**TEES: An Efficient Search Scheme over Encrypted Data on Mobile Cloud**

**ABSTRACT:**

Cloud Computing is a model of internet based computing where the resources like storage space, online software are provided by different cloud service providers to different types of cloud users who needs cloud services. When a cloud user outsources the data on cloud, it has to provide more security for outsourced data preventing data manipulated or accessed by unauthorized users. So to maintain data integrity, each and every cloud service has to be stored securely. For easier accessing of files and to generate file indexes, each file is stored in cloud server. Now cloud users search files and again send download request to cloud server. This process is time consuming and also there is a chance that the cloud service provider might access those files which stored in cloud server, because both the encrypted file and correspondent key and file indexes are stored in cloud server. To overcome these problems, this system introduces storage nodes for storing file indexes and encrypted files and cloud server stores files keys. When a cloud user uploads file, the file index is generated automatically and file is encrypted by using AES algorithm with automatically generated key. After that by Visual cryptography scheme, the key is converted into image and then generated as key image and source images respectively. The encrypted file and the file indexes are stored in storage node, key and source image are stored in cloud server and key image is passed to file owner. Whenever file owner or file users want to download or access files then perform search and then put key image as an input. If valid, it matches the key with the source image and later it can be downloaded.

**INTRODUCTION:**

Cloud storage system is a service model in which data are maintained, managed and backup remotely on the cloud side, and meanwhile data keeps available to the users over a network. Mobile Cloud Storage (MCS) denotes a family of increasingly popular on-line services, and even acts as the primary file storage for the mobile devices. MCS enables the mobile device users to store and retrieve files or data on the cloud through wireless communication, which improves the data availability and facilitates the file sharing process without draining the local mobile device resources. The data privacy issue is paramount in cloud storage system, so the sensitive data is encrypted by the owner before outsourcing onto the cloud, and data users retrieve the interested data by encrypted search scheme. In MCS, the modern mobile devices are confronted with many of the same security threats as PCs, and various traditional data encryption methods are imported in MCS. However, mobile cloud storage system incurs new challenges over the traditional encrypted search schemes, in consideration of the limited computing and battery capacities of mobile device, as well as data sharing and accessing approaches through wireless communication. Therefore, a suitable and efficient encrypted search scheme is necessary for MCS.

# Objective:

Cloud computing can be referred to as the storing and accessing of data over the internet rather than your computer’s hard drive. This means that the data from either the computer’s hard drive or over a dedicated computer network. Cloud computing is stored at a remote place and is synchronized with other web information.

**SCOPE OF THE PROJECT:**

Cloud computing can be referred to as the storing and accessing of data over the internet rather than your computer’s hard drive. This means that the data from either the computer’s hard drive or over a dedicated computer network. Cloud computing is stored at a remote place and is synchronized with other web information.

**System Analysis**

**Existing System:**

In this existing system file owner stores the file into the cloud server. So here, lots of file owners access permission in the same cloud server and at that time other file owner will access the other files. Owner can miss use the other owner’s file. And the keys generated here can be easily hacked.

**Disadvantages:**

1. Server Information Acquisition
2. Keywords-files Association Leak.
3. Statistics Information Leak.

**Proposed System:**

The Main aim of this Project is to secure the user files in cloud storage.

Firstly, user uploads the files with their respective Login id. The main purpose of Cloud provider is to upload the files with secured image and generating OPE (Order Preserving Encryption) password .The purpose of secured image is, unauthorized user can’t access the file in cloud. Here files are encrypted into two parts such as encrypted Index and encrypted files by using FAH (Fast Accumulated Hash) Algorithm. Now after splitting files, it automatically generates a secured image called as OPE password which is

Not known to the third party .The secured is spitted into two images like

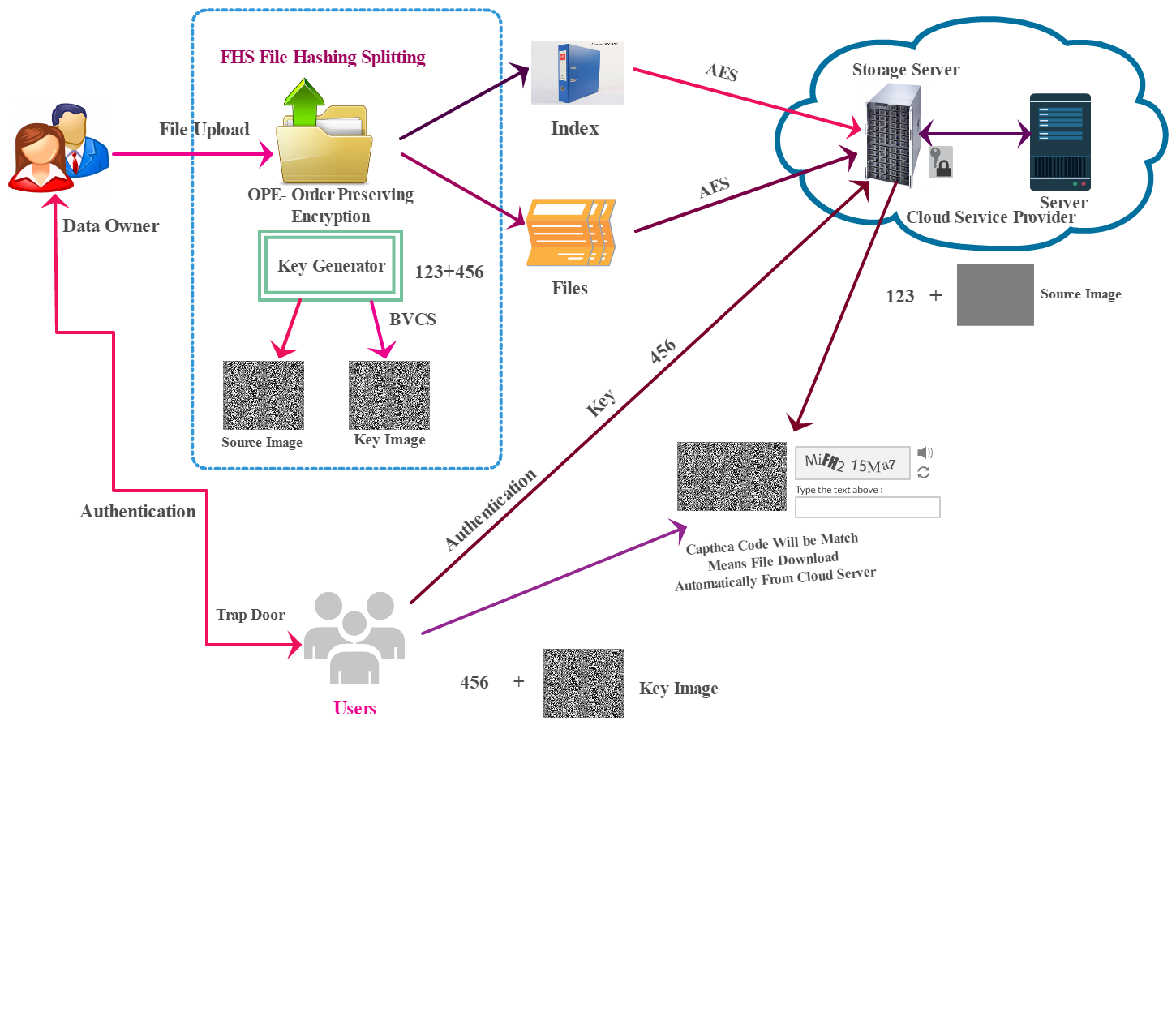
Source and key image by using BVCS (Binocular Visual Cryptography schemes) algorithm.

The encrypted file, Source image and OPE have been stored in cloud with respective file. If the user needs to view or select the particular file, the request must first be sent to the cloud service Provider. The provider verifies the user id and file request, later it will send OPE password and key image to user. Now the user has to send the key image to the cloud for accessing the files. The cloud matches the key image with the source image it already has. When both matches, it will send the file in the form of a Captcha and it can be downloaded. Hackers cannot hack the source image or key image and Captcha will be produced only when it is a valid user.

**Advantages:**

* Reducing File Search and Retrieval Time
* Reducing the Energy Consumption
* Reducing Traffic Overhead
* control the statistics information leak
* Top-k relevant files searching algorithm.

**ARCHITECTURE DIAGRAM:**



# SYSTEM CONFIGURATION:

# Processor - Intel Pentium

* Speed - 1.1 GHz
* RAM - 4 Gb
* Hard Disk - 260 GB

# S/W SYSTEM CONFIGURATION:

* Operating System - Windows 7/8/10
* Front End - J2EE
* Server side Script - Java Server Pages.

MODULES

* Secure File Uploading
* Splitting Encrypted File
* Splitting Encrypted File Image Split-up Using BVCS
* Verification

**MODULES DESCRIPTION**

**SECURE FILE UPLOADED**

In this module, to mitigate the security leakages it is implemented with security enhancement in consideration of the modified encrypted search procedure in order to mitigate statistics information leak and keywords-files association leak. The file is uploaded with secured images and password which is generated using File homomorphic Encryption algorithm. The main goal of these modules is to prevent the unauthorized user gaining the access of this file.

**SPLITTING ENCRYPTED FILE**

The primary purpose of encryption is to protect the confidentiality of digital data stored on computer system or transmitted via the internet or other computer system. Modern encryption algorithm plays a vital role in the security assurance of IT system and communication as they can provide not only confidentiality but also the integrity and non-de-duplication. Encryption is the most effective way to achieve data security. The Cloud provider uploaded the User files that will be encrypted into two parts like encrypted Index and encrypted files by using AES (Advanced Encryption Standard) Algorithm before sending them to the cloud. The encrypted file have to been stored in storage node with their respective file Id.

**IMAGE SPLITUP USING BVCS**

Image splitting is a technique most often used to slice a larger image into smaller parts to make it load faster. Cloud provider upload the user file with secured image, that image should be splitting into two images like source and key image by using BVCS (Binocular Visual Cryptography schemes algorithm. Then, the key image and the password will be send to the particular user and the necessary file can then be downloaded. The password is generated which is then splitted into source image and key image and they are stored to the user and cloud server.

**VERIFICATION**

Verification is the act of reviewing, inspecting or testing a technical standards. Now the user has to send the key image to the cloud for accessing the files. The cloud matches the key image with the source image it already has. When both matches, it will send the file in the form of a captcha. Then it can be downloaded easily. It is the act of reviewing, inspecting or testing in order to establish and document that a product, service or system meets regulatory or technical standards.

**Literature Survey:**

**Paper 1:**

**Title:**

Verifiable Symmetric Searchable Encryption For Semi-honest-but-curious Cloud Servers

**Authors:**

Q. Chai and G. Gong

**Abstract:**

Outsourcing data to cloud servers, while increasing service availability and reducing users’ burden of managing data, inevitably brings in new concerns such as data privacy, since the server may be honest-but-curious. In this paper, we investigate the searchable encryption problem in the presence of a semi-honest-but-curious server, which may execute only a fraction of search operations honestly and return a fraction of search outcome honestly. To fight against this strongest adversary ever, a verifiable SSE (VSSE) scheme is proposed to offer verifiable search ability in additional to the data privacy, both of which are further confirmed by our rigorous security analysis. Besides, we treat the practicality/efficiency as a central requirement of a searchable encryption scheme.

**Paper 2:**

**Title:**

Searchable Encryption with Conjunctive Field Free Keyword Search Scheme

**Authors:**

Fairouz Sher ALI, Songfeng LU

**Abstract:**

Prior Searchable Symmetric Encryption (SSE) works focus on single keyword search. Conjunctive Keyword Searches (CKS) schemes enhance system usability by retrieving the matching files. Most of existing conjunctive keyword works that use conjunctive keyword searches with fixed position keyword fields are not useful for many applications, such as the body of e-mail and unstructured text. In our paper, we propose a new symmetric key encryption scheme which supports a keyword field free method for conjunctive keyword search on encrypted file without needing to specify the positions of the keywords where the keywords can be in any arbitrary order. Furthermore, we introduced an efficient secure index construction based on pseudorandom functions and Bloom filters. Through analysis section, we determine how such scheme could be used to guarantee fast, low storage overhead and secure access to the database.

**Paper 3:**

**Title:**

Privacy Preserving Multiple Keyword Search for Confidential Investigation of Remote Forensics

**Authors:**

S. Hou, T. Uehara, S. Yiu, L. C. Hui, and K. Chow.

**Abstract:**

Remote forensics can help investigators perform investigation without need to ship hard drives or travel to a remote location. With increased use of cloud computing technologies, it is becoming more and more difficult to perform post-event forensic investigation. The other alternative is to let the server administrator search the relevant information and retrieve the data for the investigators provided a warrant can be provided. However, sometimes, the investigators need to keep the investigation subject confidential due to the confidentiality of the crime or the server administrator may be one of the suspects. In this paper, we address how to solve this problem by multiple keyword search over encrypted data, so that the investigators can obtain the necessary evidence while keeping the investigation subject confidential and at the same time, the irrelevant data can be protected from exposing to the investigators.

**Paper 4:**

**Title:**

Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data

**Authors:**

N. Cao, C. Wang, M. Li, K. Ren, and W. Lou

**Abstract:**

With the advent of cloud computing, data owners are motivated to outsource their complex data management systems from local sites to the commercial public cloud for great flexibility and economic savings. Thus, enabling an encrypted cloud data search service is of paramount importance. Considering the large number of data users and documents in the cloud, it is necessary to allow multiple keywords in the search request and return documents in the order of their relevance to these keywords. Related works on searchable encryption focus on single keyword search or Boolean keyword search, and rarely sort the search results. In this paper, for the first time, we define and solve the challenging problem of privacy-preserving multi-keyword ranked search over encrypted data in cloud computing (MRSE). To improve search experience of the data search service, we further extend these two schemes to support more search semantics. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given. Experiments on the real-world data set further show proposed schemes indeed introduce low overhead on computation and communication.

**Paper 5:**

**Title:**

Privacy-Preserving Multi-Keyword Fuzzy Search over Encrypted Data in the Cloud

**Authors:**

B. Wang, S. Yu, W. Lou, and Y. T. Hou

**Abstract:**

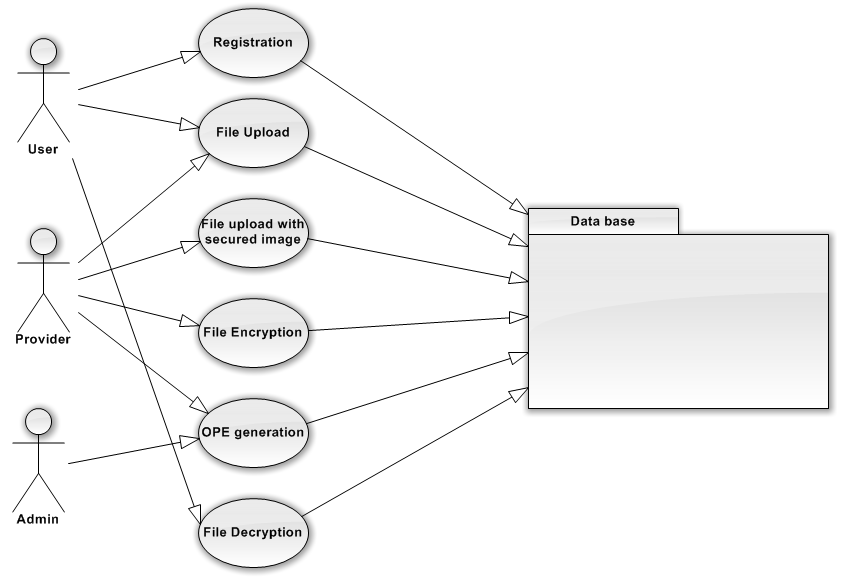
Enabling keyword search directly over encrypted data is a desirable technique for effective utilization of encrypted data outsourced to the cloud. Existing solutions provide multi keyword exact search that does not tolerate keyword spelling error, or single keyword fuzzy search that tolerates typos to certain extent. In this paper, we propose a novel multi keyword fuzzy search scheme by exploiting the locality-sensitive hashing technique. Our proposed scheme achieves fuzzy matching through algorithmic design rather than expanding the index file. It also eliminates the need of a predefined dictionary and effectively supports multiple keyword fuzzy searches without increasing the index or search complexity.

**SYSTEM DESIGN:**

**UML DIAGRAMS**

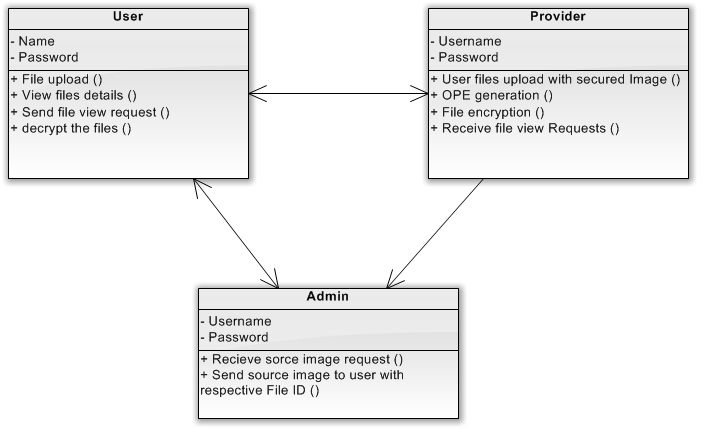
**USE CASE DIAGRAM**

Roles of the actors in the system can be depicted. In our use case diagram first user login into user window then if it is a valid user means then it can communicate with the cloud server.

****

**CLASS DIAGRAM**

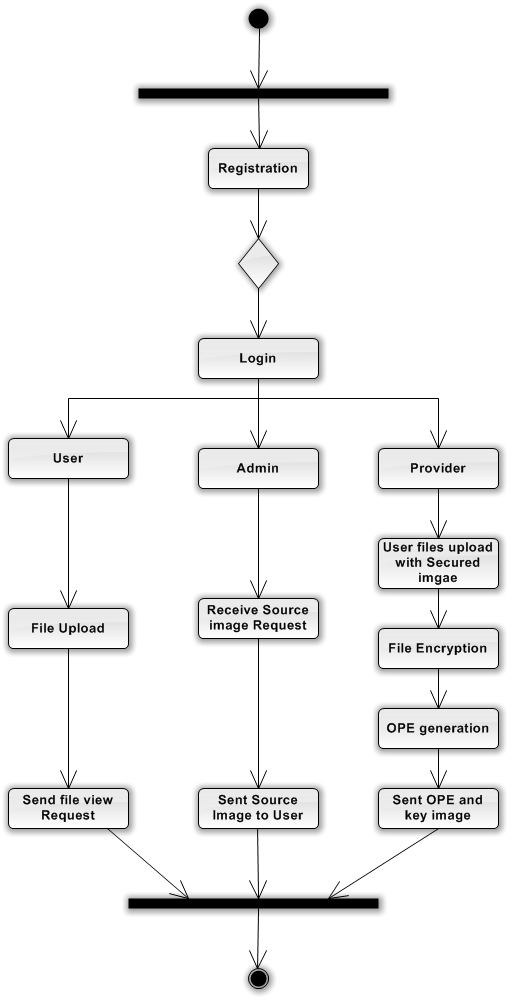
In our class diagram we having the details about user, first user login into user window then if it is a valid user, then it can communicate with the cloud server. Here ranking function is involved in order to search the file in the order of ranking basis. The storage node contains the encrypted files and the user and provider has some of the particular registrations such as username and password.

****

**ACTIVITY DIAGRAM**

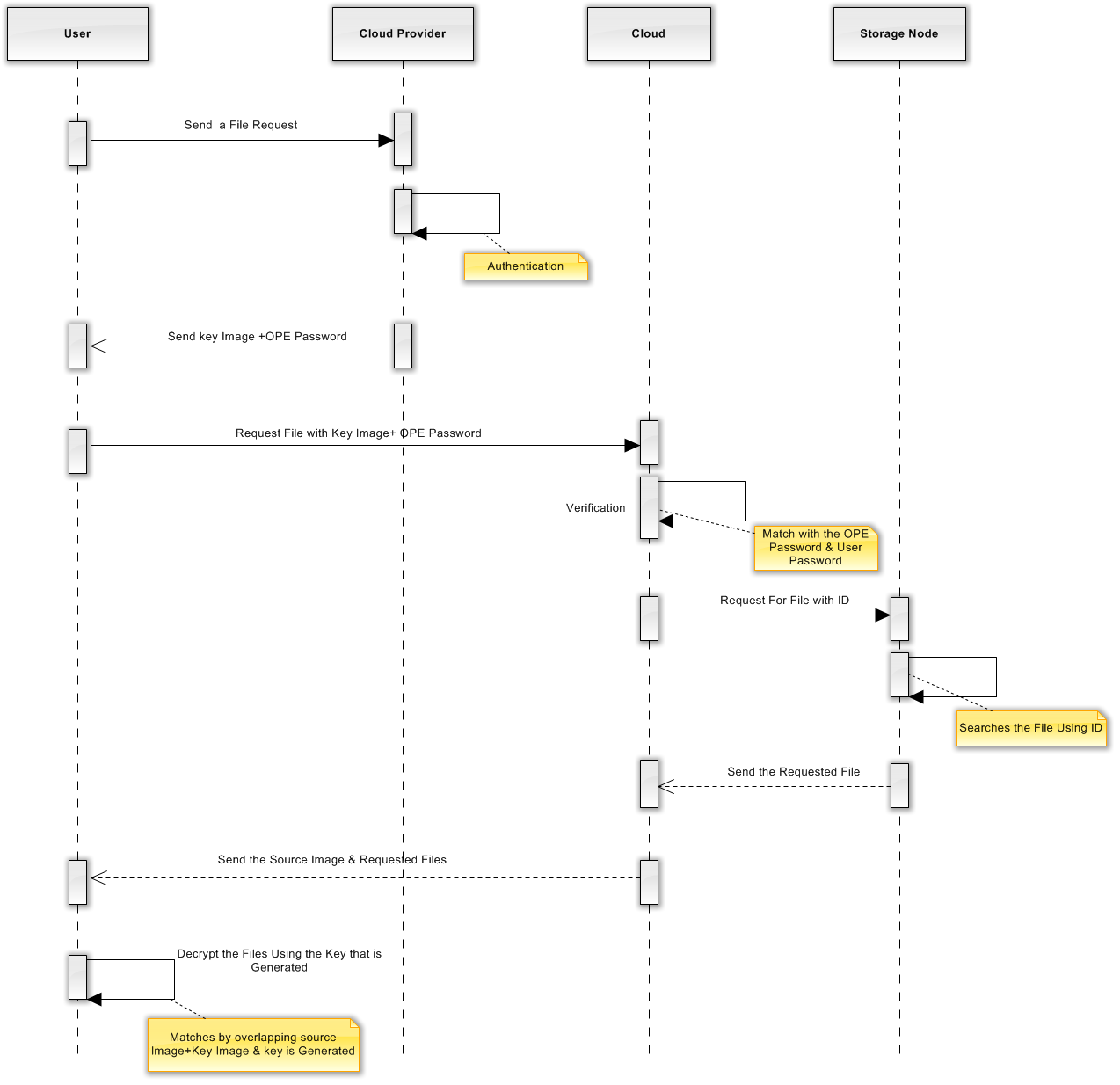
The cloud storage contains the encrypted file and files can be retrieved from the user. The cloud server contains the respective keys and later entering the correct key the files will be downloaded.

After valid registration, user uploads the file and sends the file request. Similarly, provider performs the encryption and generates OPE password and admin verifies.

****

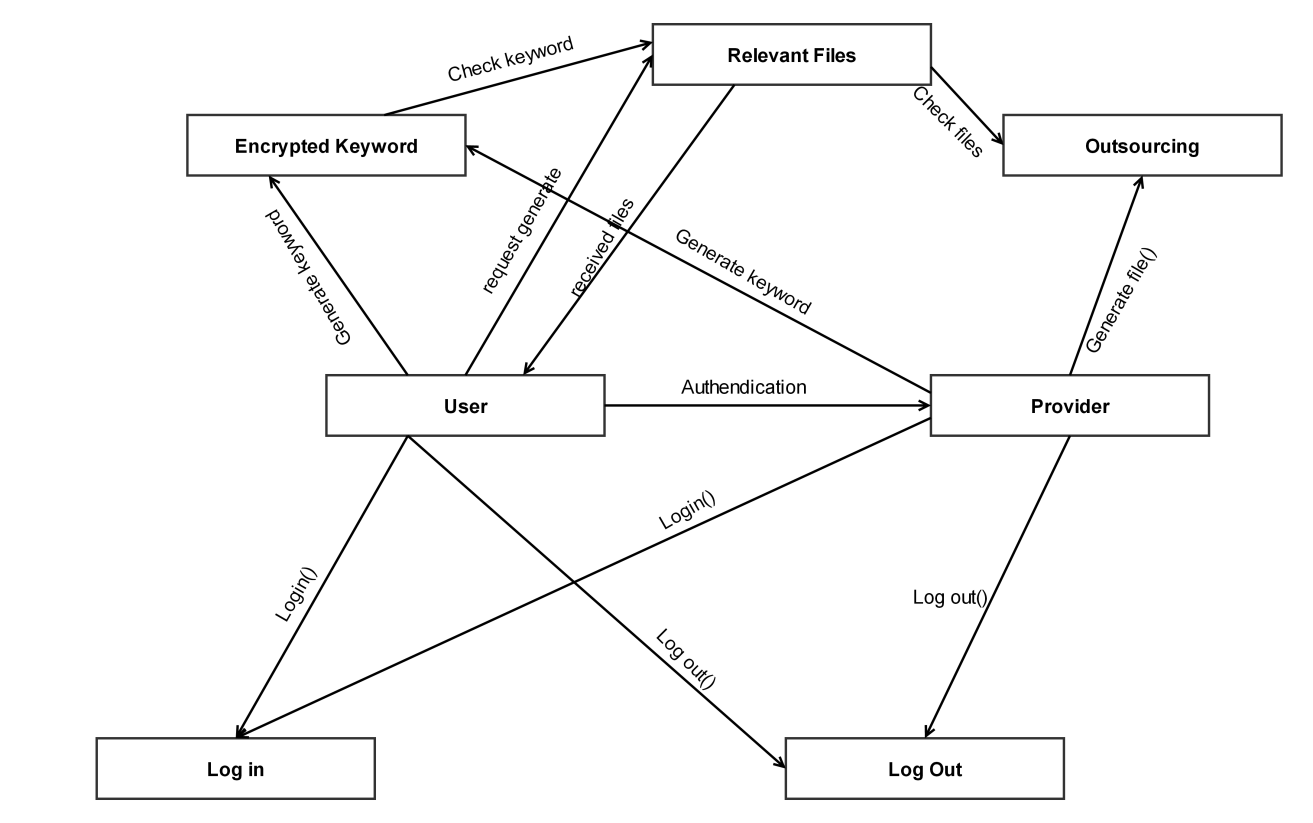
**SEQUENCE DIAGRAM**

In the sequence diagram, user enters into the cloud by performing certain authentication and user will retrieve the files available in the server. It explains about sending a file request to the provider and requesting for a OPE password and after verification the cloud server will provide the required source image to the user.



**COLLABORATION DIAGRAM**

A collaboration diagram describes interactions among objects in terms of sequenced messages it, explains about sending a file request to the provider and requesting for a OPE password and after verification the cloud server will provide the required source image to the user.

****

**DATAFLOW DIAGRAM**

User login into user window then if it is a valid user means then it can communicate with the cloud server. The registered users can publish and subscribe.

.

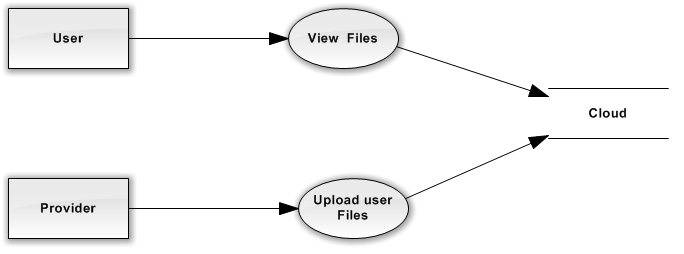
**Level 0:**



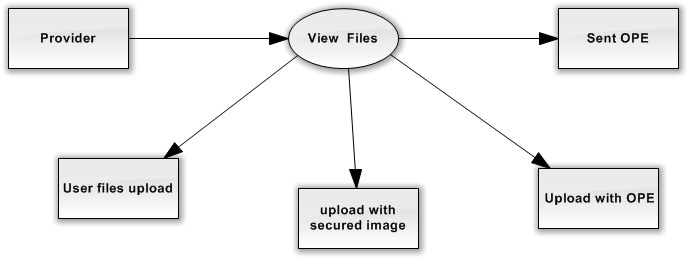
**Level 1:**



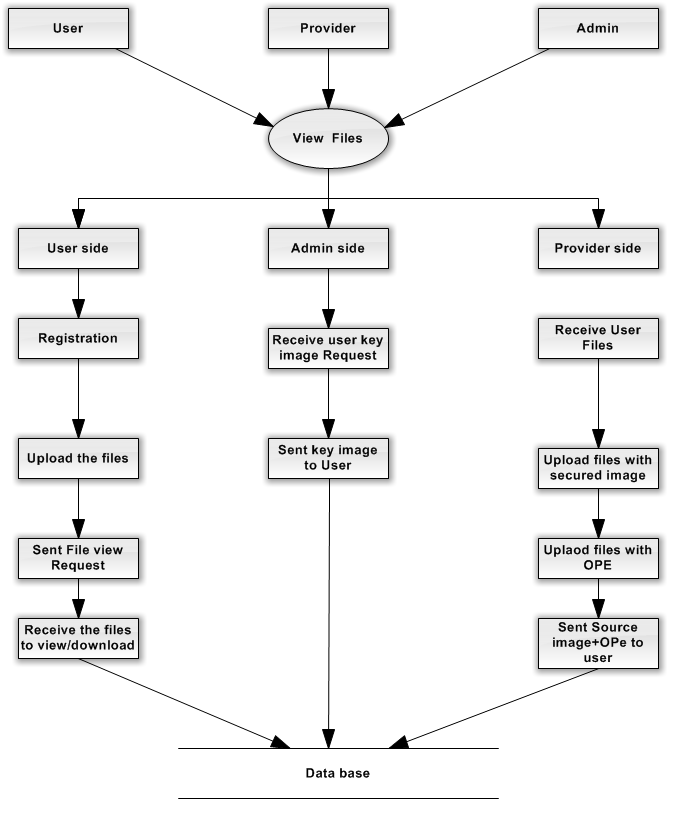
**Level 2:**



**Level 3:**

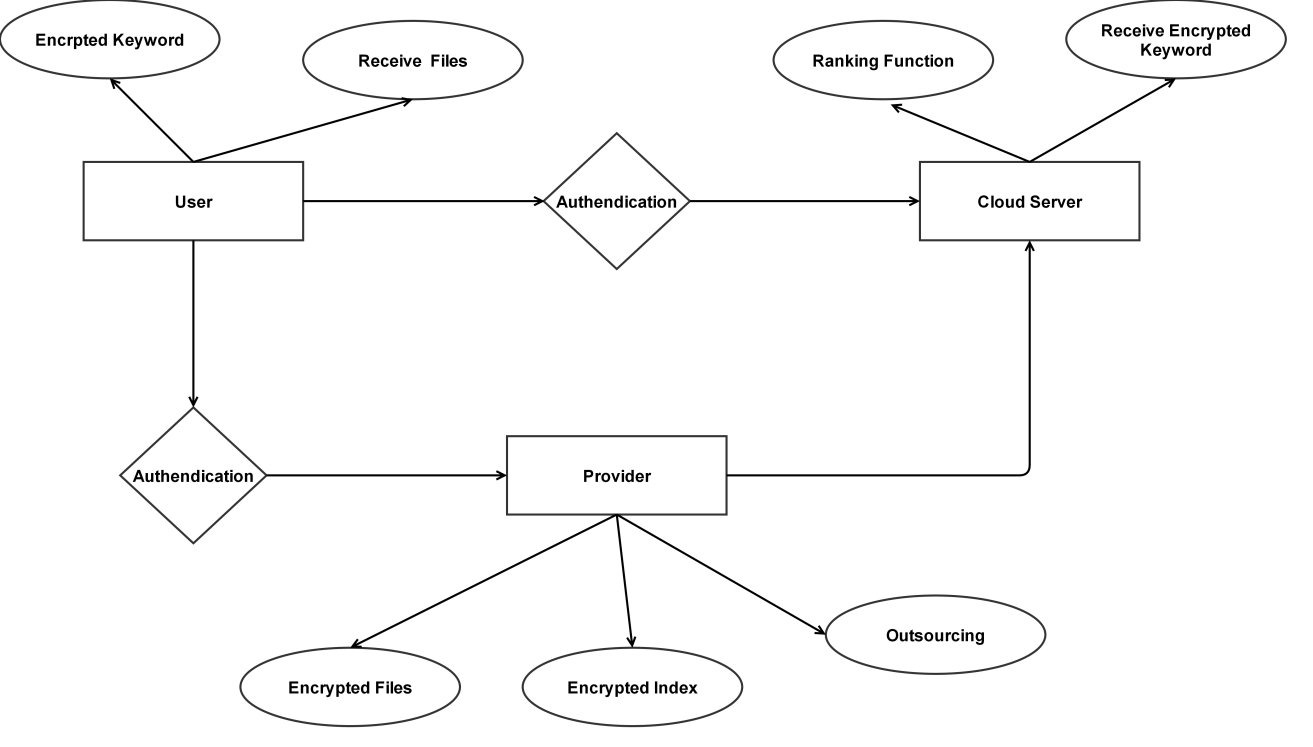


**Overall**



**Entity Relationship Diagram**

Entity-Relationship Model is an abstract and conceptual representation of data. Entity-relationship modelling is a database modelling method. It describes whether authentication between the user and server is performed correctly and the respective encrypted files and index are generated and then the image is displayed to the user.



**TESTING**

**JAVA**

Java is a programming language originally developed by James Gosling at Sun Microsystems now a subsidiary of Oracle Corporation, and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of their Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Classpath.

**Java Platform**

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code. Java bytecode instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics, threading and networking.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would, and Java suffered a reputation for poor performance. This gap has been narrowed by a number of optimization techniques introduced in the more recent JVM implementations.

**Implementations**

Sun Microsystems officially licenses the Java Standard Edition platform for Linux, Mac OS X and Solaris. Although in the past Sun has licensed Java to Microsoft, the license has expired and has not been renewed. Through a network of third-party vendors and licensees, alternative Java environments are available for these and other platforms.

Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their own. Sun sued in 1997, and in 2001 won a settlement of $20 million as well as a court order enforcing the terms of the license from Sun. As a result, Microsoft no longer ships Java with Windows, and in recent versions of Windows, Internet Explorer cannot support Java applets without a third-party plugin. Sun, and others, has made available free Java run-time systems for those and other versions of Windows.

Platform-independent Java is essential to the Java EE strategy, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications, such as Web services, Java Servlets, and Enterprise JavaBeans, as well as with embedded systems based on OSGi, using Embedded Java environments. Through the new GlassFish project, Sun is working to create a fully functional, unified open source implementation of the Java EE technologies.

Sun also distributes a superset of the JRE called the Java Development Kit (commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar and debugger.

**Performance**

Programs written in Java have a reputation for being slower and requiring more memory than those written in some other languages. However, Java programs' execution speed improved significantly with the introduction of Just-in-time compilation in 1997/1998 for Java 1.1, the addition of language features supporting better code analysis (such as inner classes, StringBuffer class, optional assertions, ect.), and optimizations in the Java Virtual Machine itself, such as HotSpot becoming the default for Sun's JVM in 2000.

To boost even further the speed performances that can be achieved using the Java language Systronix made JStik, a microcontroller based on the aJile Systems line of embedded Java processors.

**Automatic memory management**

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers be spared the burden of having to perform manual memory management. In some languages memory for the creation of objects is implicitly allocated on the stack, or explicitly allocated and deallocated from the heap. Either way, the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent 'logical' memory leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style pointer arithmetic, where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects, and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's primitive data types are not objects. Values of primitive types are either stored directly in fields (for objects) or on the stack (for methods) rather than on the heap, as commonly true for objects (but see Escape analysis). This was a conscious decision by Java's designers for performance reasons. Because of this, Java was not considered to be a pure object-oriented programming language. However, as of Java 5.0, autoboxing enables programmers to proceed as if primitive types are instances of their wrapper classes.

**MySQL**

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Sun Microsystems, a subsidiary of Oracle Corporation.

Members of the MySQL community have created several forks such as Drizzle and MariaDB. Both forks were in progress long before the Oracle acquisition (Drizzle was announced 8 months before the Sun acquisition).

Free-software projects that require a full-featured database management system often use MySQL. Such projects include (for example) WordPress, phpBB and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products including Wikipedia, Google, Drupal and Face book.

**Uses**

Many web applications use MySQL as the database component of a LAMP software stack. Its popularity for use with web applications is closely tied to the popularity of PHP, which is often combined with MySQL. Several high-traffic web sites (including Flickr, Facebook, Wikipedia, Google (though not for searches), Nokia and YouTube[11]) use MySQL for data storage and logging of user data.

**Platforms and interfaces**

MySQL code uses C and C++. The SQL parser uses yacc and a home-brewed lexer, sql\_lex.cc.

MySQL works on many different system platforms, including AIX, BSDi, FreeBSD, HP-UX, i5/OS, Linux, Mac OS X, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, eComStation, OS/2 Warp, QNX, IRIX, Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos, Tru64 and Microsoft Windows. A port of MySQL to OpenVMS also exists.

All major programming languages with language-specific APIs include Libraries for accessing MySQL databases. In addition, an ODBC interface called MyODBC allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The MySQL server and official libraries are mostly implemented in ANSI C/ANSI C++.

**Management and Graphical Frontends**

MySQL Workbench in Windows, displaying the Home Screen which streamlines use of its full capabilities

MySQL is primarily an RDBMS and therefore ships with no GUI tools to administer MySQL databases or manage data contained within. Users may use the included command-line tools, or download MySQL Frontends from various parties that have developed desktop software and web applications to manage MySQL databases, build database structure, and work with data records.

**Official**

The official MySQL Workbench is a free integrated environment developed by MySQL AB, that enables users to graphically administer MySQL databases and visually design database structure. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages but still considered the authoritative MySQL frontend, MySQL Workbench lets users manage the following:

* Database design & modeling
* SQL development — replacing MySQL Query Browser
* Database administration — replacing MySQL Administrator

MySQL Workbench is available in two editions, the regular free and open source Community Edition which may be downloaded from the MySQL website, and the proprietary Standard Edition which extends and improves the feature set of the Community Edition.

**Third party**

Several other third-party proprietary and free graphical administration applications (or "Frontends") are available that integrate with MySQL and enable users to work with database structure and data visually. Some well-known frontends are:

* phpMyAdmin - a free Web-based frontend widely installed by Web hosts worldwide, since it is developed in PHP and only requires the LAMP stack to run.
* HeidiSQL - a full featured free frontend that runs on Windows, and can connect to local or remote MySQL servers to manage databases, tables, column structure, and individual data records. Also supports specialised GUI features for date/time fields and enumerated multiple-value fields.
* Navicat - a series of proprietary graphical database management applications, developed for Windows, Macintosh and Linux.
* Other available proprietary MySQL frontends include Adminer, Aqua Data Studio, dbForge Studio for MySQL, Epictetus, Oracle SQL Developer, SchemaBank, SQLyog, SQLPro SQL Client, Toad and Toad Data Modeler.

**Deployment**

MySQL can be built and installed manually from source code, but this can be tedious so it is more commonly installed from a binary package unless special customizations are required. On most Linux distributions the package management system can download and install MySQL with minimal effort, though further configuration is often required to adjust security and optimization settings.

Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well.

It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP based web application or as a standalone database server. Much of MySQL's appeal originates in its relative simplicity and ease of use, which is enabled by an ecosystem of open source tools such as phpMyAdmin.

In the medium range, MySQL can be scaled by deploying it on more powerful hardware, such as a multi-processor server with gigabytes of memory.

There are however limits to how far performance can scale on a single server, so on larger scales, multi-server MySQL deployments are required to provide improved performance and reliability. A typical high-end configuration can include a powerful master database which handles data write operations and is replicated to multiple slaves that handle all read operations. The master server synchronizes continually with its slaves so in the event of failure a slave can be promoted to become the new master, minimizing downtime. Further improvements in performance can be achieved by caching the results from database queries in memory using memcached, or breaking down a database into smaller chunks called shards which can be spread across a number of distributed server clusters.

**Features**

As of April 2009, MySQL offers MySQL 5.1 in two different variants: the MySQL Community Server and Enterprise Server. They have a common code base and include the following features:

* A broad subset of ANSI SQL 99, as well as extensions
* Cross-platform support
* Stored procedures
* Triggers
* Cursors
* Updatable Views
* True Varchar support
* INFORMATION\_SCHEMA
* Strict mode
* X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using Oracle's InnoDB engine
* Independent storage engines (MyISAM for read speed, InnoDB for transactions and referential integrity, MySQL Archive for storing historical data in little space)
* Transactions with the InnoDB, BDB and Cluster storage engines; savepoints with InnoDB
* SSL support
* Query caching
* Sub-SELECTs (i.e. nested SELECTs)
* Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master, no automatic support for multiple masters per slave.
* Full-text indexing (Index\_(database)) and searching using MyISAM engine
* Embedded database library
* Partial Unicode support (UTF-8 and UCS-2 encoded strings are limited to the BMP)
* Partial ACID compliance (full compliance only when using the non-default storage engines InnoDB, BDB and Cluster)
* Shared-nothing clustering through MySQL Cluster
* Hot backup (via mysqlhotcopy) under certain conditions

The developers release monthly versions of the MySQL Enterprise Server. The sources can be obtained either from MySQL's customer-only Enterprise site or from MySQL's Bazaar repository, both under the GPL license. The MySQL Community Server is published on an unspecified schedule under the GPL and contains all bug fixes that were shipped with the last MySQL Enterprise Server release. Binaries are no longer provided by MySQL for every release of the Community Server.

**Distinguishing features**

MySQL implements the following features, which some other RDBMS systems may not:

* Multiple storage engines, allowing one to choose the one that is most effective for each table in the application (in MySQL 5.0, storage engines must be compiled in; in MySQL 5.1, storage engines can be dynamically loaded at run time):
  + Native storage engines (MyISAM, Falcon, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, Cluster, Berkeley DB, EXAMPLE, and Maria)
  + Partner-developed storage engines (InnoDB, solidDB, NitroEDB, Infobright (formerly Brighthouse), Kickfire, XtraDB, IBM DB2[22])
  + Community-developed storage engines (memcache\_engine, httpd, PBXT, Revision Engine)
  + Custom storage engines
* Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

**Product History**

Milestones in MySQL development include:

* Original development of MySQL by Michael Widenius and David Axmark beginning in 1994
* First internal release on 23 May 1995
* Windows version was released on 8 January 1998 for Windows 95 and NT
* Version 3.23: beta from June 2000, production release January 2001
* Version 4.0: beta from August 2002, production release March 2003 (unions)
* Version 4.01: beta from August 2003, Jyoti adopts MySQL for database tracking
* Version 4.1: beta from June 2004, production release October 2004 (R-trees and B-trees, subqueries, prepared statements)
* Version 5.0: beta from March 2005, production release October 2005 (cursors, stored procedures, triggers, views, XA transactions)

The developer of the Federated Storage Engine states that "The Federated Storage Engine is a proof-of-concept storage engine", but the main distributions of MySQL version 5.0 included it and turned it on by default. Documentation of some of the short-comings appears in "MySQL Federated Tables: The Missing Manual".

* Sun Microsystems acquired MySQL AB on 26 February 2008.
* Version 5.1: production release 27 November 2008 (event scheduler, partitioning, plugin API, row-based replication, server log tables)

Version 5.1 contained 20 known crashing and wrong result bugs in addition to the 35 present in version 5.0.

MySQL 5.1 and 6.0 showed poor performance when used for data warehousing — partly due to its inability to utilize multiple CPU cores for processing a single query.

* Oracle acquired Sun Microsystems on January 27, 2010.Oracle and Sun

**Future releases**

The MySQL 6 roadmap outlines support for:

* Referential integrity and Foreign key support for all storage engines is targeted for release in MySQL 6.1 (although it has been present since version 3.23.44 for InnoDB).
* Support for supplementary Unicode characters, beyond the 65,536 characters of the Basic Multilingual Plane (BMP); announced for MySQL 6.0.
* A new storage engine called Falcon. A preview of Falcon is available on MySQL's website.

The 2006 roadmap for future versions plans support for parallelization.

**Support and licensing**

Via MySQL Enterprise MySQL AB offers support itself, including a 24/7 service with 30-minute response time. The support team has direct access to the developers as necessary to handle problems. In addition, it hosts forums and mailing lists, employees and other users are often available in several IRC channels providing assistance.

In addition to official product support from Sun, other companies offer support and services related to usage of MySQL. For example, Pythian offers full database administration, architecture, optimization and training services. Percona and 42sql offer services related to optimization and Monty Program Ab offers non-recurring engineering such as patches to MySQL. OpenQuery provides MySQL training.

Buyers of MySQL Enterprise have access to binaries and software certified for their particular operating system, and access to monthly binary updates with the latest bug-fixes. Several levels of Enterprise membership are available, with varying response times and features ranging from how to and emergency support through server performance tuning and system architecture advice. The MySQL Network Monitoring and Advisory Service monitoring tool for database servers is available only to MySQL Enterprise customers.

Potential users can install MySQL Server as free software under the GNU General Public License (GPL), and the MySQL Enterprise subscriptions include a GPL version of the server, with a traditional proprietary version available on request at no additional cost for cases where the intended use is incompatible with the GPL.

Both the MySQL server software itself and the client libraries use dual-licensing distribution. Users may choose the GPL,[29] which MySQL has extended with a FLOSS License Exception. It allows Software licensed under other OSI-compliant open source licenses, which are not compatible to the GPL, to link against the MySQL client libraries.

Customers that do not wish to follow the terms of the GPL may purchase a proprietary license.Like many open-source programs, MySQL has trademarked its name, which others may use only with the trademark holder's permission.

**SOFTWARE TESTING**

**GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**Types of Tests**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**Performance Test**

The Performance test ensures that the output is produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

***Acceptance Testing***

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledge will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updating process

**Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**FEASIBILITY STUDY:**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**ECONOMICAL FEASIBILITY** :

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

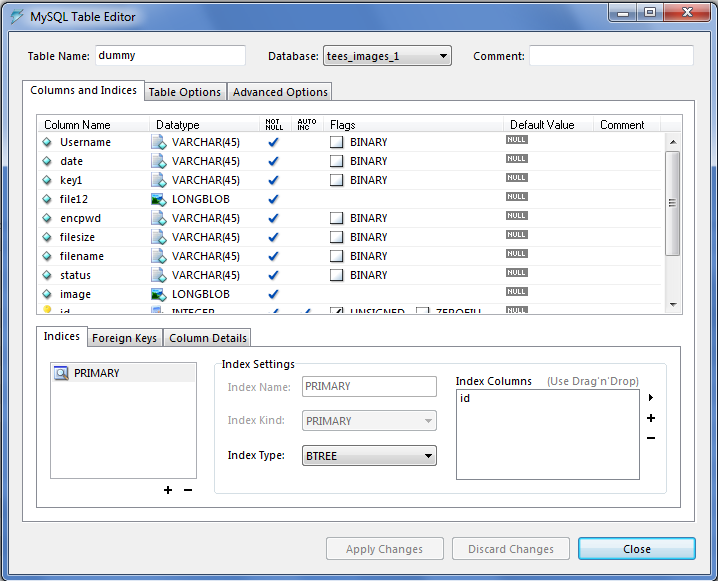
### **TECHNICAL FEASIBILITY :**

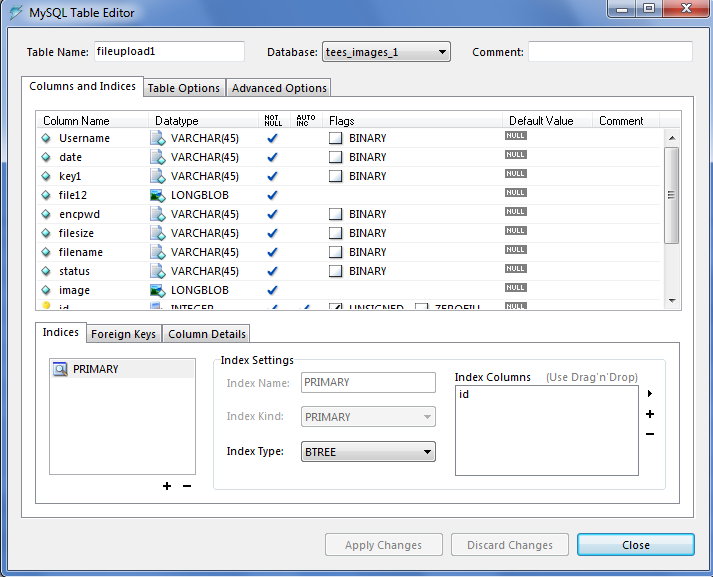
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

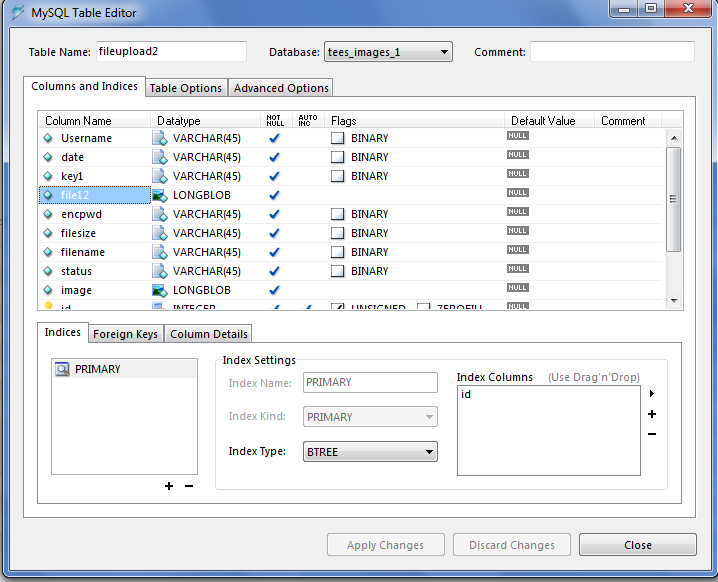
**SOCIAL FEASIBILITY:**

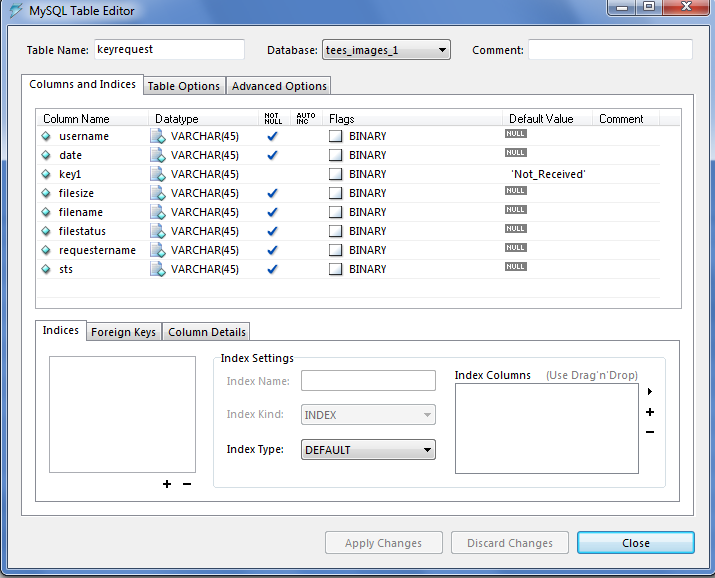
The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**DATABASE DESIGN:**

****

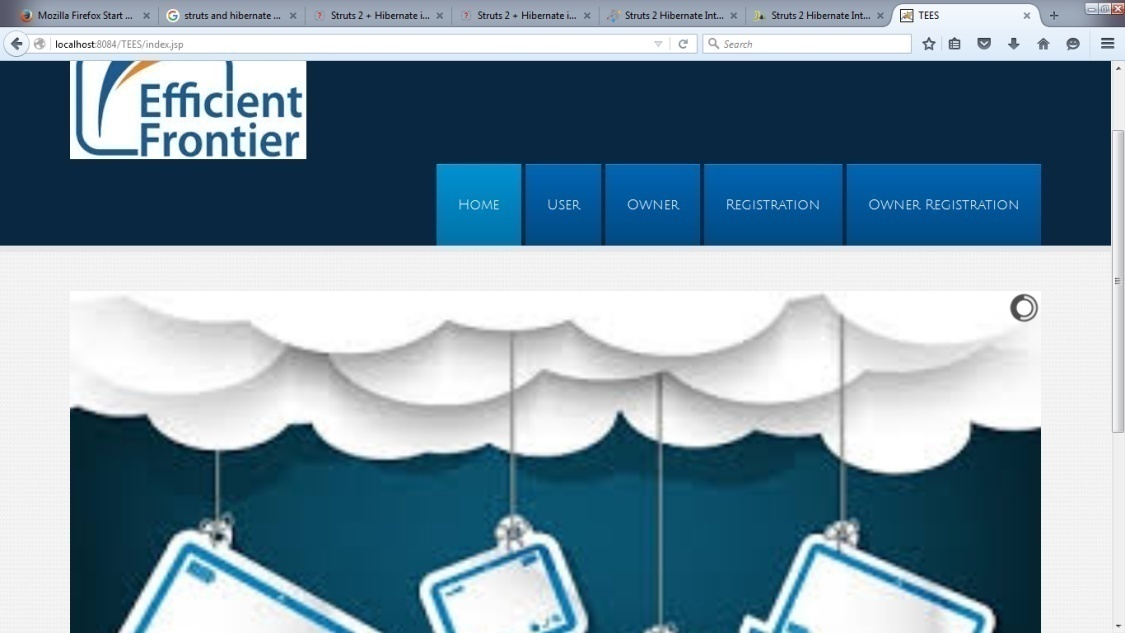
****

****

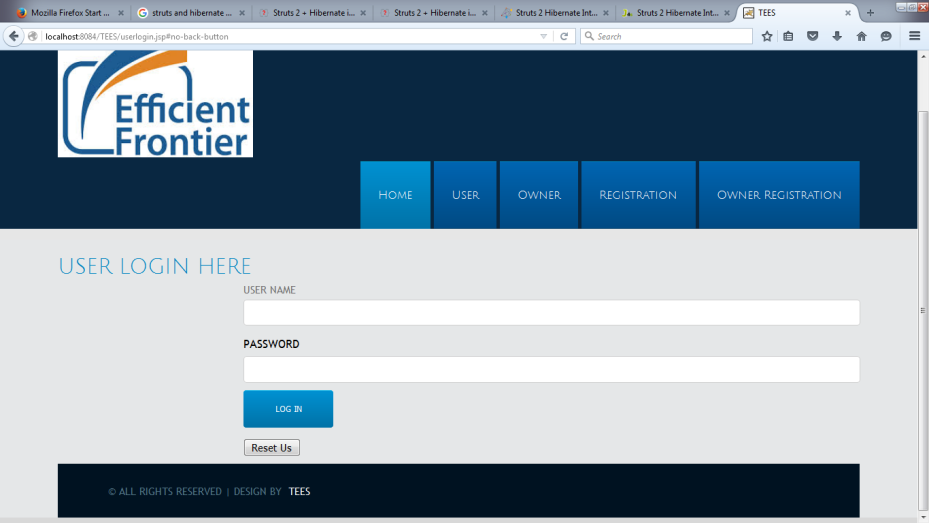
****

**SCREENSHOTS**

**HOMEPAGE**

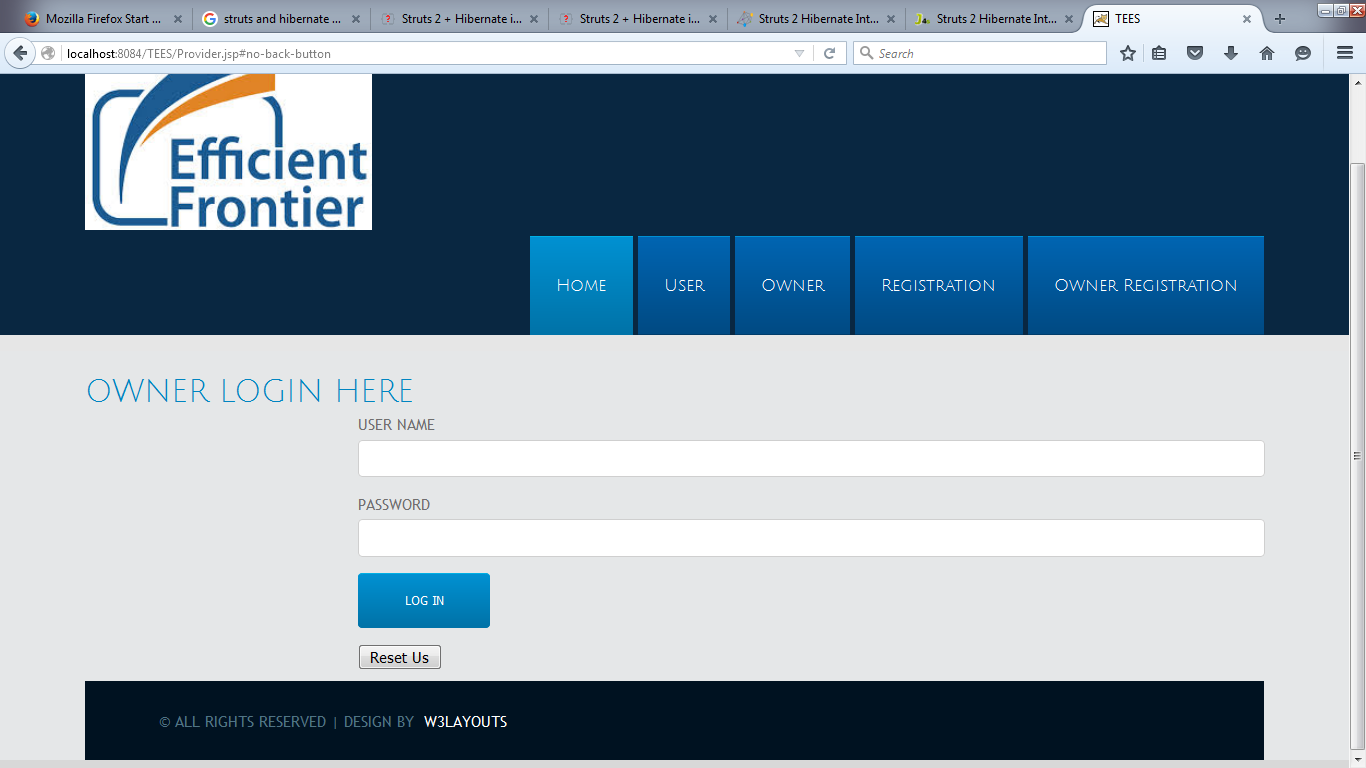


**USER LOGIN**

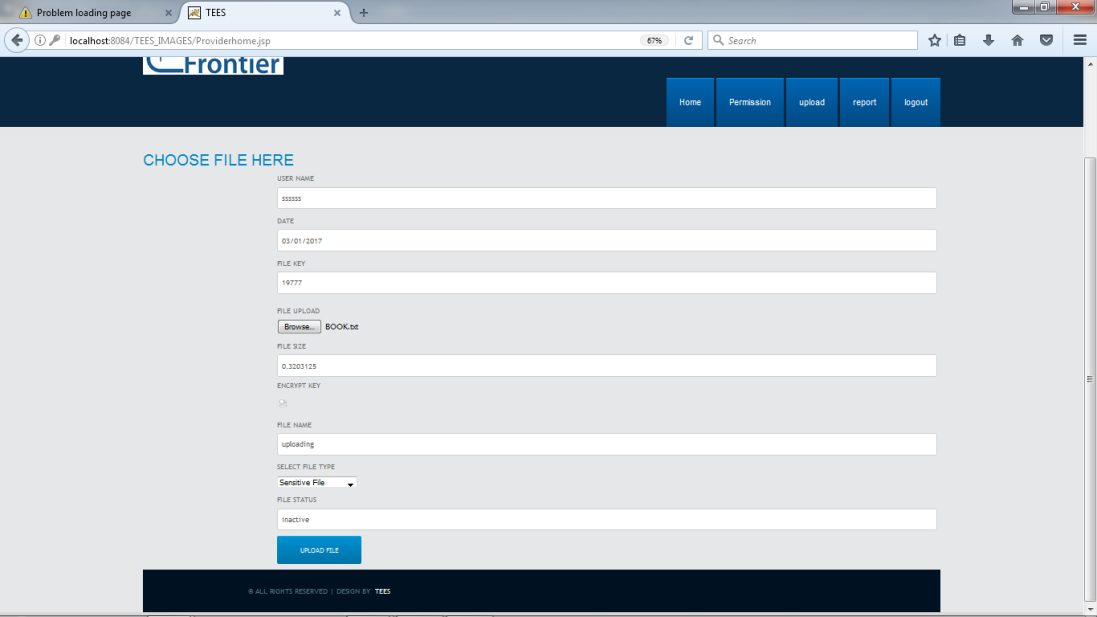


38

**OWNER LOGIN**

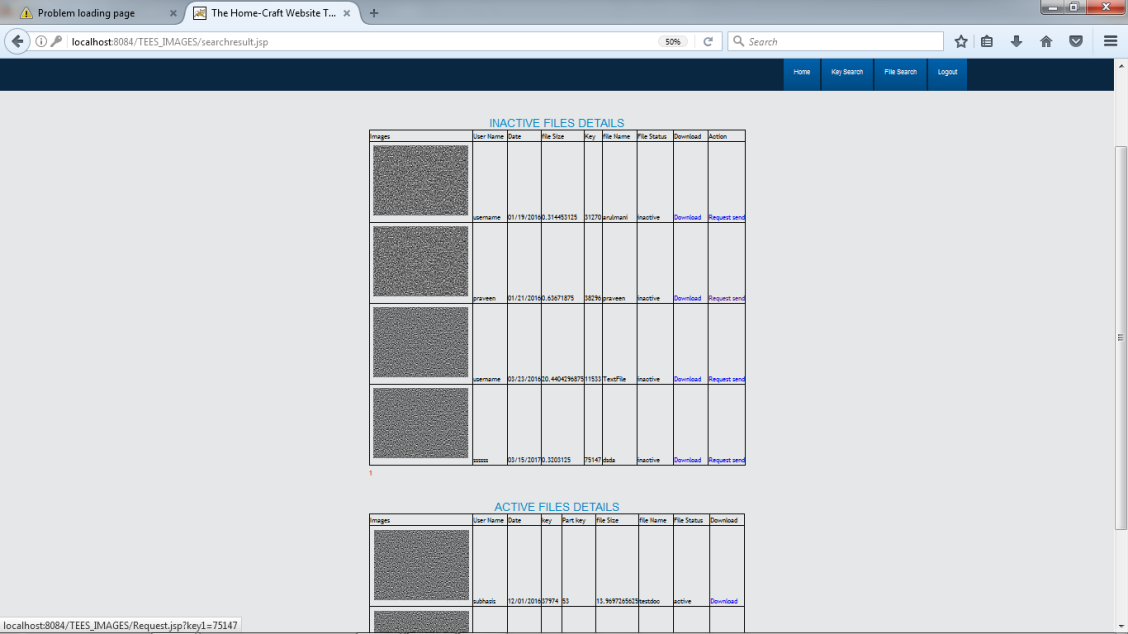


**OWNER UPLOADING FILES**

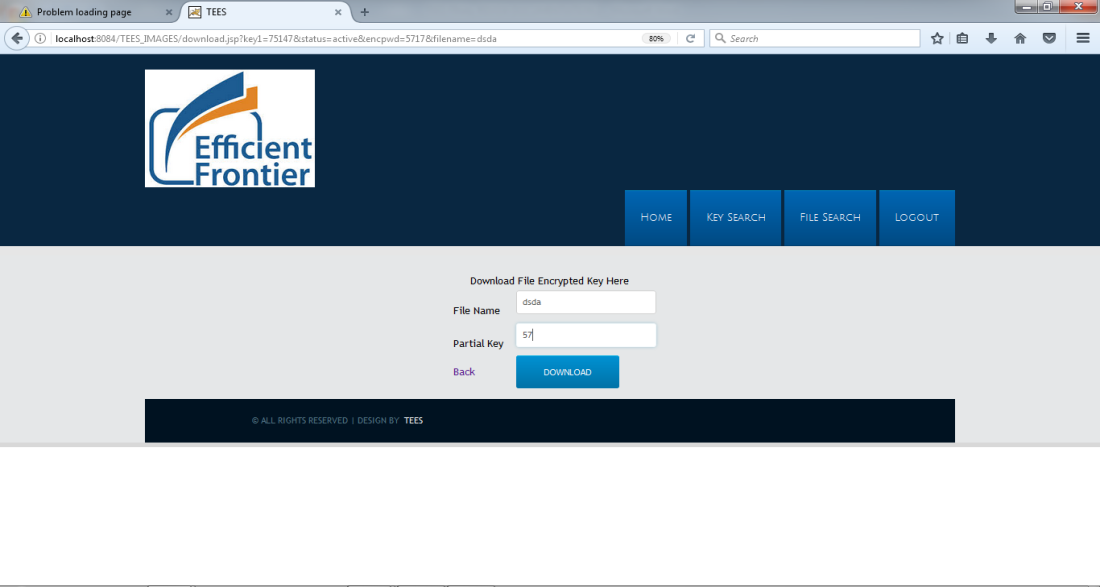
****

39

**USER REQUESTING FILES**

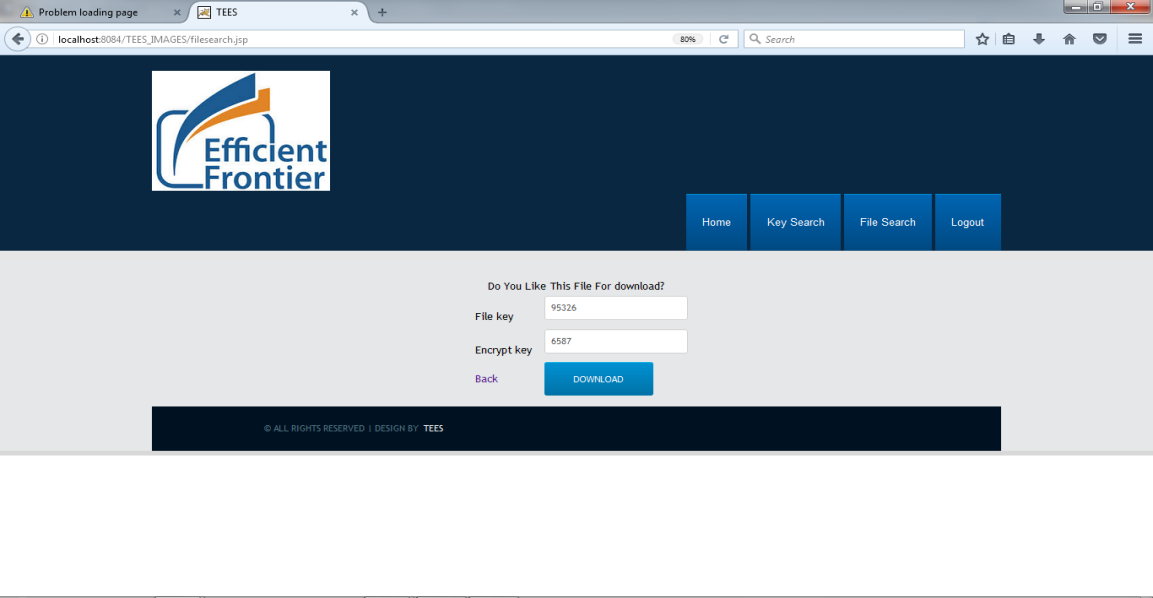


**ENTERING FILE KEY**

****

40

**FINAL OUTPUT**



**SAMPLE CODING**

**Encryption:**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import sun.misc.BASE64Encoder;

public class Encryption {

public String Encryption1(String value) throws InvalidKeyException, IllegalBlockSizeException, BadPaddingException

{

String Encry="";

try {

String plainData=value,decryptedText;

KeyGenerator keyGen = KeyGenerator.getInstance("AES");

keyGen.init(128);

SecretKey secretKey = keyGen.generateKey();

Cipher aesCipher=null;

try {

aesCipher = Cipher.getInstance("AES");

} catch (NoSuchPaddingException ex) {

Logger.getLogger(Encryption.class.getName()).log(Level.SEVERE, null, ex);

}

aesCipher.init(Cipher.ENCRYPT\_MODE,secretKey);

byte[] byteDataToEncrypt = plainData.getBytes();

byte[] byteCipherText = aesCipher.doFinal(byteDataToEncrypt);

Encry = new BASE64Encoder().encode(byteCipherText);

try {

aesCipher.init(Cipher.DECRYPT\_MODE,secretKey,aesCipher.getParameters());

} catch (InvalidAlgorithmParameterException ex) {

}

byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);

decryptedText = new String(byteDecryptedText);

System.out.println("\n Plain Data : "+plainData+" \n Cipher Data : "+Encry+" \n Decrypted Data : "+decryptedText);

} catch (NoSuchAlgorithmException ex) {

Logger.getLogger(Encryption.class.getName()).log(Level.SEVERE, null, ex);

}

return Encry;

}

}

**File Upload:**

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

import java.util.Random;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.PBEParameterSpec;

import javax.servlet.ServletException;

import javax.servlet.annotation.MultipartConfig;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import javax.servlet.http.Part;

import org.apache.commons.fileupload.FileItem;

import org.apache.commons.fileupload.disk.DiskFileItemFactory;

import org.apache.commons.fileupload.servlet.ServletFileUpload;

import connection.DatabaseConnection;

@WebServlet("/FileUpload")

@MultipartConfig(maxFileSize = 16177215) // upload file's size up to 16MB

public class FileUpload extends HttpServlet {

private static final long serialVersionUID = 1L;

private static final String UPLOAD\_DIRECTORY = "upload";

private static final int MEMORY\_THRESHOLD = 1024 \* 1024 \* 3; // 3MB

private static final int MAX\_FILE\_SIZE = 1024 \* 1024 \* 40; // 40MB

private static final int MAX\_REQUEST\_SIZE = 1024 \* 1024 \* 50; // 50MB

private String dbURL = "jdbc:mysql://localhost:3306/tees";

private String dbUser = "root";

private String dbPass = "admin";

private String message;

private Object conn;

protected void doPost(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

String username1 = request.getParameter("userName");

String date1 = request.getParameter("startdate");

String key1 = request.getParameter("key");

String encpwd1 = request.getParameter("enckey");

String filesize1 = request.getParameter("filesize");

String filename1=request.getParameter("filename");

String privatekey1 = "Not Updated";

String outsourcekey1 = "Not Updated";

String updatedkey1 = "Not Updated";

InputStream inputStream12 = null;

Part filePart = request.getPart("file12");

if (filePart != null) {

// prints out some information for debugging

System.out.println(filePart.getName());

System.out.println(filePart.getSize());

System.out.println(filePart.getContentType());

// obtains input stream of the upload file

inputStream12 = filePart.getInputStream();

}

Connection conn = null; // connection to the database

String message = null; // message will be sent back to client

HttpSession session = request.getSession(true);

try {

String file, password;

FileInputStream inFile;

FileOutputStream outFile;

file = getFileName(filePart);

password = request.getParameter("enckey");

// inFile = new FileInputStream("D:\\2015 Projects\\IdentityBased\\web\\Files\\upload\\" + file);

outFile = new FileOutputStream("D:\\arulmani\\TEES\\web\\Files\\upload" + file + ".aes");

try {

PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray());

SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("PBEWithMD5AndDES");

SecretKey passwordKey = keyFactory.generateSecret(keySpec);

byte[] salt = new byte[8];

Random rnd = new Random();

rnd.nextBytes(salt);

int iterations = 100;

PBEParameterSpec parameterSpec = new PBEParameterSpec(salt, iterations);

Cipher cipher = Cipher.getInstance("PBEWithMD5AndDES");

cipher.init(Cipher.ENCRYPT\_MODE, passwordKey, parameterSpec);

outFile.write(salt);

byte[] input = new byte[64];

int bytesRead;

/\* while ((bytesRead = inFile.read(input)) != -1) {

byte[] output = cipher.update(input, 0, bytesRead);

if (output != null) {

outFile.write(output);

}

}\*/

byte[] output = cipher.doFinal();

if (output != null) {

outFile.write(output);

}

// inFile.close();

outFile.flush();

outFile.close();

} catch (Exception e) {

System.out.println(e.getMessage());

System.out.println(e.getLocalizedMessage());

System.out.println(e.getCause());

}

if (!ServletFileUpload.isMultipartContent(request)) {

// if not, we stop here

PrintWriter writer = response.getWriter();

writer.println("Error: Form must has enctype=multipart/form-data.");

writer.flush();

return;

}

// configures upload settings

DiskFileItemFactory factory = new DiskFileItemFactory();

// sets memory threshold - beyond which files are stored in disk

factory.setSizeThreshold(MEMORY\_THRESHOLD);

// sets temporary location to store files

factory.setRepository(new File(System.getProperty("java.io.tmpdir")));

ServletFileUpload upload = new ServletFileUpload(factory);

String uploadPath = getServletContext().getRealPath("")

+ File.separator + UPLOAD\_DIRECTORY;

String path = "D:\\arulmani\\TEES\\web\\Files\\upload" + File.separator + UPLOAD\_DIRECTORY;

// creates the directory if it does not exist

File uploadDir = new File(path);

if (!uploadDir.exists()) {

uploadDir.mkdir();

}

try {

// parses the request's content to extract file data

@SuppressWarnings("unchecked")

List<FileItem> formItems = upload.parseRequest(request);

if (formItems != null && formItems.size() > 0) {

// iterates over form's fields

for (FileItem item : formItems) {

// processes only fields that are not form fields

if (!item.isFormField()) {

String fileName = new File(item.getName()).getName();

String filePath = path + File.separator + fileName;

File storeFile = new File(filePath);

// saves the file on disk

item.write(storeFile);

}

}

}

} catch (Exception ex) {

request.setAttribute("message", "There was an error: " + ex.getMessage());

}

// redirects client to message page

// connects to the database

String verify1 = "Request";

String keystatus1 = "Not Verified";

DriverManager.registerDriver(new com.mysql.jdbc.Driver());

conn = DriverManager.getConnection(dbURL, dbUser, dbPass);

DatabaseConnection db1 = new DatabaseConnection();

ResultSet rs1 = null;

// constructs SQL statement

String sql = "INSERT INTO fileupload1(Username, date, key1, file12, encpwd, filesize, filename) values(?,?,?,?,?,?,?)";

PreparedStatement statement = conn.prepareStatement(sql);

statement.setString(1, username1);

statement.setString(2, date1);

statement.setString(3, key1);

if (inputStream12 != null) {

// fetches input stream of the upload file for the blob column

statement.setBlob(4, inputStream12);

}

statement.setString(5, encpwd1);

statement.setString(6, filesize1);

statement.setString(7, filename1);

// sends the statement to the database server

int row = statement.executeUpdate();

if (row > 0) {

session.setAttribute("msg", "Successfully Uploaded");

response.sendRedirect("Providerhome.jsp");

} else {

session.setAttribute("msg", "Database Error");

response.sendRedirect("Providerhome.jsp");

}

} catch (SQLException ex) {

message = "ERROR: " + ex.getMessage();

ex.printStackTrace();

} finally {

if (conn != null) {

// closes the database connection

try {

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

// sets the message in request scope

request.setAttribute("Message", message);

// forwards to the message page

//getServletContext().getRequestDispatcher("/moviedetails.jsp").forward(request, response);

}

}

private String getFileName(Part part) {

String contentDisp = part.getHeader("content-disposition");

System.out.println("content-disposition header= " + contentDisp);

String[] tokens = contentDisp.split(";");

for (String token : tokens) {

if (token.trim().startsWith("filename")) {

return token.substring(token.indexOf("=") + 2, token.length() - 1);

}

}

return "";

}

}

**Registration:**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

import connection.DatabaseConnection;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.ResultSet;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\*

\* @author arulmani

\*/

public class OwnerRegistration extends HttpServlet {

/\*\*

\* Processes requests for both HTTP

\* <code>GET</code> and

\* <code>POST</code> methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session = request.getSession(true);

try {

String firstname = request.getParameter("firstname");

String lastname = request.getParameter("lastname");

String userName = request.getParameter("userName");

String mail = request.getParameter("mail");

String mobile = request.getParameter("mobile");

String dateofbirth = request.getParameter("dateofbirth");

String city = request.getParameter("city");

String address = request.getParameter("address");

String pwd = request.getParameter("pwd");

String cpwd = request.getParameter("cpwd");

String gender = request.getParameter("gender");

Connection con = null;

DatabaseConnection db1 = new DatabaseConnection();

String query = "select \* from ownerregister where username='" + userName + "'";

ResultSet i1 = db1.Select(query);

if (!i1.next()) {

int i = db1.Insert("insert into ownerregister values('" + firstname + "','" + lastname + "','" + userName + "','" + mail + "','" + mobile + "','" + dateofbirth + "','" + city + "','" + address + "','" + pwd + "','" + cpwd + "','" + gender + "','No')");

session.setAttribute("msg", "Successfully Register");

response.sendRedirect("index.jsp");

} else {

session.setAttribute("msg", "This username already exist! Change username");

response.sendRedirect("OwnerRegistration.jsp");

}

} catch (Exception e) {

System.out.print("e");

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**Login:**

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import connection.DatabaseConnection;

public class userlogin extends HttpServlet {

@Override

protected void service(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

//processRequest(request, response);

HttpSession session = request.getSession(true);

try {

Connection con = null;

Statement st = null;

DatabaseConnection db1 = new DatabaseConnection();

String userName = request.getParameter("userName");

String password = request.getParameter("password");

String query = "select \* from register where username='" + userName + "' and password='" + password + "'";

ResultSet i = db1.Select(query);

if (!i.next()) {

session.setAttribute("msg", "Username & password incorrect!...");

response.sendRedirect("userlogin.jsp");

} else {

session.setAttribute("msg", "Successfully Login");

session.setAttribute("username", userName);

session.setAttribute("password", password);

response.sendRedirect("userhome.jsp");

}

} catch (Exception e) {

System.out.print("e");

}

}

}

**File Request:**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

import connection.DatabaseConnection;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\*

\* @author arulmani

\*/

@WebServlet(urlPatterns = {"/Request1"})

public class Request1 extends HttpServlet {

/\*\*

\* Processes requests for both HTTP

\* <code>GET</code> and

\* <code>POST</code> methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

HttpSession session=request.getSession(true);

try {

Connection con=null;

Statement st=null;

ResultSet rf=null;

DatabaseConnection db1=new DatabaseConnection();

String username=request.getParameter("username");

String userid=request.getParameter("userid");

String mail=request.getParameter("mail");

String mobile=request.getParameter("mobile");

String city=request.getParameter("city");

int i=db1.Insert("insert into request values('"+username+"','"+userid+"','"+mail+"','"+mobile+"','"+city+"')");

session.setAttribute("msg","Request Sent Successfully....");

//session.setAttribute("userid",rf.getString(2).toString());

response.sendRedirect("searchresult.jsp");

}

catch(Exception e)

{

System.out.print("e");

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**CONCLUSION AND FUTURE ENHANCEMENT**

**CONCLUSION:**

In this work, we proposed a novel encrypted search system Tees over the mobile cloud, which improves network traffic and search time efficiency compared with the traditional system. We started with a thorough analysis of the traditional encrypted search system and analysed its bottlenecks in the mobile cloud: network traffic and search time inefficiency. Then we developed an efficient architecture of Tees which is suitable for the mobile cloud to address these issues, where we utilized the TMT module and the RSBS algorithm to cope with the inefficient search time issue, while a trapdoor compression method was employed to reduce network traffic costs. Finally our evaluation study experimentally demonstrates the performance advantages of Tees.

**FUTURE ENHANCEMENT:**

Our IEEE concept has to be keep on information in a large number of cloud computing environments by adding extra features like listed below

1. To design search schemes which allow multi keyword query and provide result similar it ranking for valuable data retrieval, instead of returning undifferentiated results
2. To prevent cloud server from learning additional information from dataset and index, and to meet privacy requirements.
3. Above goals on functionality and privacy should be achieved with low communication and computation over head

All these process can be give as update to our project users throughout the world via Using this Application.

**REFERENCES:**

[1] Q. Chai and G. Gong, “Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers,” in Communications (ICC), 2012 IEEE International Conference on. IEEE, 2012, pp. 917– 922.

[2] S. Hou, T. Uehara, S. Yiu, L. C. Hui, and K. Chow, “Privacy preserving multiple keyword search for confidential investigation of remote forensics,” in Multimedia Information Networking and Security (MINES), 2011 Third International Conference on. IEEE, 2011, pp. 595–599.

[3] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-preserving multi-keyword ranked search over encrypted cloud data,” Parallel and Distributed Systems, IEEE Transactions on, vol. 25, no. 1, pp. 222–233, 2014.

[4] B. Wang, S. Yu, W. Lou, and Y. T. Hou, “Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud,” in INFOCOM, 2014 Proceedings IEEE.

[5] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50–55, 2008.

[6] X. Yu and Q. Wen, “Design of security solution to mobile cloud storage,” in Knowledge Discovery and Data Mining. Springer, 2012, pp. 255–263.

[7] D. Huang, “Mobile cloud computing,” IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter, 2011.

[8] O. Mazhelis, G. Fazekas, and P. Tyrvainen, “Impact of storage acquisition intervals on the cost-efficiency of the private vs. public storage,” in Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012, pp. 646–653.

[9] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, “Virtualized in-cloud security services for mobile devices,” in Proceedings of the First Workshop on Virtualization in Mobile Computing. ACM, 2008, pp. 31–35.

[10] J. Oberheide and F. Jahanian, “When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments,” in Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications. ACM, 2010, pp. 43–48.