab	39
3.13.2	View Documents Tab	40
3.13.3	Document Request Tab	40
3.13.4	Grant Permission(s) Tab	41
3.13.5	Revoke Permission(s) Tab	41

LIST of FIGURES

Figure No.	Description	Page No.
3.13.6	Review Tab	42
3.13.7	Report Center Tab	42
3.14	User Login Interface	43
3.15	Forgot Something? Interface	43
3.16 (a)	New User Registration Interface	44
3.16 (b)	Success Message after Request is Posted	44
3.17	Check Request Status Interface	45
3.18	User's Home Page	45
3.19	User's View of Documents in the System	46
3.20	User's Document Request Tab	46
3.21	Access Restricted Area Tab	47
3.22	Suggestion/Report Tab	47

LIST of TABLES

Table No.	Description	Page No.
3.1	Structure of Administration Table.	33
3.2	Structure of User Request Table.	33
3.3	Structure of Users' Table.	33
3.4	Structure of Report Table.	34
3.5	Structure of Document Table.	34
3.6	Structure of Document Assign Table.	34
3.7	Structure of Document Review User Upload Table.	35
3.8	Structure of Document Request Table.	35
3.9	Structure of Document Reviewed Table.	35
4.1	Functional Requirement List.	50
4.2	Test Case Tables with Description.	50
4.3	Test Cases and Their Respective Results.	53
5.1	Smart Share vs. OpenKM©.	55

1. INTRODUCTION

These days, a lot of company documents are stored on computers and servers which can be easily accessible to users at any place as long as they get permission to view the documents. Companies use these documents in a number of ways: to simply store them at one location, to train users, or to maintain training documents and logs.

The same need has come along since the plans for expansion came into being. This needed to developing a “Smart Share – The Repository of Information”, a.k.a. Document Management System, which can be used to store the college’s documents for the standard operating procedure used and to reduce the paperwork that needs to be maintained.

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 documents produced in the College. It is the task of Quality Assurance Specialist to maintain all the documents necessary for later audit trails by either the College or Inspection Bureau. These documents can be Protocols, Statistical Analysis Plans or Standard Operating Procedures or Simple Documents. Maintaining such documents as manual paper work can result in reduce performance and even lead to errors. Hence a need for a system that can be used for storing such documents and can also be used for assigning tasks to users for making revisions to such documents arise.

1.2. Aim of Software

This software is developed in order to help everyone at SVIT to keep track of the document changes and to reduce the paperwork in the process. Smart Share allows the users to view and save different versions of the same document, and allows the Admin to assign tasks for users to make changes to the current version. The system users are able to view any tasks assigned to them for document revision and to upload the changed documents. Smart Share, 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, Activity 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 Smart Share – The Repository of Information. 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 Smart Share.

2.1. Product Perspective

Smart Share – The Repository of Information is a web-based system and can be accessed using: Internet Explorer 8.0 and above, Mozilla Firefox 2.0, and Google Chrome.

2.1.1. User Interface

The two interface types in the Document Management System are as follows:

1. **User Interface:** Users shall be able to view the tasks assigned to them and to upload the new/revised documents to the system.
2. **Admin Interface:** The Admin shall be able to assign tasks to users for creating new documents/making changes to revised documents, approving/rejecting the new/changed documents, and viewing the detail summary for each document history.

2.1.2. Hardware Interface

The Document Management System shall provide minimum hardware requirements. The following hardware configurations are required for a PC using the Document Management System a.k.a. Smart Share:

- 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:

- Register Users and Boot/Remove Users.
- Controlled Access to Resource Information.
- Upload/Remove - Files/Documents.
- Query Processing and Grievance/Suggestions Reporting.
- Grant Permissions.
- Customizability and Extensibility.

2.3 User Classes and Characteristics

The users of the Document Management System, based on their roles, are Employees, Students (Users) and the Administrators (Quality Assurance Specialist). 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 and android platform.

- **Admin:** The Admin is the College's employed Quality Assurance Specialist. 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 all the training documents required for the system. The Admin should be able to perform the following functions:
 - Can assign tasks to specific users for creating new documents/making revisions to the current documents.
 - Accept/reject uploads from the user.
 - View the entire history of the documents' revisions and changes.

- **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:
 - View tasks assigned to them.
 - Download required documents for making modifications.
 - Upload new/updated documents.

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:

- Users and Admin are accustomed to the paper-based system and would require training for using the Document Management System.
- 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 Smart Share – The Repository of Information. 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 Document Management System shall have two types of access/login:
 - a. User.
 - b. Admin.
 - 2. **FR 02:** The Document Management System shall only be accessible to specified Users and Admin with a valid username/password.
- **User: View Documents:** The users shall be able to see their designated page after login and to select the view task from the tab interface. The user shall be able to view their assigned tasks.
 - 1. **FR 03:** The users shall be able to view and change the documents assigned to them.
- **User: Upload/Download Document:** After logging into the system, the users shall be able to see their designated pages. After viewing their task, the users should be able to view the documents added to the system. The users shall be able to download the document they need to update. The Admin shall be able to upload a new document or a revised document.
 - 1. **FR 04:** The user shall be able to view the documents added to the system

2. **FR 05:** The user shall be able to download existing documents.
 3. **FR 06:** The Admin shall be able to upload new/revised documents.
- **Admin: Assign Document**
 1. **FR 07:** The Admin shall be able to view all the users registered in the system.
 2. **FR 08:** The Admin shall be able to select users and to assign them a document.
 - **Additional Functional Requirements**
 1. **FR 09:** The Admin shall be to view the entire history for a document's revisions.
 - **User: Edit Profile/Add/Delete Users:** After logging into the system, the users shall be able to see their designated pages. The Admin should be able to view all the users enrolled in the system. The system shall allow the Admin to edit/add users enrolled in the system. The system shall allow all the users to edit their personal profiles.
 1. **FR 10:** The system allows the users to edit their personal profiles.
 2. **FR 11:** The system allows the Admin to create new users.
 3. **FR 12:** The system allows the Admin to remove users.

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 file 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 Smart Share.
 2. **QA 02:** Based on the type of user, Smart Share shall provide a user-specific interface.
- **Correctness:**
 1. **QA 03:** The assigned task should be received by the specified user.
- **Availability:**
 1. **QA 04:** The system shall be made available to the User/Admin year round.
- **Robustness:**
 1. **QA 05:** The system shall be able to save the documents. If the database server goes down, the system should generate an error message for the users (Admin and 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

Smart Share is based on the concepts or 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.

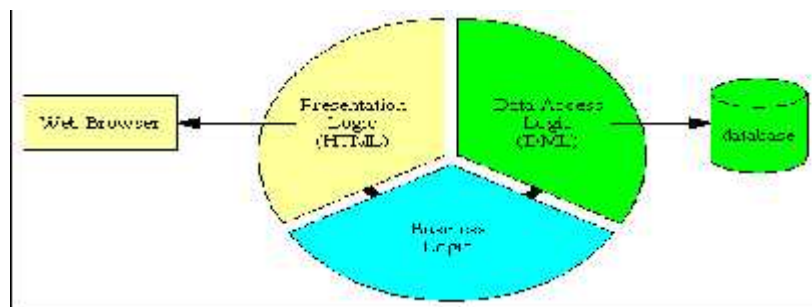


Figure 2.1: 3-Tier Architecture.

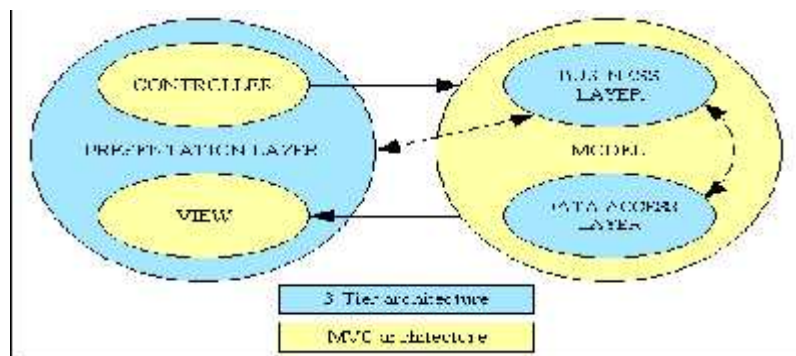


Figure 2.2: MVC and 3-Tier Architecture.

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.

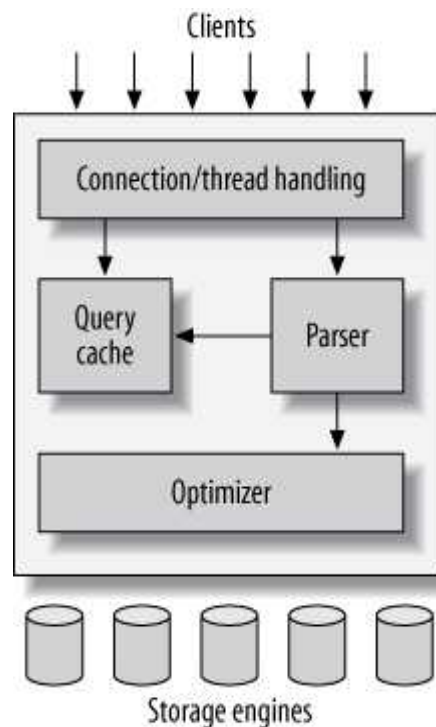


Figure 2.3: logical view of MySQL's Architecture.

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

3. IMPLEMENTATION

This chapter includes the detailed design used to build the Document Management System. The system's mock 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 customer profile and case.
 - Add users.
 - Update user profile
 - Add documents.
 - Give document rights to users.
- In the second delivery, the application must be able to review documents uploaded by users.
- After the first two phases of the project have been completed and thoroughly tested, the system would be enhanced into a joint Document Management and User Training System.

3.2 Static Decomposition and Dependency Description

This section contains the Smart Share'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 are as 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.

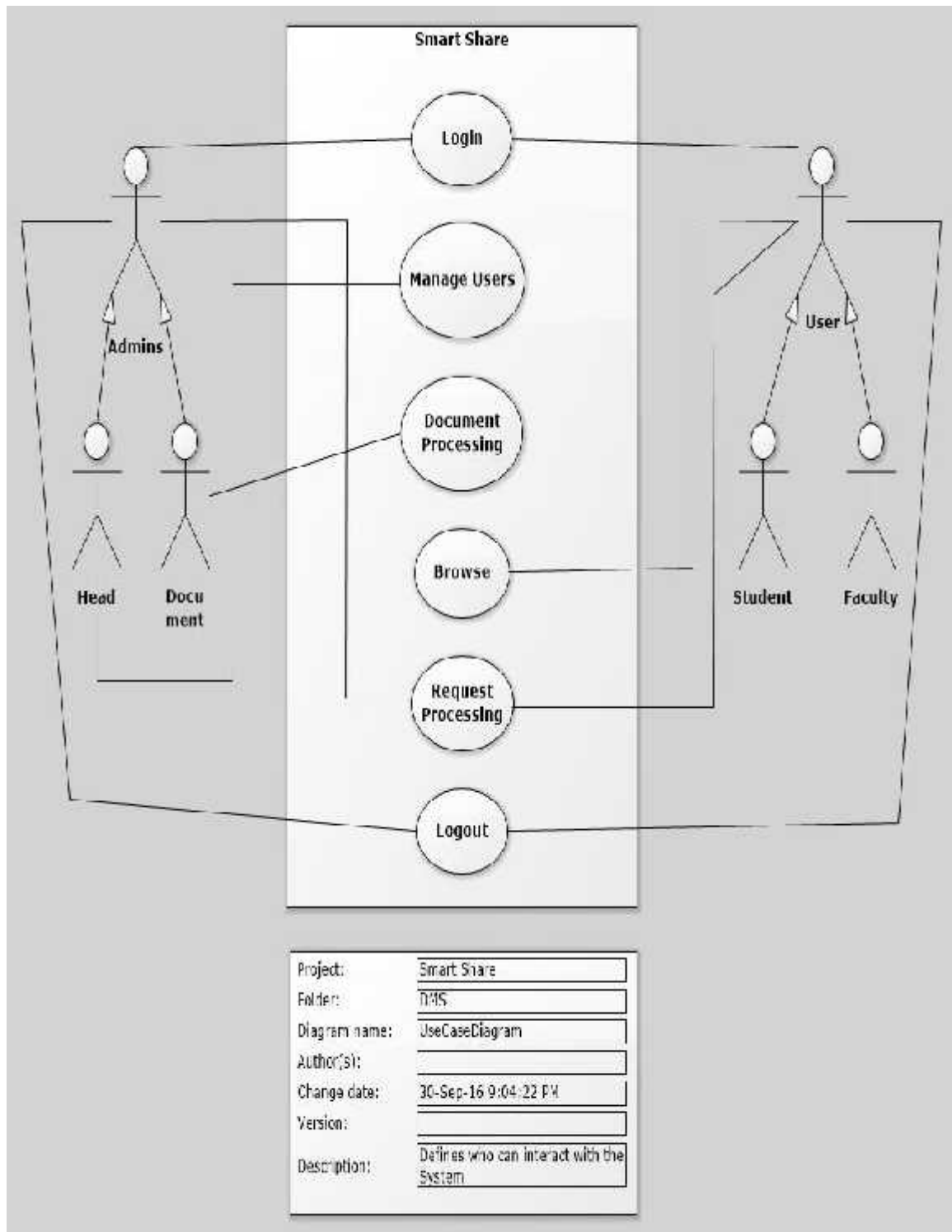


Figure 3.1: Use Case of Smart Share

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:** Smart Share.
 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:** Smart Share.
 2. **Use Case Name:** Manage Users.
 3. **Use Case Description:** This details the user management process of the system.
 4. **Primary Actor:** Head Administrator.
 5. **Precondition:** The system should have either pending or answered user requests.
 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:** Smart Share.
 2. **Use Case Name:** Document Processing.
 3. **Use Case Description:** This details the document management process of the system.
 4. **Primary Actor:** Document Administrator.
 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:** Smart Share.
 2. **Use Case Name:** Browse.
 3. **Use Case Description:** This details the user aspect of document management process; that is, it enables users to view/download documents, tasks assigned to them.
 4. **Primary Actor:** User.
 5. **Precondition:** only authorized users, who are required to logon to the system; are allowed to view or download documents assigned or to do simple download.
 6. **Post Condition:** The system needs to maintain its idle state irrespective of the user operation performed.
- **Use Case Number:** US-005.
 1. **Application:** Smart Share.
 2. **Use Case Name:** Request Processing.
 3. **Use Case Description:** This details the system's ability to process request(s) from users and service the request(s) if found sound.
 4. **Primary Actor:** User/Admin.
 5. **Precondition:** The system should store each request that has a sound reason.
 6. **Post Condition:** The request is either Serviced or Rejected based on the reason submitted by a user.
- **Use Case Number:** US-006.
 1. **Application:** Smart Share.
 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).

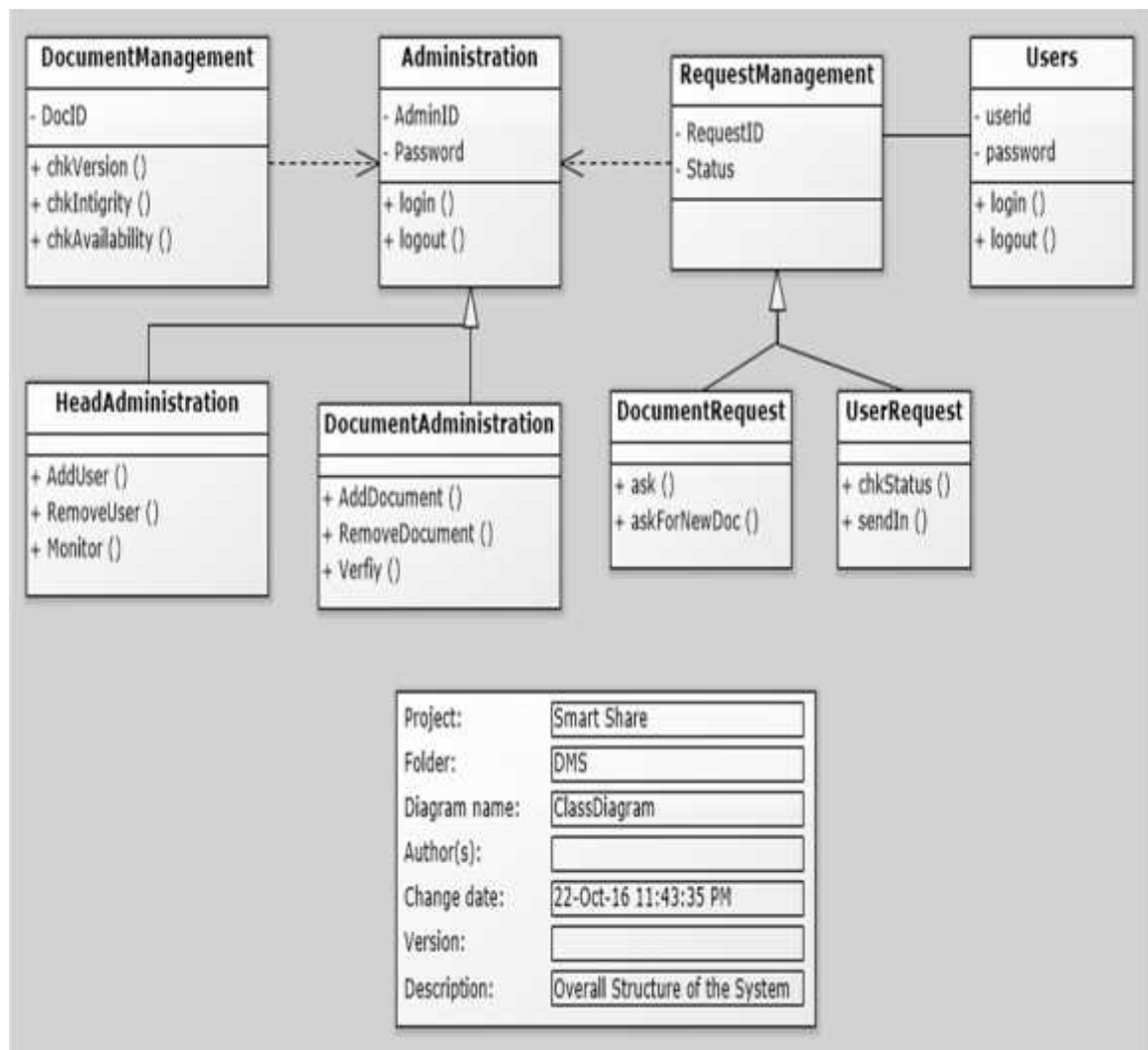


Figure 3.2: Class Diagram of Smart Share.

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.

- **Administration:** This class is used by the administrators of the system, and it's the most important and a main class, that is responsible for checking the proper functioning of the system with the defined attributes and methods.
 - i. **Attribute(s):** a) **AdminID**; b) **Password**;
 - ii. **Method(s):**
 1. **Login():** This method allows an administrator to login and gain access to the system. Based on the role, the respective screen is displayed to the administrator.
 2. **Logout():** This method allows an administrator to logout of the system.
 - iii. Two classes are, namely **Head-Administration** and **Document-Administration** are sub classed; and two classes are dependent on this class for its usage, namely the **Request-Management** and **Document-Management**.
- **Head-Administration:** This class represents the portion of the system, which deals with the creation, removal and monitoring of user activities.
 - i. **Method(s):**
 1. **AddUser():** This method will save the details provided, in the database. The user login will be functional immediately after the execution of this method.
 2. **RemoveUser():** This method will remove the details provided from the database. The user login will be nonfunctional immediately after the execution of this method.
 3. **Monitor():** This method allows the Head-Administrator to observe the activities or malpractices carried out each user of the system and a necessary action is taken to maintain the idle state of the system.
- **Document-Administration:** This class represents the core or the heart of the system. Here a Document-Administrator uses the following methods to maintain the idle state of the system.
 - i. **Method(s):**
 1. **AddDocument():** This method will save the details provided such as filename, files size and path, in the database. The users will be

able to use documents uploaded, immediately after the execution of this method.

2. **RemoveDocument():** This method will remove the details provided for documents from the database. The users' can no longer use/download/view the document; immediately after the execution of this method.
 3. **Verify():** This method enables the Document-Administrator to check/review the documents that are uploaded by the users.
- **Document Management:** This class defines the idle state of the system; that is, it helps to maintain the system that are defined under nonfunctional requirements.
 - i. **Method(s):**
 1. **chkVersion():** This method is used when the administrators want to check the whole document history.
 2. **chkIntegrity():** This method is used when the administrators want to determine the Uprightness and the Reliability of the document(s).
 3. **chkAvailability():** This method is used when the administrators want to determine whether a document is available, so that it can be assigned to a new user for further modifications.
 - ii. **Attribute(s):**
 1. **DocID,** This attribute defines a unique number assigned to each document.
 - **Request Management:** This class details about the system's ability to accept and service request(s).
 - i. **Attribute(s):**
 1. **RequestID:** is a numeric value, which is assigned to each request raised by a user.
 2. **Status:** is a string value, that's returned to a user for a particular request; that is, it returns values such as "PENDING", "SERVICED", "REJECTED".
 - ii. Two classes namely, **Document-Request** and **User-Request** are subclassed from this class.

- **Document-Request:** This class is responsible for servicing the user's need for documents.
 - i. **Method(s):**
 1. **ask():** This method is used by a user, mainly for requesting permissions to change the document.
 2. **askForNewDoc():** This method is used by a user, this mainly requests for a document that is not available in the system.
- **User-Request:** This class is responsible for servicing basic user needs.
 - i. **Method(s):**
 1. **chkStatus():** This method returns the current state/position of a new user request; that is, the registration of a user.
 2. **sendIn():** This method stores a new user request in the database with provided fields, and is called when a new user wants to use the system.
- **Users:** This class defines all user related operations.
 - i. **Attribute(s):**
 1. **userid:** This attribute defines a unique numeric value, that distinguishes one user from another.
 2. **password:** This attribute is the unique alphanumeric value, that's created when the administrator create a user and assigns a role. This value needs to be changed when the user first logs into the system.
 - ii. **Method(s):**
 1. **Login():** This method allows a user to login and gain access to the system.
 2. **Logout():** This method allows an administrator to logout of 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

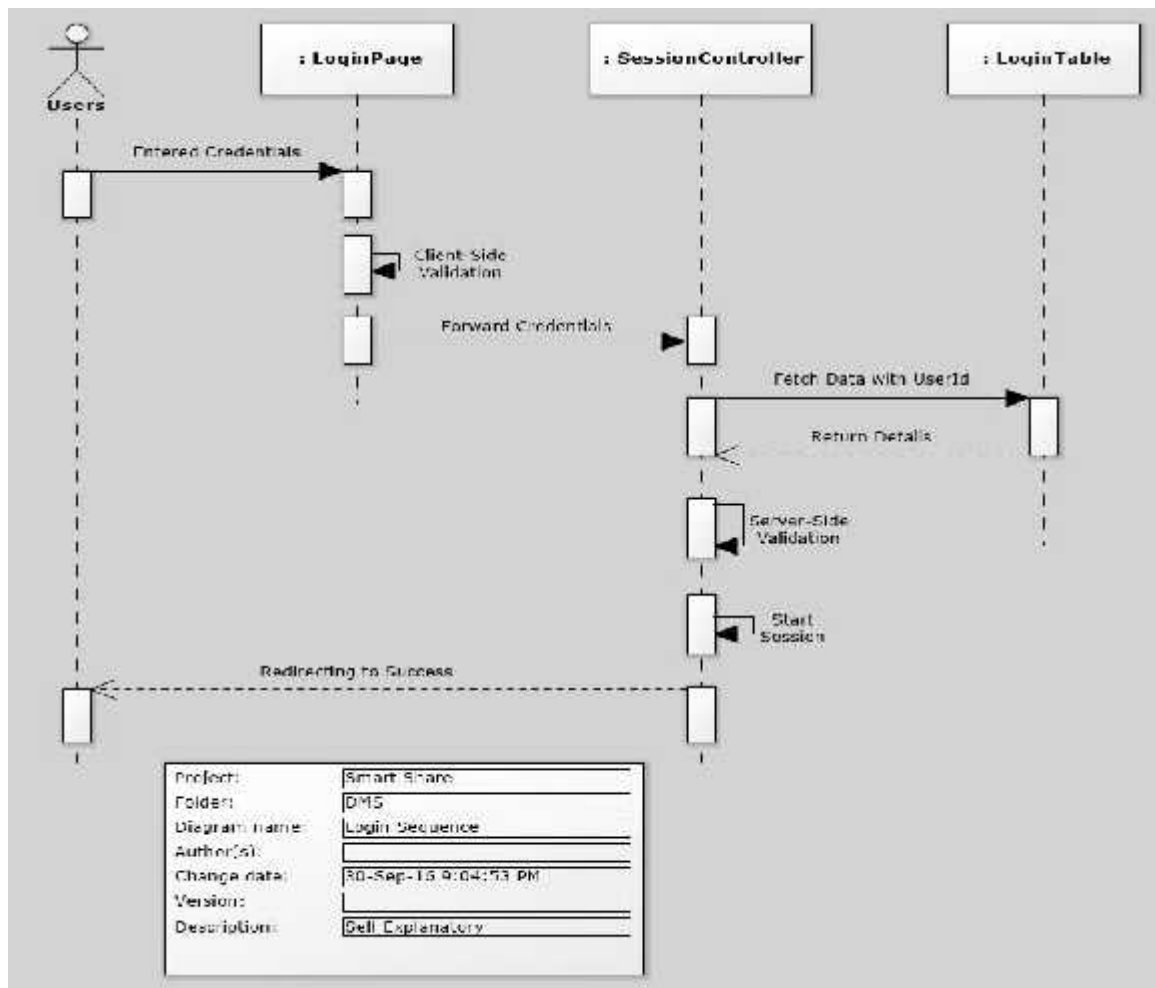


Figure 3.3: Sequence Diagram for Login

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 updated with data into 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 Manage Users

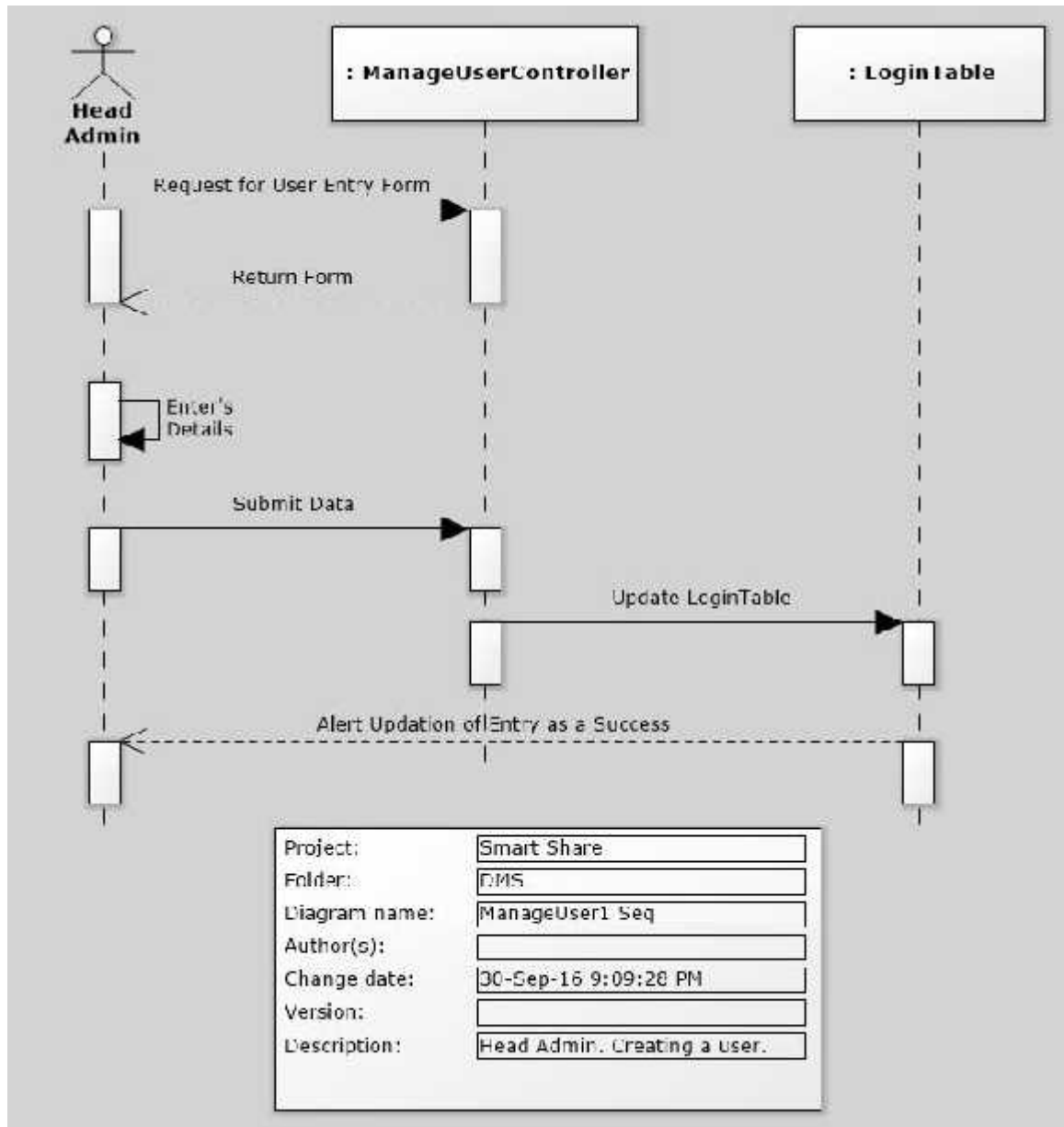


Figure 3.4 (a): Sequence Diagram for User Management

The following is the explanation of the Figure 3.4 (a).

- **Primary Actor:** Head Administrator.
- **Basic Flow:**
 - I. Admin enters the user submitted credentials into the user registration form.
 - II. A basic client side validation is done.
 - III. The form is then submitted. Upon submission, the database is updated.
 - IV. Admin is then alerted with a success message.
- **Exceptional Flow:**
 - I. Admin enters the user submitted credentials into the user registration form.
 - II. A basic client side validation is done upon failure errors are alerted.
 - III. Credentials are then updated with data into the database, on failure reports an error which are alerted.
 - IV. Admin is then alerted with a failure message.

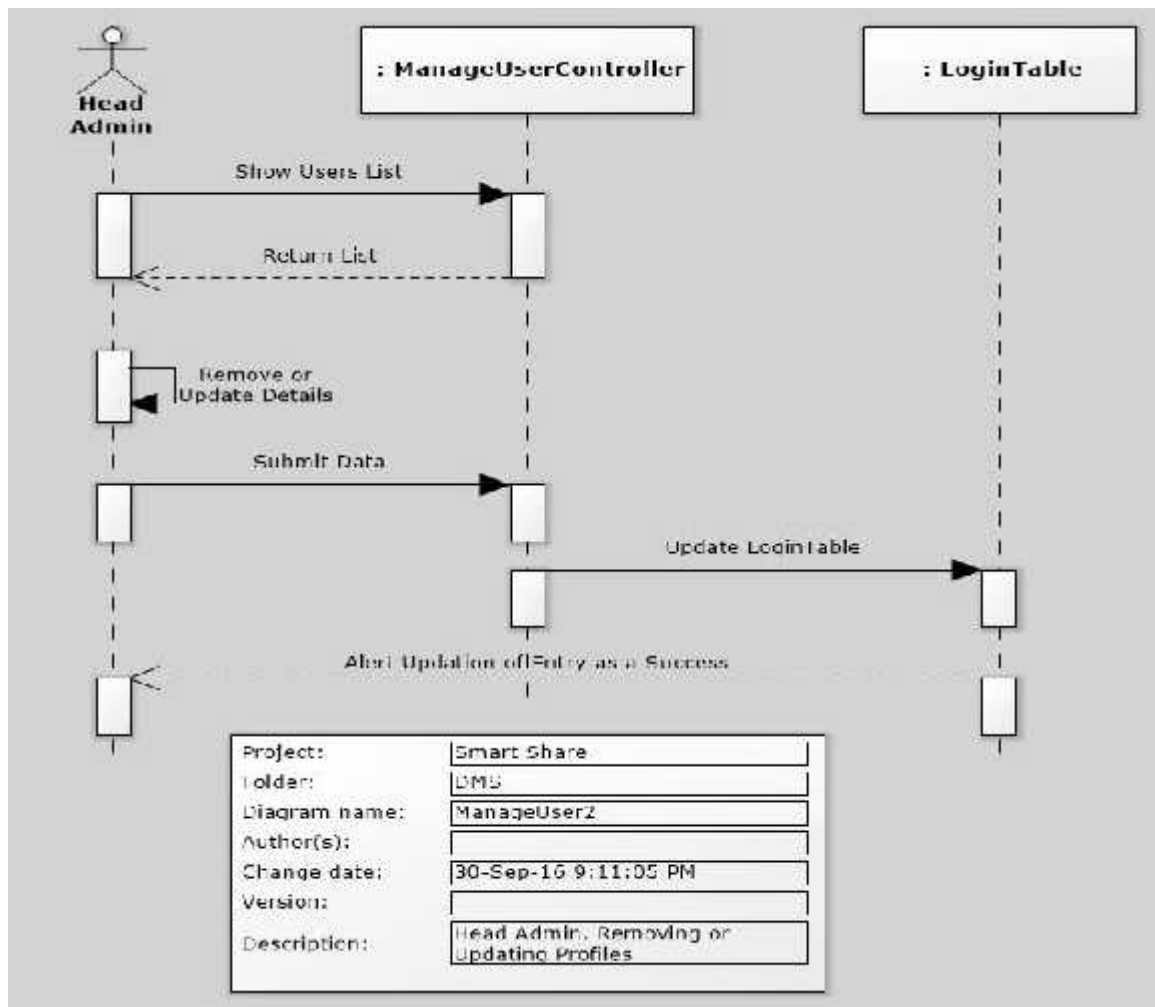


Figure 3.4 (b): Sequence Diagram for User Management

The following is the explanation of the Figure 3.4 (b).

- **Primary Actor:** Head Administrator.
- **Basic Flow:**
 - I. Admin fetches all users using the system.
 - II. A list of user is then displayed to the Admin.
 - III. Admin now monitors all active user activities and removes dormant users and updates user information.
 - IV. Admin is then alerted with a success message, when a user is removed.
- **Exceptional Flow:**
 - I. Admin fetches all users using the system.
 - II. A list of user is then displayed to the Admin.
 - III. Credentials that are being updated with new data into the database, on failure reports an error which are alerted.
 - IV. Admin is then alerted with a failure message.

3.2.3.3 Users Basic Activity

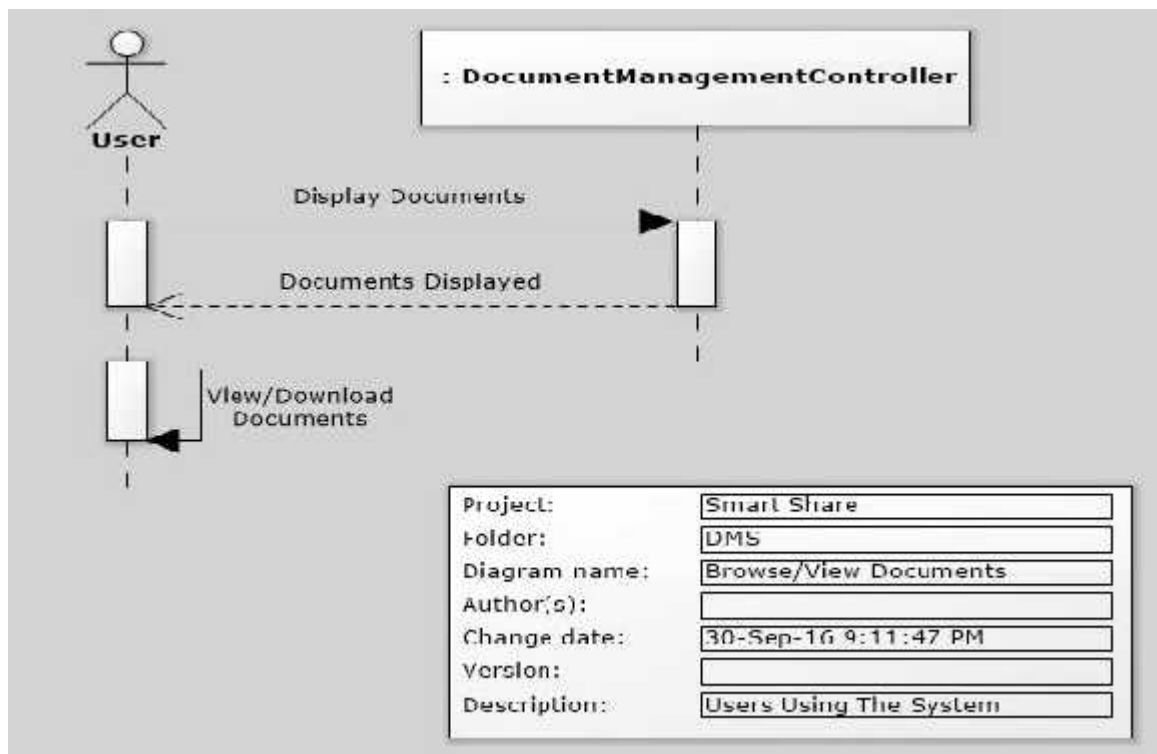


Figure 3.5: Sequence Diagram for Normal User Operation

The following is the explanation of the Figure 3.5.

- **Primary Actor:** A normal user.
- **Basic Flow:**
 - I. User logs into the system.
 - II. The user the selects the tab to view documents that are available in the system.
 - III. The user does a basic search for a document and he/she either download or view the document.
- **Exceptional Flow:**
 - I. User logs into the system.
 - II. The user the selects the tab to view documents that are available in the system.
 - III. The user then does a basic search, if the document is not found, an error is alerted.

3.2.3.4 Document Request Processing

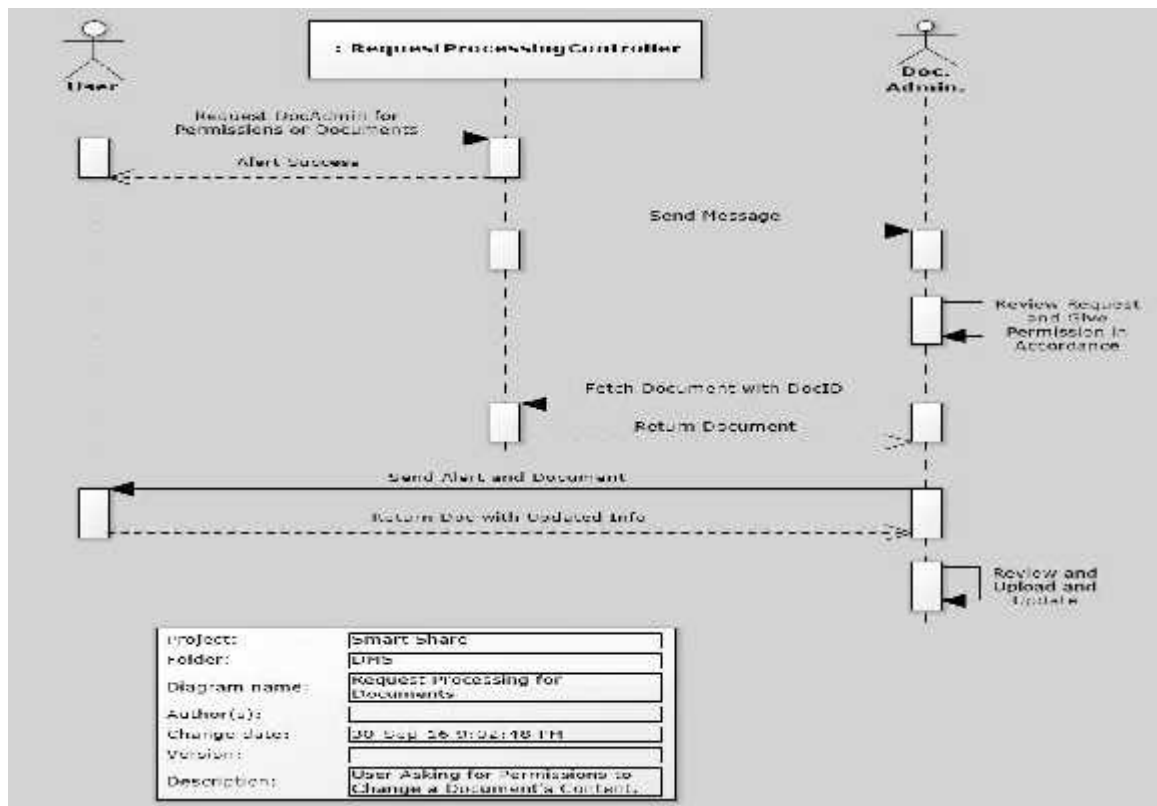


Figure 3.6: Sequence Diagram for Document Requests

The following the explanation of the Figure 3.6.

- **Basic Flow:**

- I. The user enter details in the document request form, and then submits it.
- II. Upon submission, the user is alerted with a success message.
- III. This request is then administered by the Document Administrator and all necessary actions are carried out to service the user's need.
- IV. If the document is existing one, required measures are taken to service the user's request, else if the document is a new one, it is uploaded.
- V. The user is then alerted.

- **Exceptional Flow:**

- I. The user enter details in the document request form, and then submits it.
- II. Upon submission, the user is alerted with a success message.
- III. This request is then administered by the Document Administrator and if the reason found is not sound, the request is rejected.
- IV. The user is then alerted, with a report stating the denial of the request.

3.2.3.5 Document Processing

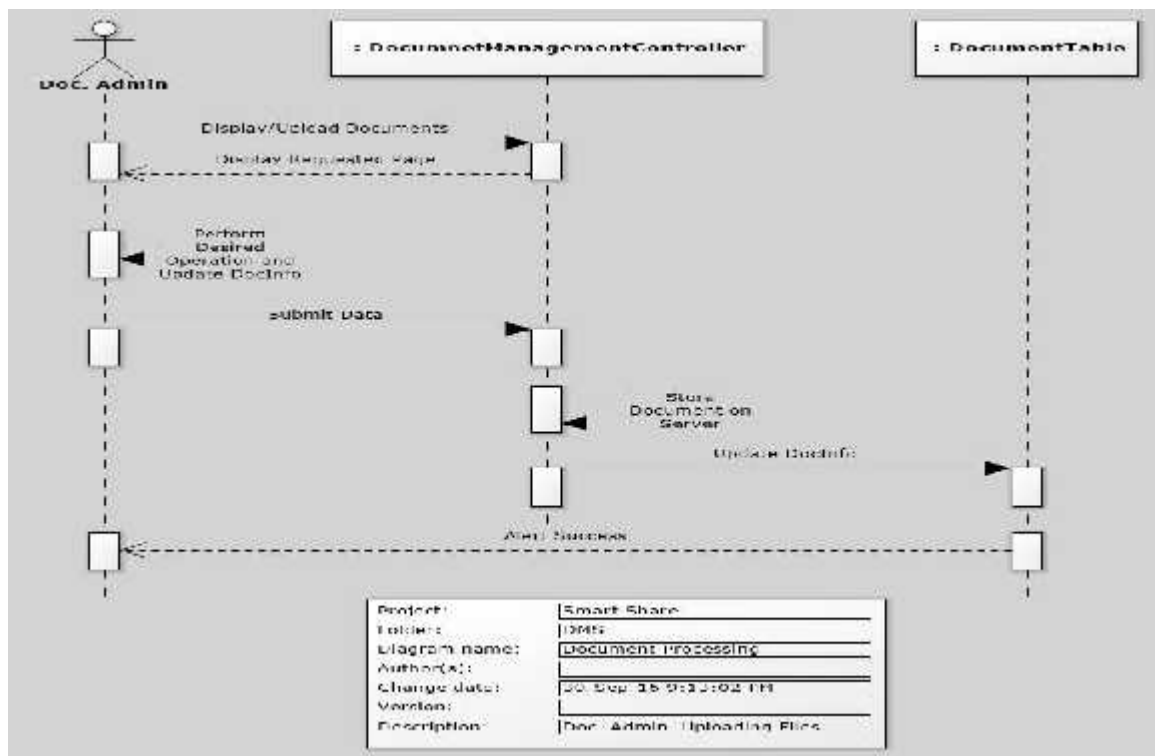


Figure 3.7: Sequence Diagram for Document Processing

The following is the explanation of the Figure 3.7.

- **Primary Actor:** Document Administrator.
- **Basic Flow:**
 - I. The Document Administrator selects a tab to view and upload documents.
 - II. If a new document is to be uploaded, a form is then displayed to the Doc. Admin, and all the required fields are entered.
 - III. Upon submission, the document is saved on the server, and the document tables in the database are updated.
 - IV. The Admin is then alerted upon success.
- **Exceptional Flow:**
 - I. The Document Administrator selects a tab to view and upload documents.
 - II. If a new document is to be uploaded, a form is then displayed to the Doc. Admin, and all the required fields are entered.
 - III. Upon submission, if the document type and size aren't supported by the system, a failure message is reported.
 - IV. The Admin is then alerted upon with failure message.

3.2.3.6 Logout Sequence

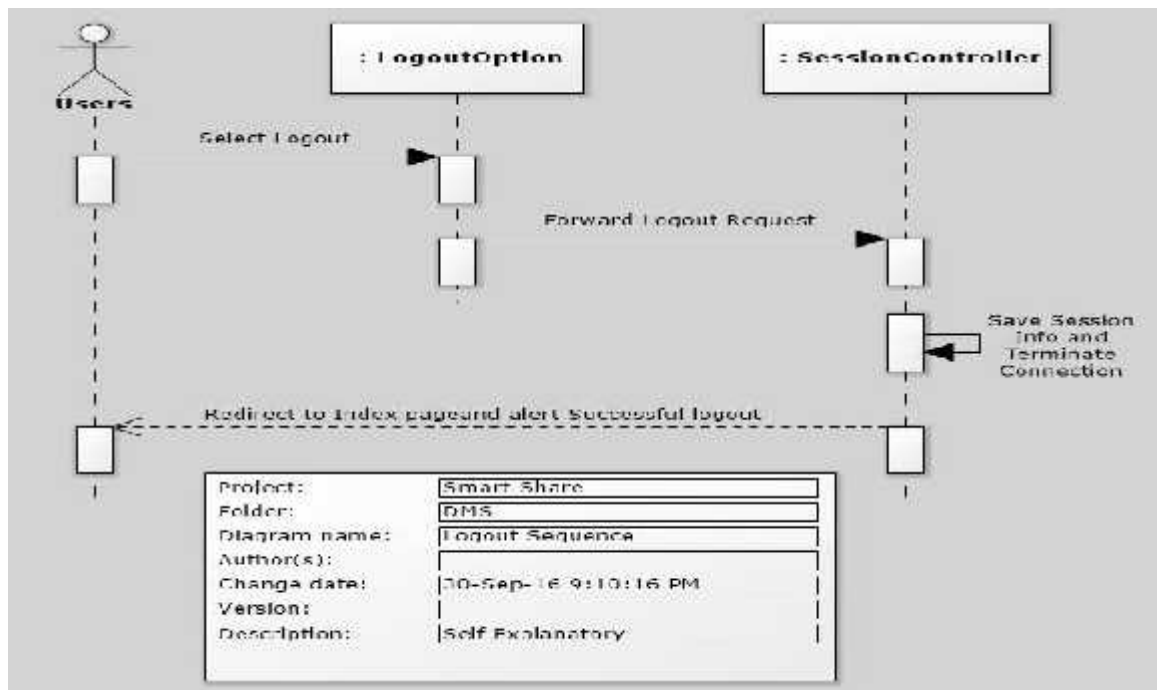


Figure 3.8: Sequence Diagram for Logout.

The following is the explanation of Figure 3.8.

- **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.

3.3 The Database Design

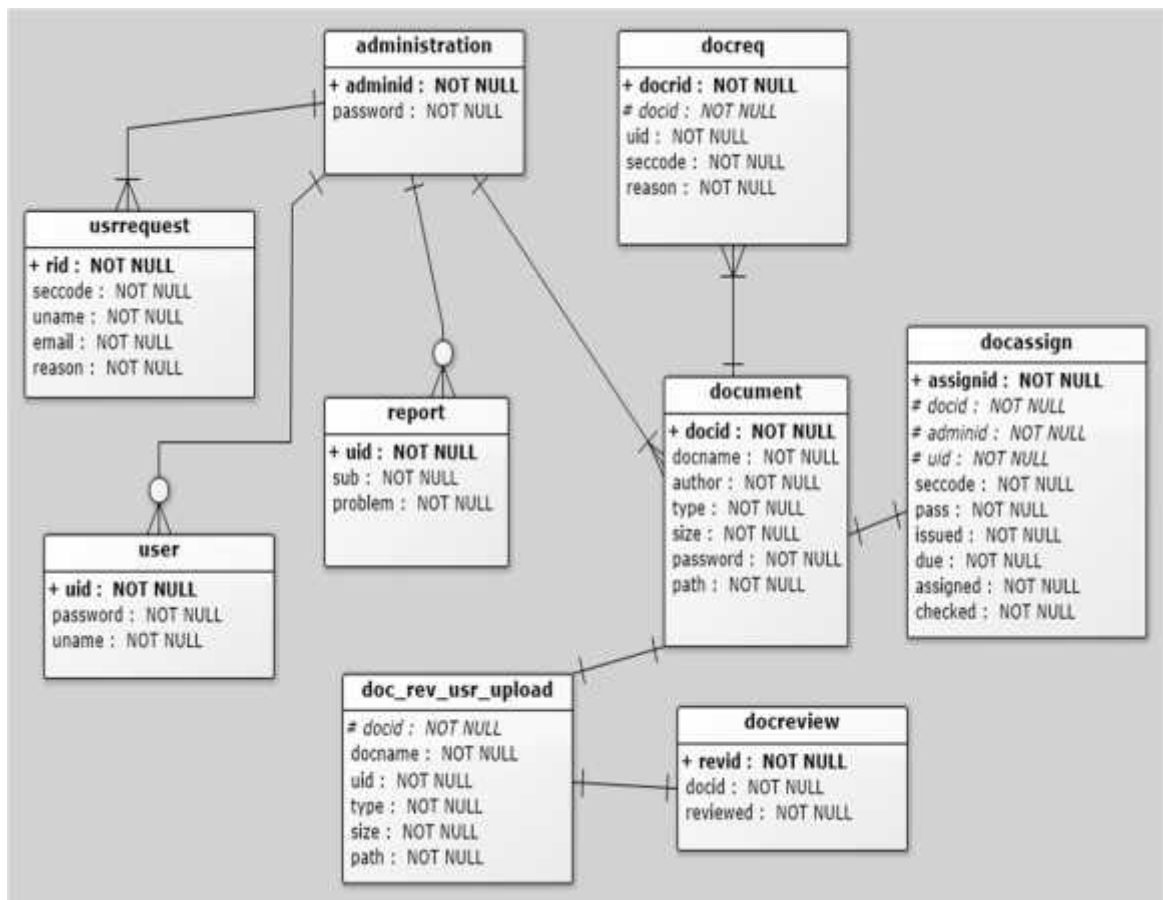


Figure 3.9: Entity Relationship Diagram of Smart Share

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 3.9.

- **Administration Table: administration**

- **Description:** This table is the main table, which is responsible for authenticating users who are assigned with roles defined as Admins.

Attributes	Type	Null	Description
AdminID (Primary)	int(1)	No	Unique id number
AdName	varchar(30)	Yes	Name of the Administrator
PSWD	varchar(30)	Yes	password

Table 3.1: Structure of Administration Table.

- **User Request Table: usrrequest**

- **Description:** This table is responsible for storing all the user request that are generated when a user wants to register and use the system.

Attributes	Type	Null	Description
rid (Primary)	int(9)	No	Unique Request Id
uname	varchar(50)	Yes	User Name
email	varchar(35)	Yes	e-mail address
reason	varchar(250)	Yes	Reason for using the application
seccode (Primary)	int(6)	No	Security code for validation
created	tinyint(1)	No	User is created or not

Table 3.2: Structure of User Request Table.

- **User Table: user**

- **Description:** This table stores all users' information.

Attributes	Type	Null	Description
rid	int(1)	Yes	User Request Id
uid (Primary)	int(5)	No	Unique User Id
uname	varchar(30)	No	User Name
pswd	varchar(50)	No	Password
seccode	int(6)	No	Security code
count	int(1)	Yes	Viwed credentials or not

Table 3.3: Structure of Users' Table.

- **Report Table: report**

- **Description:** This table stores all the users' suggestions and problems faced by them, it's like a feedback form.

Attributes	Type	Null	Description
uid	int(5)	Yes	User Id
sub	varchar(35)	Yes	Subject of the problem
prob	varchar(500)	Yes	Problem/Suggestions

Table 3.4: Structure of Report Table.

- **Document Table: document**

- **Description:** This table stores all the information related to the documents that are uploaded onto the server.

Attributes	Type	Null	Description
docid (Primary)	int(5)	No	Unique id referencing a document
docname	varchar(30)	Yes	Document's Name
author	varchar(250)	Yes	Authors' Name
version	decimal(3,1)	No	Version of the Document
doctype	varchar(10)	No	Type of the Document
docsize	int(11)	No	Size of the Document
url	varchar(250)	No	Path where the Document is Stored
verchd	tinyint(1)	Yes	Version of the Document is changed or not
pass	varchar(30)	No	Password that protects the Document

Table 3.5: Structure of Document Table.

- **Document Assign Table: docassign**

- **Description:** This table stores all the information related to documents that are assigned to particular user.

Attributes	Type	Null	Description
assignid (Primary)	int(5)	No	Document Assign ID
docid (Foreign)	int(5)	No	Document's unique ID
adminid (Foreign)	int(1)	No	Administrator's ID
uid (Foreign)	int(5)	No	User ID
seccode	int(11)	No	Security Code for validating user request
issued	varchar(30)	No	Date of Issue
due	varchar(30)	No	Due Date
assigned	tinyint(1)	No	Assigned - yes/no
checked	tinyint(1)	No	Document Checked or Not
pass	varchar(30)	No	Password set to the document

Table 3.6: Structure of Document Assign Table.

- **Document Review User Upload Table: doc_rev_usr_upload**

- **Description:** This table stores all the information related to user uploads of the documents that underwent changes

Attributes	Type	Null	Description
docid (Foreign)	int(5)	Yes	Document's unique ID
docname	varchar(50)	Yes	Document's Name
uid	int(5)	Yes	User ID
type	varchar(50)	Yes	Type of the Document
size	int(10)	Yes	Size of the Document
path	varchar(250)	Yes	Path where the Document is Stored

Table 3.7: Structure of Document Review User Upload Table.

- **Document Request Table: docreq**

- **Description:** This table stores all the information related to each document that a user requests for modification of documents, that need to undergo some changes.

Attributes	Type	Null	Description
docrid (Primary)	int(5)	No	Unique Document Request ID
docid (Foreign)	int(5)	No	Document's unique ID
uid (Foreign)	int(5)	No	User ID
seccode	int(11)	No	Security code that is used for validation
created	tinyint(1)	Yes	Request is Serviced or Not
reason	varchar(250)	No	User's Reason for changing document.

Table 3.8: Structure of Document Request Table.

- **Document Reviewed Table: docreview**

- **Description:** This table stores all the information related to documents being reviewed.

Attributes	Type	Null	Description
revid (Primary)	int(5)	No	Unique Review ID (Internal Purpose)
docid (Foreign)	int(5)	No	Document's unique ID
reviewed	tinyint(1)	Yes	Doc. Reviewed - Yes/No

Table 3.9: Structure of Document Reviewed 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 3.10 describes about the main interface, which displays the Index Page of Smart Share, which is common to all. It's the primary navigation point, where Users (Users/Admins) decide their destination.

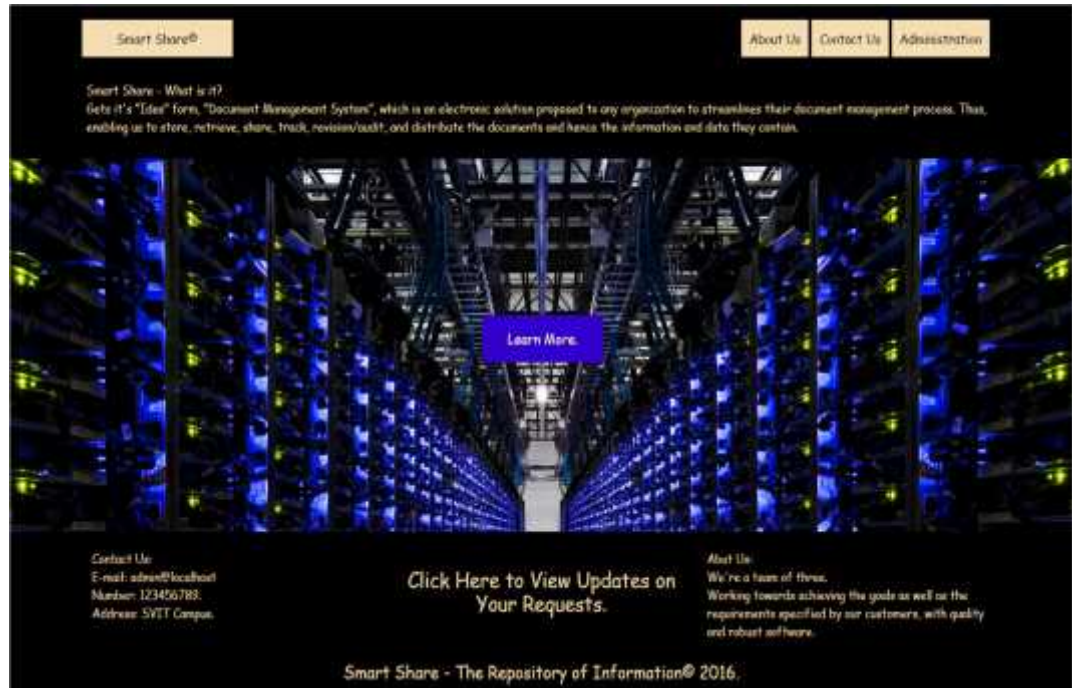


Figure 3.10: Index Page of Smart Share.

- **Administrator Login Interface:** Figure 3.11 describes about the interface that enables administrators to log into the system and service, maintain the system's state and redirects to respective pages based on roles.

Figure 3.11: Administrator Login

- **Head-Administrator's Interface:** Figure 3.12 describes the various operations that a head administrator can do, once she's/he's logged in.



Figure 3.12: Head Administrator's Operations.

1. **System Status Interface:** This interface as shown in Figure 3.12.1, displays the system's overall state, which helps in maintaining and monitoring the system's state.



Figure 3.12.1: System Status Interface.

2. **User Request Interface:** This interface as shown in Figure 3.12.2, displays a list of user requests; that is, user request for registration, which are either serviced or rejected based on the reason.



Figure 3.12.2: User Request Tab.

3. **Create Users Interface:** This interface as shown in Figure 3.12.3, allows the Head Administrator to create users/register them.

Welcome to Administration, Head Administrator: Mad Max

Home System Status User Requests **Create Users** Remove Users Logout

Register a User

Create a User

User ID:
Create a Unique ID

User Name:
Enter User's Name

Secret Code:
Enter Secret Code

Password:
Create a Password

Create User

Figure 3.12.3: Create User(s) Tab.

4. **Remove Users Interface:** This interface as shown in Figure 3.12.4, enables the Head Administrator to boot any user at any time without any prior notice to the user, or is done when a user no longer uses the system, so as to maintain the system's state.

Welcome to Administration, Head Administrator: Mad Max

Home System Status User Requests Create Users **Remove Users** Logout

Lets Delete Some Unwanted Users

User ID	User Name	Password	Secret Code	Delete User(s)
85	Kantik	1245	454	<input type="checkbox"/>
95	myname	abodehj	1549974	<input type="checkbox"/>

Delete Unwanted Users

Figure 3.12.4: Remove User(s) Tab.

- **Document-Administrator's Interface:** Figure 3.13 describes the various operations/interfaces that a Document Administrator wishes to perform, once he/she have logged into the system.



Figure 3.13: Document Administrator's Operations.

1. **Document Status Interface:** This interface enables a Doc. Admin., to check and monitor the system. In this interface, a list is displayed to the Doc. Admin., where a count is displayed, as in Figure 3.13.1.

The screenshot shows the Document Status Interface. It has the same navigation menu as Figure 3.13, but the "Document's Status" link is highlighted. The main content area displays a table with the following data:

Basis	Count
Document Count	1
Documents Assigned	1
Documents to be Reviewed	0
Documents Reviewed	0
Documents Uploaded After Review	0

Figure 3.13.1: Document Status Tab.

2. **View Documents Interface:** This interface enables a Doc. Admin., to view all the documents uploaded into the system. A list is displayed to the Doc. Admin., as shown in Figure 3.13.2 along with operations such as download or removing documents from the system.

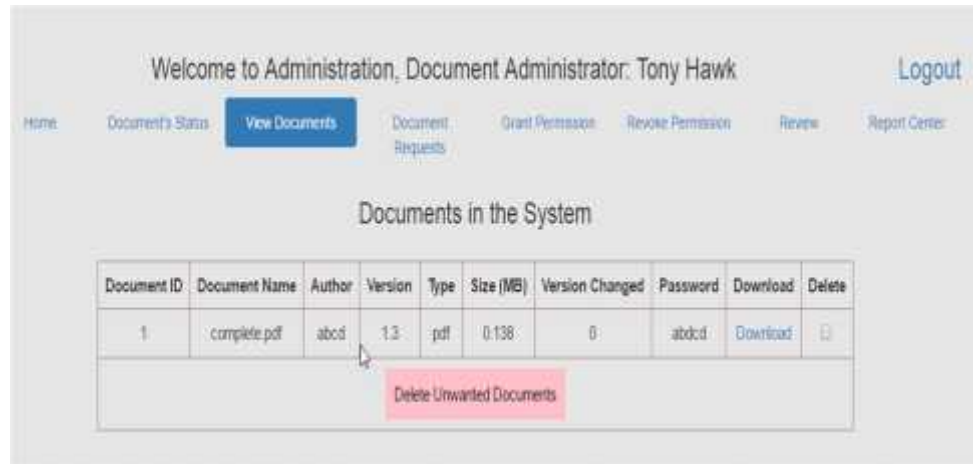


Figure 3.13.2: View Documents Tab.

3. **Document Request Interface:** This is the interface, which enables the Doc. Admin., to service the user request for document(s) based on the reason. It also enables the Doc. Admin. to upload files onto the server, which is available to all. Figure 3.13.3 shows this view/interface.

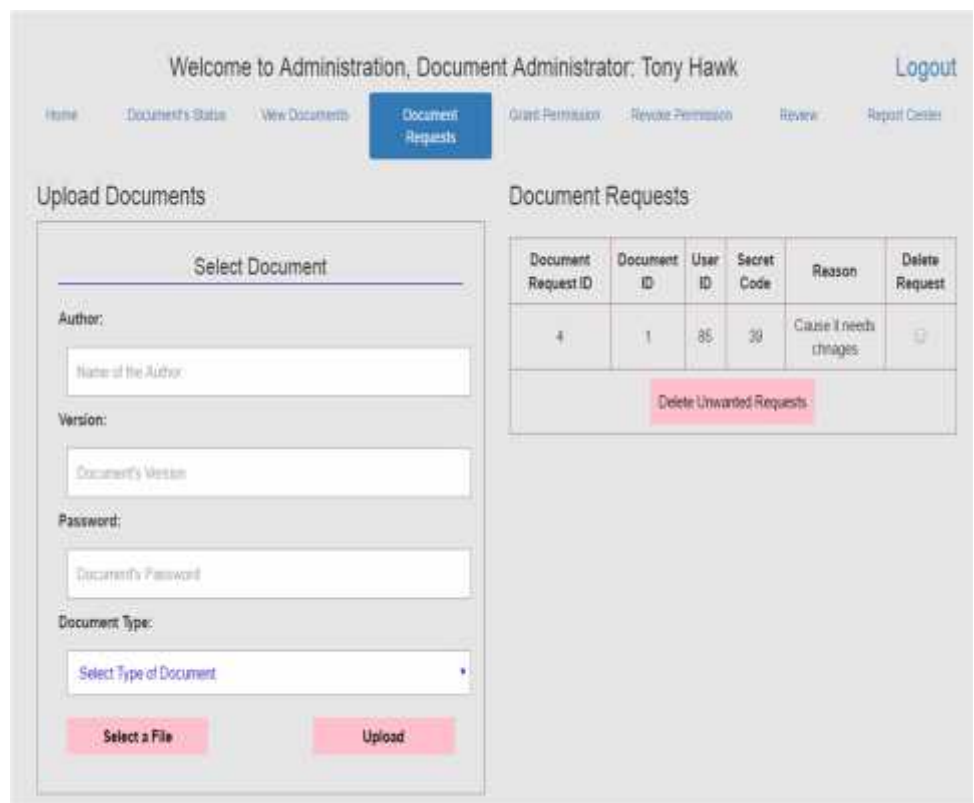


Figure 3.13.3: Document Upload/Request Tab.

4. **Grant Permission(s) Interface:** This is the interface, which enables the Doc. Admin., to assign documents to users. Figure 3.13.4 shows this view/interface.

Welcome to Administration, Document Administrator: Tony Hawk

Home Documents Status View Documents Document Requests **Grant Permission** Revoke Permission Review Report Center Logout

Issue Documents to Users

Grant Permissions

Document ID:

Administrator ID:

User ID:

Secret Code:

Issue Date: dd—yyyy Due Date: dd—yyyy

Assigned: Yes ☐ No ☐

Password:

Grant Access

Figure 3.13.4: Grant Permission(s) Tab.

5. **Revoke Permission(s) Interface:** This is the interface, which enables the Doc. Admin., to revoke any permissions granted to users, if the Document isn't submitted on time; that is, if a user fails to comply with the Terms and Conditions of the Document Agreement of Modification Policy. The “**Access Code**”, as shown in figure 3.13.5, is to authenticate the user's request.

Welcome to Administration, Document Administrator: Tony Hawk

Home Documents Status View Documents Document Requests Grant Permission **Revoke Permission** Review Report Center Logout

Revoke Permissions

Assign ID	Document ID	Administrator ID	User ID	Access Code	Issued On	Due On	Assigned	Checked	Password	Revoke
1	1	1	85	39	2016-11-01	2016-11-03	1	1	nothing	1

Revoke Permissions

Figure 3.13.5: Revoke Permission(s) Tab.

6. **Review Documents Interface:** This is the interface, which enables the Doc. Admin., to view the review status on the each document that were changed. It also provides an interface for uploading documents that were reviewed, along with the option for specifying the new version of the document. Figure 3.13.6, shows this interface.

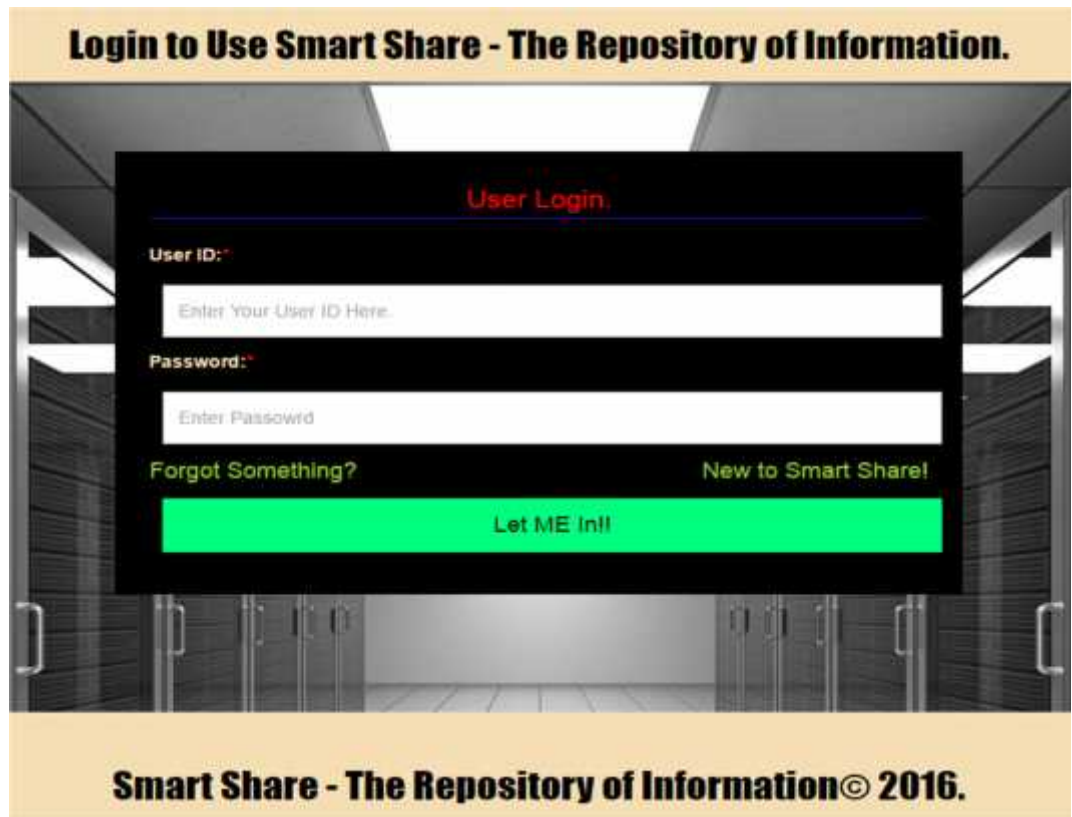
Figure 3.13.6: Upload Edited Docs. /Review Tab.

7. **Report Center Interface:** This is the interface, which enables the Doc. Admin., to view all the user uploaded documents for review, with a provision of either accepting or rejecting them, as shown in the Figure 3.13.7.

Document ID	Document Name	User ID	Type	Size (KB)	Download	Delete
20	2.1.pdf	85	pdf	3.207	Download	

Figure 3.13.7: Report Center Tab.

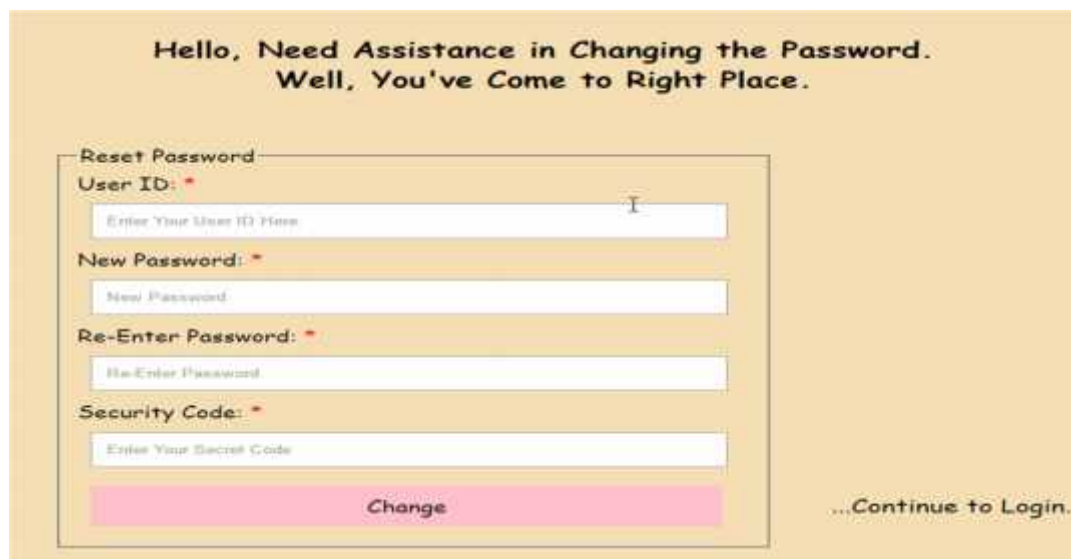
- **User Login Interface:** This interface enables users to gain access to the system. Figure 3.14 details this interface.



The image shows a login interface for 'Smart Share - The Repository of Information'. At the top, a yellow banner contains the text 'Login to Use Smart Share - The Repository of Information.' Below this is a black rectangular login box. Inside the box, the title 'User Login.' is written in red. There are two input fields: 'User ID: *' with a placeholder 'Enter Your User ID Here.' and 'Password: *' with a placeholder 'Enter Password'. Below these fields are two links: 'Forgot Something?' and 'New to Smart Share!'. A large green button labeled 'Let ME In!!' is at the bottom of the login box. The background of the page shows a server room with rows of server racks. At the bottom, a yellow banner contains the text 'Smart Share - The Repository of Information© 2016.'

Figure 3.14: User Login Interface.

- **Forgot Something? Interface:** This interface enables users to gain access to their accounts; that is, this interface is similar to “forgot password”, in most of the websites. Figure 3.15 details this interface.



The image shows a 'Forgot Something?' page for 'Smart Share - The Repository of Information'. The page has a yellow background. At the top, it says 'Hello, Need Assistance in Changing the Password. Well, You've Come to Right Place.' Below this is a white rectangular box with a black border. Inside the box, the title 'Reset Password:' is written. There are four input fields: 'User ID: *' with a placeholder 'Enter Your User ID Here.', 'New Password: *' with a placeholder 'New Password', 'Re-Enter Password: *' with a placeholder 'Re-Enter Password', and 'Security Code: *' with a placeholder 'Enter Your Secret Code'. Below these fields is a pink button labeled 'Change'. To the right of the pink button, there is a link that says '...Continue to Login.'

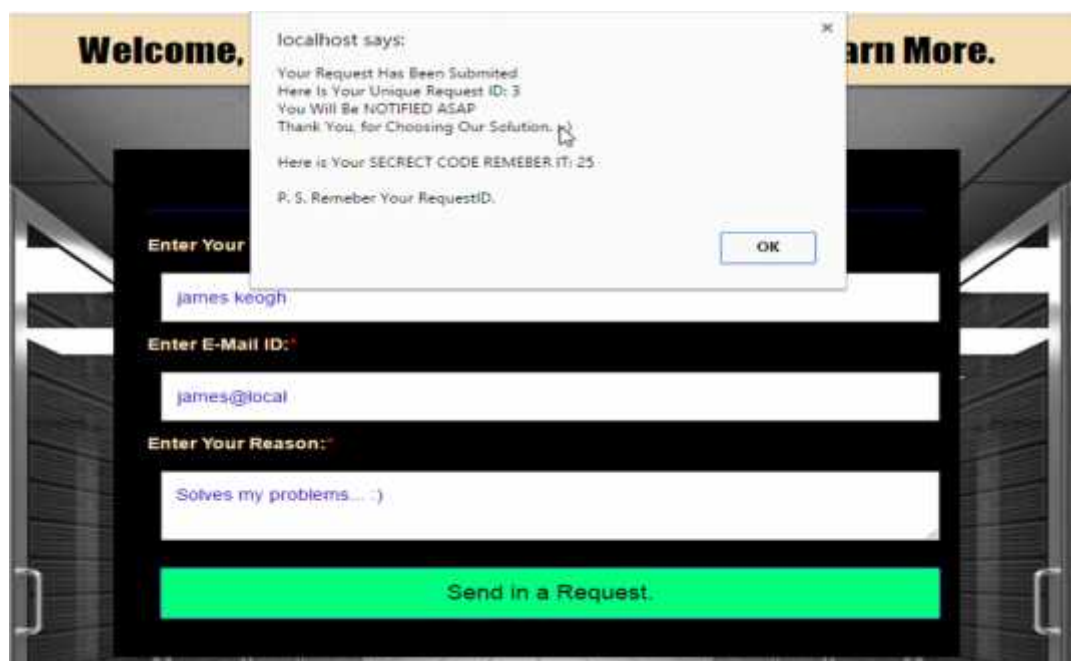
Figure 3.15: Forgot Something? Page.

- **New to Smart Share Interface:** This interface enables new users, to register and submit a request for registration. Figure 3.16 (a), details this interface and Figure 3.16 (b) details about the successful request that gets registered.



The image shows a web interface for user registration. At the top, a yellow banner reads "Welcome, You've Come to the Right Place to Learn More." Below this is a black rectangular form with a red title "Send in a Request." inside. The form contains three input fields: "Enter Your Name:" with a placeholder "Enter Your Name", "Enter E-Mail ID:" with a placeholder "Enter Your E-Mail ID", and "Enter Your Reason:" with a large empty text area. A bright green button labeled "Send in a Request." is at the bottom of the form. The footer of the page is a yellow banner that reads "Smart Share - The Repository of Information© 2016."

Figure 3.16 (a): New User Registration Interface.



The image shows the same registration form as in Figure 3.16 (a), but with a success message displayed. The form fields are now populated: "Enter Your Name:" contains "james keogh", "Enter E-Mail ID:" contains "james@local", and "Enter Your Reason:" contains "Solves my problems... :)". A modal dialog box titled "localhost says:" is overlaid on the form. The dialog contains the following text: "Your Request Has Been Submitted", "Here Is Your Unique Request ID: 3", "You Will Be NOTIFIED ASAP", "Thank You, for Choosing Our Solution.", "Here is Your SECRET CODE REMEBER IT: 25", and "P. S. Remeber Your RequestID:". An "OK" button is at the bottom right of the dialog. The background banner and footer are the same as in Figure 3.16 (a).

Figure 3.16 (b): Success Message after Request is posted.

- **Check Request Status Interface:** This interface enables new users to keep a track of their request, posted for registration. Figure 3.17 details this interface.



Figure 3.17: Check Status Interface.

- **The User's Interface:** Figure 3.18, details about the user's homepage, common to all users, and various tabs are defined to service the user's need.



Figure 3.18: User Home Page, Common to all users.

1. **View Documents Interface:** This interface details about all the available documents in the system. To narrow down overhead on users, users can customize the interface's view, by either narrowing down by searching or by removing columns that aren't essential. This interface is detailed using Figure 3.19.



Figure 3.19: User's View on Documents in the System.

2. **Request Document Interface:** This interface allows a user to raise a request for modifying the contents of a Document. Figure 3.20 details this interface.

Figure 3.20: User's Request Document Tab.

3. **Access Restricted Area Interface:** This interface enables a user, to gain access to the assigned documents along with uploading the documents that were edited. Figure 3.21, details this view/interface.

Figure 3.21: Access Restricted Area Tab.

4. **Suggestions/Report Interface:** This Interface enables users to share, view and submit the problems that were faced and suggestions that would make this application more confined to the Software Quality Attributes defined in chapter 2 under section 2.8. This view/interface is displayed to the user as shown in the Figure 3.22.

User ID	Subject	His/Her Problems/Reports
85	asdasd	sadasdasd

Figure 3.22: Suggestion/Report Tab.

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 Smart Share – The Repository of Information, 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	Smart Share shall have two types of access/login: User/Admin.
FR 02	Smart Share shall only be accessible to specified users and Admin with a valid username/password.
FR 03	The users shall be able to view and change the documents assigned to them.
FR 06	The Admin shall be able to upload new/revised documents.
FR 07	The Admin shall be able to view all the users registered in the system.
FR 08	The Admin shall be able to select users and to assign them a document.
FR 09	The Admin shall be to view the entire history for a document's revisions.
FR 10	The system allows the users to edit their personal profiles.
FR 11	The system allows the Admin to create new users.
FR 12	The system allows the Admin to remove users.

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 users can to view and change the documents assigned to them.
FR 06	TC 04	To test Admin can upload new/revised documents.
FR 07	TC 05	To test Admin can view all the users registered in the system.
FR 08	TC 06	To test Admin can select users and assign them a document.
FR 09	TC 07	To test Admin can view the entire history for a document's revisions.
FR 10	TC 08	To test system allows the users to edit their personal profiles.
FR 11	TC 09	To test system allows the Admin create new user/profile.
FR 12	TC 10	To test system allows the Admin remove a user/profile.

Table 4.2: Test Case Tables 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
 1. **Description:** To test the Login Interface for Admin and to test the Login Interface for User and their validation.
 2. **Input:** UserID/AdminID and Password.
 3. **Valid Range:** UserId/AdminID – Integer; Password – Alphanumeric.
 4. **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
 1. **Description:** To test users can to view and change the documents assigned to them.
 2. **Input:** User selects a document.
 3. **Output:** Browser or third party tools.
 4. **End Message/Result:**
 - If (selection == document and document == exist and access_code == user_access_code), the user will be able to download and make changes to document.
 - If (selection == document and document == exist and access_code != user_access_code), the user will not be able to download and make changes.
 - If (selection == document and document != exist and access_code == user_access_code), an error is to be alerted stating “no document”.
- **Test Case Number:** TC 04
 1. **Description:** To test Admin can upload new/revised documents.
 2. **Input:** AdminID, Password, Document.
 3. **Output:** Success Message Alerted.
 4. **End Message/Result:**
 - If (login type == “Doc. Admin” & uploadbutton.clicked = ‘true’ and document != existing and verchd == false) then display “Document saved Successfully”.
 - If (login type == “Doc. Admin” & uploadbutton.clicked = ‘true’ and document == existing and verchd == true) then display “Document saved Successfully with new Version”.
 - If (login type == “Doc. Admin” & uploadbutton.clicked = ‘true’ and document == existing and verchd == false) then display “Document already existing”.

- **Test Case Number: TC 05**
 1. **Description:** To test Admin can view all the users registered in the system.
 2. **Input:** AdminId; Password; User-List Selection.
 3. **Output:** User List.
 4. **End Message/Result:**
 - If (login type == “Head Admin” & UserManagement.clicked == ‘true’ and list.clicked == true and userlist.exists==true), then display users.
 - If (login type == “Admin” & UserManagement.clicked == ‘true’ and list.clicked == true and userlist.exists == false), then display message “No users Exist”.
- **Test Case Number: TC 06**
 1. **Description:** To test Admin can select users and assign them a document.
 2. **Input:** AdminID; Password; User-List and Permission.
 3. **Output:** Messsage/Alert.
 4. **End Message/Result:**
 - If (login type == “Doc. Admin” & Permission.clicked = ‘true’ and Document.clicked == true and userlist.exists == true and name.clicked == true), then assign user and display message “task/document assigned”.
 - If (login type == “Admin” & Permission.clicked == ‘false’ and Document.clicked == true and userlist.exists == true) then display message “Permission Denied”.
- **Test Case Number: TC 07**
 1. **Description:** To test Admin can view the entire history for a document’s revisions.
 2. **Input:** AdminId; Password; Document-List Selection.
 3. **Output:** System history.
 4. **End Message/Result:**
 - If (login type == “Doc. Admin” & DocList.clicked == true and Document.exists == true), then display document with Date and time stamps.
 - If (login type == “Doc. Admin” & DocList.clicked == true and Document.exists == false), then display message “No Documents exists”.
- **Test Case Number: TC 08**
 1. **Description:** To test system allows the users to edit their personal profiles.
 2. **Input:** AdminID/UserID, Password, View Profile.
 3. **Output:** Profile displayed.
 4. **End Message/Result:**
 - If (login type == “Admin”/”User” & myAccount.clicked == true and Profile.clicked == true and field.edited == true and

- submit.clicked == true) then display message “Profile saved successfully” and update the database.
- If (login type == “Admin”/”User” & profile.clicked == true and field.edited == empty and submit.clicked == true) then display “Mandatory fields should be filled” and do not update the system.
- **Test Case Number: TC 09**
 1. **Description:** To test system allows the Admin create new user/profile.
 2. **Input:** AdminID; New UserID; Password and selection create users.
 3. **Output:** New User Created.
 4. **End Message/Result:**
 - If (logintype == “Head Admin” and CreateUsers.clicked == true and Form == filled and submit.clicked == true and userid == unique), then update database with new user’s credentials and alert success.
 - If (logintype == “Head Admin” and CreateUsers.clicked == true and Form == filled and submit.clicked == true and userid == existing), then alert error, and don’t update database.
 - **Test Case Number: TC 10**
 1. **Description:** To test system allows the Admin remove a user/profile.
 2. **Input:** AdminID; UserID; Password and selection remove users.
 3. **Output:** User(s) Removed.
 4. **End Message/Result:**
 - If (logintype == “Head Admin” and RemoveUsers.clicked == true and UserID == set and submit.clicked == true and userid == existing), then update database by removing user’s credentials and alert success.
 - If (logintype == “Head Admin” and CreateUsers.clicked == true and UserID == set and submit.clicked == true and userid == not existing), then alert error, and don’t update database.

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
TC 07	Pass	Pass
TC 08	Pass	Pass
TC 09	Pass	Pass
TC 10	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. The Comparison that was done between the System built and another system that is available online in regards to the features available. It would also list the future work that is intended to be accomplished in the later versions of the Software.

5.1 Conclusion

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

- ✓ **Should allow users to view and save different versions of the same document:** This is achieved when the Admin gives the permission to a certain user to view the files. The selected files then appear to the user under the Document Tab in the user interface.
- ✓ **Save different versions of the same document:** The users can upload the files through an internal operation and send them to the Admin, the admin then can change the version of the file and upload them to the System.
- ✓ **Should allow the Admin to assign the users of tasks to make changes to the current version:** This is achieved when an Admin or a User realizes the need for a Document Update or need for creating a new Document. A request is raised by the user to Create/update a Document.
- ✓ **Approve/Reject the new/changed documents:** This is achieved when a user has created /updated a Document and uploads the file for review. The Admin, can then download and review the document. They would then reply to the user with their comments and let them know if he/she have accepted the document or not.
- ✓ **View detail summary for each document history:** This is achieved when a document is uploaded to the System by the Admin a date and time stamp is attached to the document automatically by the system and is in tabular form when the Admin tries to view the Documents in the System.
- ✓ **Upload Files in compressed format:** This is achieved by the inbuilt function that compresses the files uploaded to the system either by the Document upload.

5.2 Comparison

This section list the difference between the system developed and another system called OpenKM©. OpenKM© is an Open Document Management System which allows users to manage the creation, approval, distribution and archiving of all controlled documents. After briefly working with the Smart Share and working on a trial version of the OpenKM© system a comparison of the two systems were done on the basis of the

common functionalities offered by both the systems and the results were noted in the table below.

Table 5.1 shows the difference between the two System and lists the features that Smart Share and OpenKM© offers and features that lack and should be implemented in Smart Share.

Features	Description	Smart Share	OpenKM
Import Documents	Feature that allows users to import documents into the system.	Yes	Yes
Assign Documents	Feature used to assign documents to other individuals.	Yes	Yes
Electronic Signature	Feature that allows users to sign documents with electronic signatures.	No	Yes
Messaging	Feature that allows users to interact with each other within the system.	No	Yes
File Compression	Feature that compress uploaded files for memory management.	Yes	Yes

Table 5.1: Smart Share vs. OpenKM©.

After the comparison of the online available ETQ system with the DMS developed for Swami Vivekananda Institute of Technology certain features have been realized of importance and were proposed to be included in the next release of the system and will be discussed in detail in section 5.3.

5.3 Future Work

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

- ✓ **Dashboard:** After much discussion with the QA another need for the system has been realized. A Dashboard has been proposed that would be included in the next release of the system. It would allow the Admin to view all the documents in the System by Category and Department. It would also show the Admin then permissions that have been given to all the users of the system for each document along with the date and time stamp when permission was given.
- ✓ **Delete/Store Expired Documents:** Another requirement has been realized after the reviewing the current system, which would be to tag an expiration date with each document, so that when that expiration date arrive the document

automatically get removed from the Document lists and gets stored in the Expired Section. This would help keeping track of all the Expired Documents and making sure that the Admin does not assign that document to a user and the user should not be able to see the document he/she been assigned previously.

- ✓ **Automatic Message Generation for assigned task:** This has been proposed to save work redundancy and is expected to save extra effort for Admin. An Automatic Message generation by the system has been proposed for future release so that the system sends out an email to all the Users that have been given the permission to view the Document, hence saving time and effort for the Admin to type a message every time a permission to view the document is assigned to a user.
- ✓ **Reducing Overhead for Administrators:** This has been proposed to save work and overhead among administrators and increase the efficiency of administrators by in-corporation of new set of administrators especially for assigning and reviewing documents that are of high priority.
- ✓ **Setting up a Local Mail Server:** This has been proposed to serve the needs of various user, so as to eliminate communication gap and misunderstandings, which in turn will enable the automation of messages.

BIBLIOGRAPHY

- Anderson, C. (2012, December 7). Is document control really that important? *Bizmanuals OnPolicy*. Retrieved September 23, 2012, from <http://www.onpolicy.com/2010-12/is-document-control-really-that-important.html>.
- OpenKM @ <https://www.openkm.com>
- Noyes, M. J., Garland, J. K. (2008, September). Computer- vs. Paper-Based Tasks: Are They Equivalent? *www.princeton.edu*, 51, 1352–1375. Retrieved on August 25, 2012, from http://www.princeton.edu/~sswang/Noyesa_Garland_computer_vs_paper.pdf.
- MSDN (2001). File Compression: How to: Compress and Extract Files. Retrieved on October 7, 2012, from <http://msdn.microsoft.com/en-us/library/ms404280.aspx>.
- R. V. Binder. (1999, October). Testing Object-Oriented Systems Models, Patterns, and Tools. Addison Wesley, Reading, Massachusetts.
- Saru, D. (2012, February). Impact of UML Techniques in Test Case Generation. *International Journal of Engineering Science & Advanced Technology*, 2(2), 214 – 217. Retrieved on September 15, 2012, from http://ijesat.org/Volumes/2012_Vol_02_Iss_02/IJESAT_2012_02_02_12.pdf.
- Apache HTTPd Project @ <https://www.apache.org>
- MySQL Open Community Server Project @ <https://www.mysql.com>

Appendix: Sample Code

This section includes or provides snapshots of the code/programming parts that were used in making this project's interface.

Main Interface

```
<!DOCTYPE html>

<html>
  <head>
    <title>Smart Share</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/bootstrap.min.css" type="text/css"/>
    <link rel="stylesheet" href="css/main.css" type="text/css"/>
  </head>
  <body>
    <header class="container">
      <div class="row">
        <a href="#" class="col-sm-2 col-lg-2 text-left but">Smart Share&copy;</a>
        <span class="col-sm-4 col-lg-4"></span>
        <nav class="col-lg-6 col-sm-6 text-right">
          <a href="#1" class="but">About Us</a>
          <a href="#2" class="but">Contact Us</a>
          <a href="imp/adminlogin.php" class="but">Administration</a>
        </nav>
      </div>
    </header>
    <section class="container">
      <div class="row">
        <article class="two">
          Smart Share - What is it?<br />Gets it's "Idea" form,
          "Document Management System" , which is an electronic
          solution proposed to any organization to streamlines their
          document management process. Thus, enabling us to store,
          retrieve, share, track, revision/audit, and distribute the
          documents and hence the information and data they contain.
        </article>
      </div>
    </section>
    <section class="jumbotron">
      <a href="usrlogin.php" class="button">Learn More.</a>
    </section>
  </body>
</html>
```



```

<div class="row">
  <p class="col-sm-4 col-lg-4" id='1'>
    Contact Us:<br/>
    E-mail: admin@localhost<br/>
    Number: 123456789.<br/>
    Address: SVIT Campus.
  </p>
  <span class="col-sm-4 col-lg-4">
    <a href="usrupdates.php">
      <h3>
        Click Here to View Updates on Your Requests.
      </h3>
    </a>
  </span>
  <p class="col-lg-4 col-sm-4" id='2'>
    Abut Us:<br/>
    We're a team of three.<br/>
    Working towards achieving the goals as well as the<br/>
    requirements specified by our customers, with quality<br/>
    and robust software.
  </p>
</div>
</section>
<footer class="container">
  Smart Share - The Repository of Information&copy; 2016.
</footer>
</body>
</html>

```

User Interface

```
<!DOCTYPE html>
<html>
  <head>
    <title>User Login</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
    <link rel="stylesheet" type="text/css" href="css/login.css">
    <style>
      legend{text-align: center; border-bottom: solid #3300cc 1px;}
      #a1{
        float: left; text-decoration: none; color: yellowgreen;
        font-size: 18px;
      }
      #a2{
        float: right; text-decoration: none; color: yellowgreen;
        font-size: 18px;
      }
    </style>
  </head>
  <body>
    <header class='container'>
      Login to Use Smart Share - The Repository of Information.
    </header>
    <section class='jumbotron'>
      <div class="col-sm-3"></div>
      <div class='col-sm-6 left'>
        <form action="" method="POST">
          <fieldset>
            <legend>User Login.</legend>
            <label>User ID:<span>*</span></label>
            <input type='text' placeholder="Enter Your User ID Here."
              required maxlength="10" name='username'/>
            <label>Password:<span>*</span></label>
            <input type="password" placeholder="Enter Passowrd"
              required maxlength="30" name='password'/>
            <a href="imp/pswdchg.php" id="a1">Forgot Something?</a>
            <a href="regis.php" id="a2">New to Smart Share!</a>
            <button type='submit' class='but'>Let ME In!!</button>
          </fieldset>
        </form>
      </div>
      <div class="col-sm-3"></div>
    </section>
  </body>
</html>
```

```
<a href="index.php">  
    Smart Share - The Repository of Information&copy; 2016.  
</a>  
</footer>  
</body>  
</html>
```

Registration Interface

```
<!DOCTYPE html>
<html>
  <head
    <title>Login/Register</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
    <link rel="stylesheet" type="text/css" href="css/login.css">
    <style>

      legend {text-align: center; border-bottom: solid #3300cc 1px;}
    </style>
  </head>
  <body>
    <header class='container'>
      Welcome, You've Come to the Right Place to Learn More.
    </header>
    <section class="jumbotron">
      <div class='col-sm-3'></div>
      <div class='col-sm-6 right'>
        <form action="" method='POST'>
          <fieldset>
            <legend>Send in a Request.</legend>
            <label>Enter Your Name:<span>*</span></label>
            <input type='text' name='uname'
              placeholder="Enter Your Name" required/>
            <label>Enter E-Mail ID:<span>*</span></label>
            <input type='email' name='email'
              placeholder="Enter Your E-Mail ID." required/>
            <label>Enter Your Reason:<span>*</span></label>
            <textarea maxlength="150" name="reason"
              placeholder="Maximum Limit is 75 Characters">
            </textarea>
            <button type="submit" class='but'>Send in a Request.</button>
          </fieldset>
        </form>
      </div>
      <div class="col-sm-3"></div>
    </section>
    <footer class="container">
      <a href="index.php">
        Smart Share - The Repository of Information&copy; 2016.
      </a>
    </footer>
  </body>
</html>
```