ABSTRACT

Many schemes have been recently advanced for storing data on multiple clouds. Distributing data over different cloud storage providers (CSPs) automatically provides users with a certain degree of information leakage control, for no single point of attack can leak all the information. However, unplanned distribution of data chunks can lead to high information disclosure even while using multiple clouds. In this paper, we study an important information leakage problem caused by unplanned data distribution in multicloud storage services. Then, we present **StoreSim**, an information leakage aware storage system in multicloud. StoreSim aims to store syntactically similar data on the same cloud, thus minimizing the user's information leakage across multiple clouds. We design an approximate algorithm to efficiently generate similarity-preserving signatures for data chunks based on MinHash and Bloom filter, and also design a function to compute the information leakage based on these signatures. Next, we present an effective storage plan generation algorithm based on clustering for distributing data chunks with minimal information leakage across multiple clouds.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

With the increasingly rapid uptake of devices such as laptops, cellphones and tablets, users require an ubiquitous and massive network storage to handle their ever-growing digital lives. To meet these demands, many cloud-based storage and file sharing services such as Dropbox, Google Drive and Amazon S3, have gained popularity due to the easy-to-use interface and low storage cost. However, these centralized cloud storage services are criticized for grabbing the control of users' data, which allows storage providers to run analytics for marketing and advertising . Also, the information in users' data can be leaked e.g., by means of malicious insiders, backdoors, bribe and coercion. One possible solution to reduce the risk of information leakage is to employ multicloud storage systems in which no single point of attack can leak all the information. A malicious entity, such as the one revealed in recent attacks on privacy , would be required to coerce all the different CSPs on which a user might place her data, in order to get a complete picture of her data. Put simply, as the saying goes, do not put all the eggs in one basket.

Yet, the situation is not so simple. CSPs such as Dropbox, among many others, employe sync-like protocols to synchronize the local file to remote file in their centralized clouds . Every local file is partitioned into small chunks and these chunks are hashed with fingerprinting algorithms such as SHA-1, MD5. Thus, a file's contents can be uniquely identified by this list of hashes. For each update of local file, only chunks with changed hashes will be uploaded to the cloud. This synchronization based on hashes is different from diff -like protocols that are based on comparing two versions of the same file line by line and can detect the exact updates and only upload these updates in a patch style .

Instead, the hash-based synchronization model needs to upload the whole chunks with changed hashes to the cloud. Thus, in the multicloud environment, two chunks differing only very slightly can be distributed to two different clouds. The following motivating example will show that if chunks of a user's data are assigned to different CSPs in an unplanned manner, the information leaked to each CSP can be higher than expected. Suppose that we have a storage service with three CSPs S1; S2; S3 and a user's dataset D. All the user's data will be firstly chunked and then uploaded to different clouds. The dataset D is represented as a set of hashes generated by each data chunk. This scenario is shown in Figure 1. In addition, we consider that the data chunks are distributed to different clouds in a round robin (RR) way.

Apparently, RR is good for balancing the storage load and each cloud thus obtains the same amount of data. However, the same amount of datadoes not necessarily mean the same amount of information. For example, if we find that the set of chunks fC3;C6;C9g are almost same, it means S3 actually obtains the information equivalent to that in only one chunk. If all other chunks are different, S1 and S2 obtain three times as much information of data. The problem does not exist in a single storage cloud such as Dropbox since users have no other choice but to give all their information to only one cloud.

When the storage is in the multicloud, we have the opportunity to minimize the total information that is leaked to each CSP. The optimal case is that each CSP obtains the same amount of information. In our example, data distribution based on RR can achieve the optimal result only if all the chunks are different. However this is not the case in cloud storage service due to two reasons:

1) Frequent modifications of files by users result in large amount of similar chunks1;
2) Similar chunks across files, due to which existing CSPs use the data deduplication technique.

Centralized cloud storage services are criticized for grabbing the control of users' data, which allows storage providers to run analytics for marketing and advertising . Also, the information in users' data can be leaked e.g., by means of malicious insiders, backdoors, bribe and coercion. One possible solution to reduce the risk of information leakage is to employ multicloud storage systems in which no single point of attack can leak all the information.

Instead, the hash-based synchronization model needs to upload the whole chunks with changed hashes to the cloud. Thus, in the multicloud environment, two chunks differing only very slightly can be distributed to two different clouds. The following motivating example will show that if chunks of a user's data are assigned to different CSPs in an unplanned manner, the information leaked to each CSP can be higher than expected.
Suppose that we have a storage service with three CSPs S1; S2; S3 and a user's dataset D. All the user's data will be firstly chunked and then uploaded to different clouds. The dataset D is represented as a set of hashes generated by each data chunk.In addition, we consider that the data chunks are distributed to different clouds.

CHAPTER 2

LITERATURE SURVEY

2.LITERATURE SURVEY

DepSky: Dependable and Secure Storage in a Cloud-of-Clouds:

DepSky, a system that improves the availability, integrity, and confidentiality of information stored in the cloud through the encryption, encoding, and replication of the data on diverse clouds that form a cloud-of-clouds. We deployed our system using four commercial clouds and used PlanetLab to run clients accessing the service from different countries. We observed that our protocols improved the perceived availability, and in most cases, the access latency, when compared with cloud providers individually. Moreover, the monetary costs of using DepSky in this scenario is at most twice the cost of using a single cloud, which is optimal and seems to be a reasonable cost, given the benefits.

On the Duality of Resilience and privacy:

Protecting information has long been an important problem. We would like to protect ourselves from the risk of loss: think of the library of Alexandria; and from unauthorized access: consider the very business of the 'Scandal Sheets', going back centuries. This has never been more true than today when vast quantities of data (dare one say lesser quantities of information) are stored on computer systems, and routinely moved around the Internet, at almost no cost. Computer and communication systems are both fragile and vulnerable, and so the risk of catastrophic loss or theft is potentially much higher. A single keystroke can delete a public database, or expose a private dataset to the world. In this paper, I consider the problems of providing resilience against loss, and against unacceptable access as a *dual*. Here, we see that two apparently different solutions to different technical problems may be transformed into one another, and hence give better insight into both problems.

NCCloud: A Network-Coding-Based Storage System in a Cloud-of-Clouds

We present a proxy-based storage system for fault-tolerant multiple-cloud storage called NCCloud, which achieves cost-effective repair for a permanent single-cloud failure. NCCloud is built on top of a network-coding-based storage scheme called the functional

minimum-storage regenerating (FMSR) codes, which maintain the same fault tolerance and data redundancy as in traditional erasure codes (e.g., RAID-6), but use less repair traffic and, hence, incur less monetary cost due to data transfer. One key design feature of our FMSR codes is that we relax the encoding requirement of storage nodes during repair, while preserving the benefits of network coding in repair. We implement a proof-of-concept prototype of NCCloud and deploy it atop both local and commercial clouds. We validate that FMSR codes provide significant monetary cost savings in repair over RAID-6 codes, while having comparable response time performance in normal cloud storage operations such as upload/download.

Algorithms for Delta Compression and Remote File Synchronization

Delta compression and remote file synchronization techniques are concerned with efficient file transfer over a slow communication link in the case where the receiving party already has a similar file (or files). This problem arises naturally, e.g., when distributing updated versions of software over a network or synchronizing personal files between different accounts and devices. More generally, the problem is becoming increasingly common in many networkbased applications where files and content are widely replicated, frequently modified, and cut and reassembled in different contexts and packagings. In this chapter, we survey techniques, software tools, and applications for delta compression, remote file synchronization, and closely related problems. We first focus on delta compression, where the sender knows all the similar files that are held by the receiver. In the second part, we survey work on the related, but in many ways quite different, problem of remote file synchronization, where the sender does not have a copy of the files held by the receiver. Work supported by NSF CAREER Award NSF CCR-0093400 and by Intel Corporation.

Personal Cloud Storage Benchmarks and Comparison

The large amount of space offered by personal cloud storage services (e.g., Dropbox and OneDrive), together with the possibility of synchronizing devices seamlessly, keep attracting customers to the cloud. Despite the high public interest, little information about system design and actual implications on performance is available when selecting a cloud storage

service. Systematic benchmarks to assist in comparing services and understanding the effects of design choices are still lacking. This paper proposes a methodology to understand and benchmark personal cloud storage services. Our methodology unveils their architecture and capabilities. Moreover, by means of repeatable and customizable tests, it allows the measurement of performance metrics under different workloads. The effectiveness of the methodology is shown in a case study in which 11 services are compared under the same conditions. Our case study reveals interesting differences in design choices. Their implications are assessed in a series of benchmarks. Results show no clear winner, with all services having potential for improving performance. In some scenarios, the synchronization of the same files can take 20 times longer. In other cases, we observe a wastage of twice as much network capacity, questioning the design of some services. Our methodology and results are thus useful both as benchmarks and as guidelines for system design.

CHAPTER 3

SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

3.1 THE STUDY OF THE SYSTEM

- To conduct studies and analyses of an operational and technological nature, and
- To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how

the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

3.2 INPUT & OUTPUT REPRESENTATION

Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

a. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
b. Select methods for presenting information.
c. Create document, report, or other formats that contain information produced by the system.

3.3 INTRODUCTION TO CLOUD COMPUTING

What is cloud computing?

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

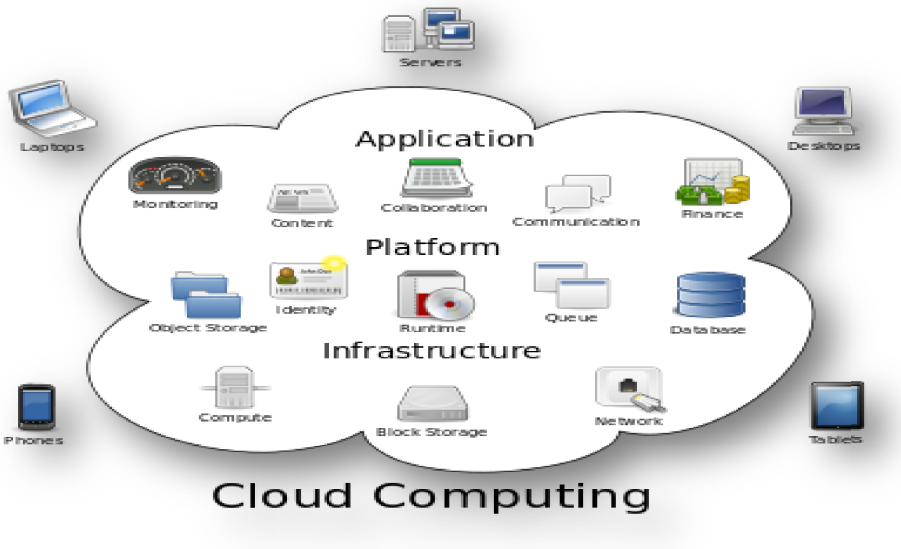


Figure 3.3. 1: Structure of cloud computing

How Cloud Computing Works?

The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games.

The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

Characteristics and Services Models:

The salient characteristics of cloud computing based on the definitions provided by the National Institute of Standards and Terminology (NIST) are outlined below:

- **On-Demand Self-Service**: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- **Broad Network Access**: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- **Resource Pooling**: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- **Rapid Elasticity**: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- **Measured Service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

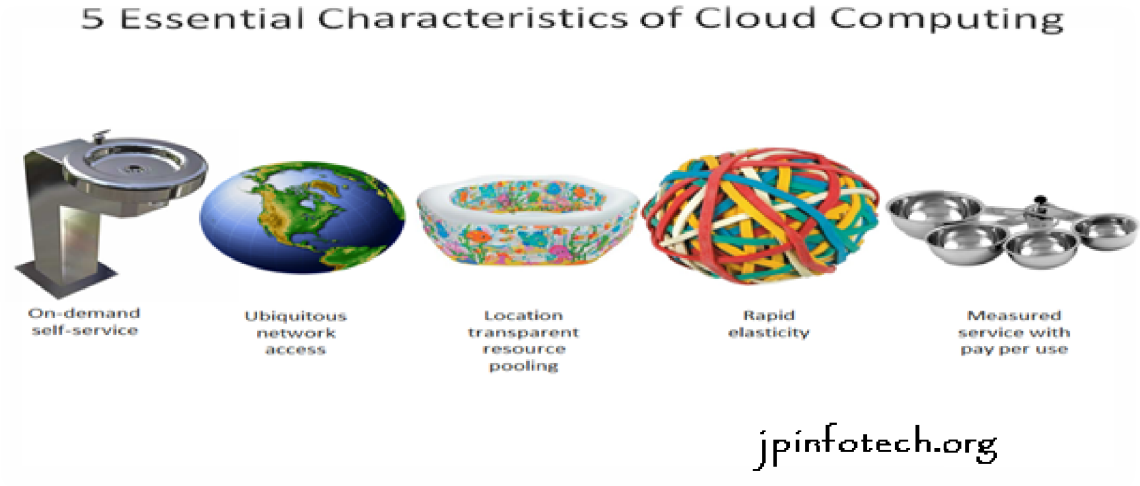


Figure 3.3. 2: Characteristics of cloud computing

Services Models:

Cloud Computing comprises three different service models, namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The three service models or layer are completed by an end user layer that encapsulates the end user perspective on cloud services. The model is shown in figure below. If a cloud user accesses services on the infrastructure layer, for instance, she can run her own applications on the resources of a cloud infrastructure and remain responsible for the support, maintenance, and security of these applications herself. If she accesses a service on the application layer, these tasks are normally taken care of by the cloud service provider.

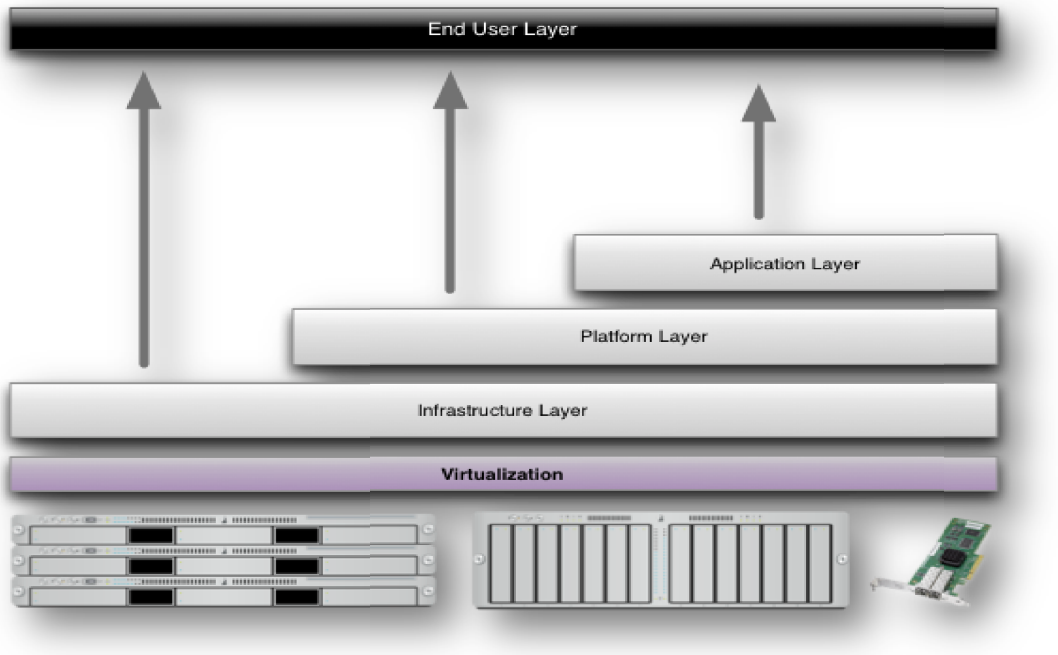


Figure 3.3. 3: Structure of service models

Benefits of cloud computing:

1. **Achieve Economies Of Scale** – increase volume output or productivity with fewer people. Your cost per unit, project or product plummets.
2. **Reduce Spending On Technology Infrastructure.** Maintain easy access to your information with minimal upfront spending. Pay as you go (weekly, quarterly or yearly), based on demand.
3. **Globalize Your Workforce On The Cheap.** People worldwide can access the cloud, provided they have an Internet connection.
4. **Streamline Processes.** Get more work done in less time with less people.
5. **Reduce Capital Costs.** There's no need to spend big money on hardware, software or licensing fees.
6. **Improve Accessibility.** You have access anytime, anywhere, making your life so much easier!
7. **Monitor Projects More Effectively.** Stay within budget and ahead of completion cycle times.
8. **Less Personnel Training Is Needed.** It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.

9. **Minimize Licensing New Software.** Stretch and grow without the need to buy expensive software licenses or programs.
10. **Improve Flexibility.** You can change direction without serious "people" or "financial" issues at stake.

Advantages:

1. **Price:** Pay for only the resources used.
2. **Security**: Cloud instances are isolated in the network from other instances for improved security.
3. **Performance:** Instances can be added instantly for improved performance. Clients have access to the total resources of the Cloud's core hardware.
4. **Scalability:** Auto-deploy cloud instances when needed.
5. **Uptime:** Uses multiple servers for maximum redundancies. In case of server failure, instances can be automatically created on another server.
6. **Control:** Able to login from any location. Server snapshot and a software library lets you deploy custom instances.
7. **Traffic:** Deals with spike in traffic with quick deployment of additional instances to handle the load.

3.4 Introduction to Java

Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed

- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

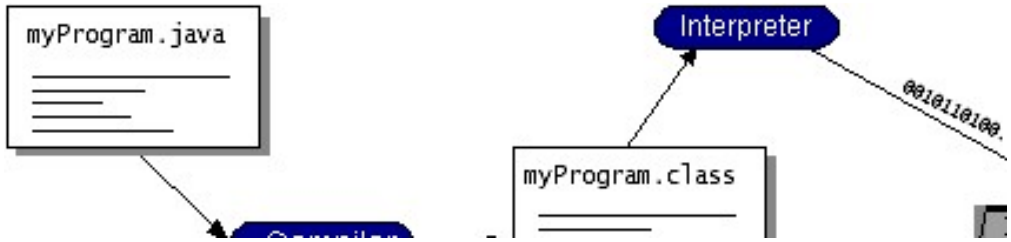


Figure 3.4. 1: Java Program Run

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

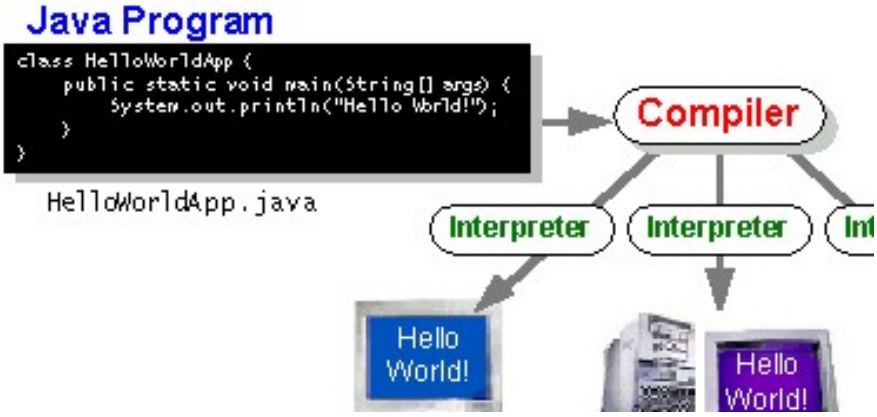


Figure 3.4. 2: Java As Platform Independent

The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

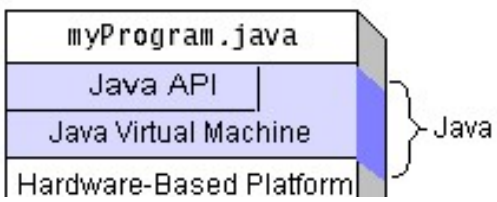


Figure 3.4.3: Java Platform

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than **native** code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The Essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software Components**: Known as JavaBeans™, can plug into existing component architectures.
- **Object Serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.

- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server

database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. **SQL Level API**

   The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user.

2. **SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. **JDBC must be implemental on top of common database interfaces**

The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. **Provide a Java interface that is consistent with the rest of the Java system**

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. **Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:

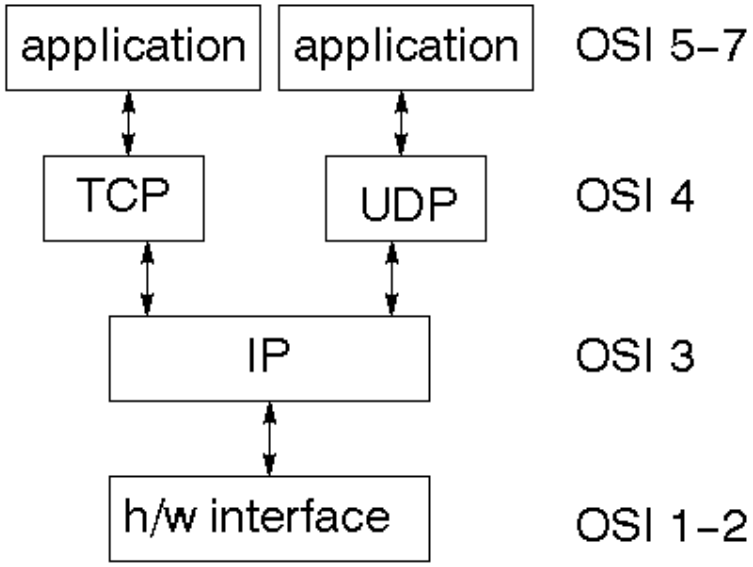


Figure 3.4.4:TCP/IP Stack

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address

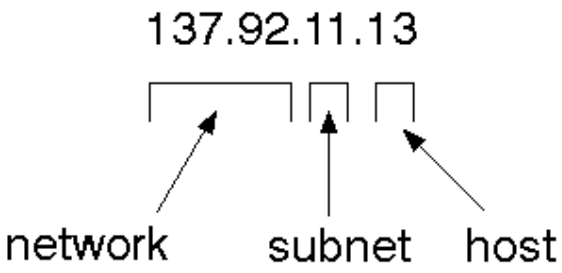


Figure 3.4. 5: 32 Bit address

The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with `Read File` and `Write File` functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be `AF_INET` for IP communications, `protocol` will be zero, and `type` will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

JFree Chart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

J2ME (Java 2 Micro edition):-

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1. General J2ME architecture

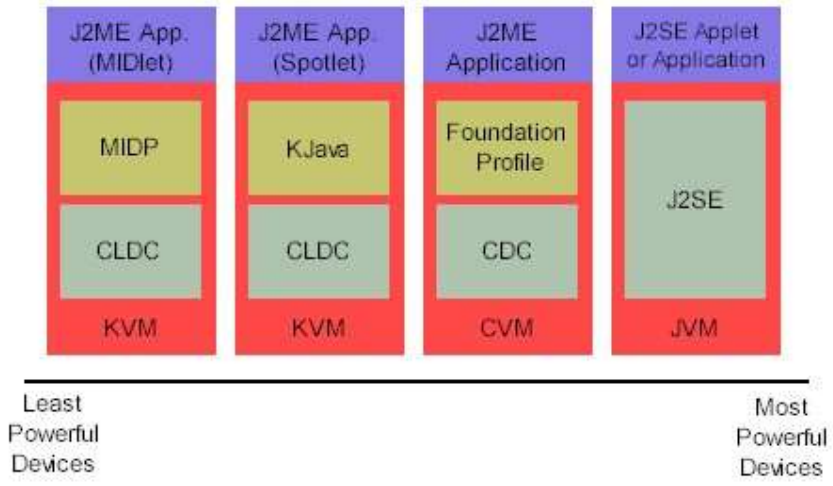


Figure 3.4. 6: General J2ME Architecture

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

* Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

4. Configurations overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5. J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both

KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

* java.lang

* java.io

* java.util

* javax.microedition.io

* javax.microedition.lcdui

* javax.microedition.midlet

* javax.microedition.rms

3.5 SYSTEM REQUIREMENTS

3.5.1 Hardware Requirements:

• RAM: 2GB and Higher

• Processor: Intel i3 and above

• Hard Disk: 500GB Minimum

3.5.2 Software Requirements:

• Operating System : Windows

• NetBean IDE : 8.0.1 Version

• Database : MySQL

• Programming Language : Java

• Front End : HTML, CSS

CHAPTER 4

DESIGN ANALYSIS

4.DESIGN ANALYSIS

4.1 SYSTEM ARCHITECTURE :

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.

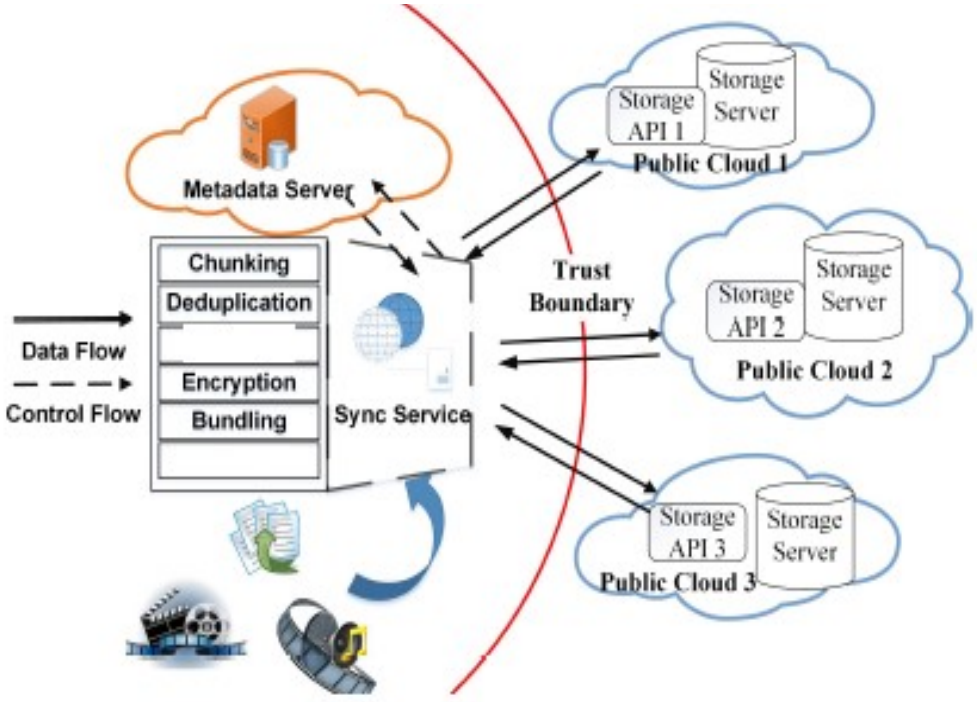


Figure 4. 1: 3-Tier Architecture

The three-tier software architecture (a three layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and

can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems

Advantages of Three-Tier:

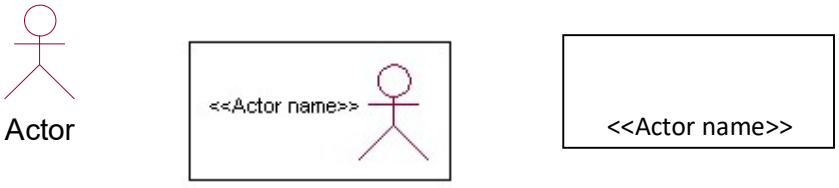
- Separates functionality from presentation.
- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.
- Effective network performance.

4.2 UML DIAGRAMS :

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



5.2. 1: Garphical Representation

An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

- Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

a. System Administrator
b. Customer
c. Customer Care

Identification of usecases:

Usecase: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:

A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What usecases will support and maintains the system?

Flow of Events

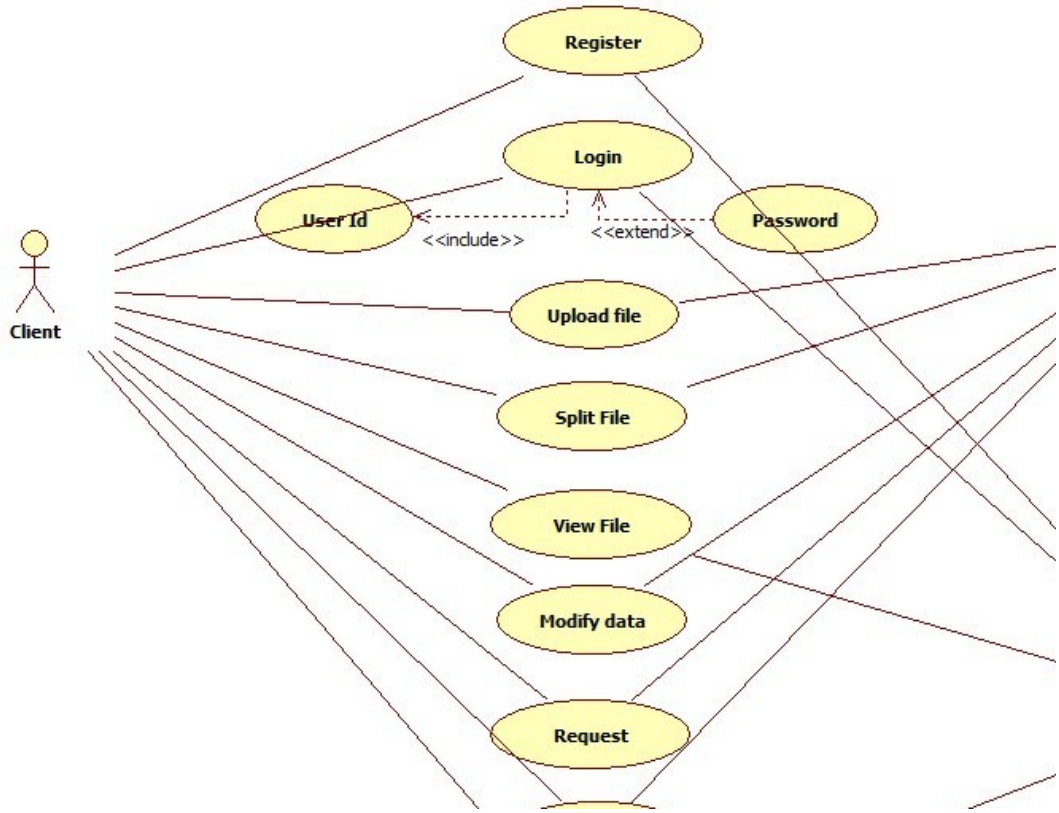
A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

4.2.1 Construction of Use Case diagrams:

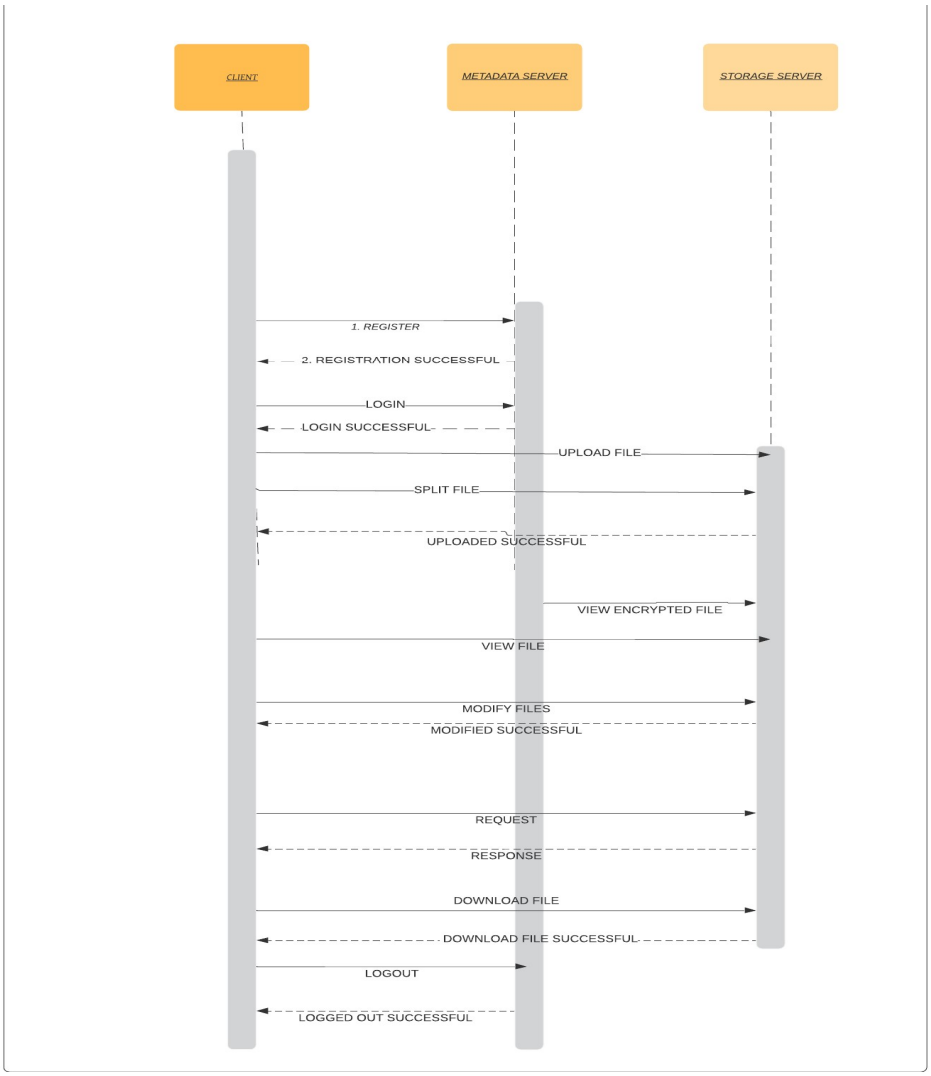
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



4.2.1. 1: Use Case Diagram

4.2.2 SEQUENCE DIAGRAMS:

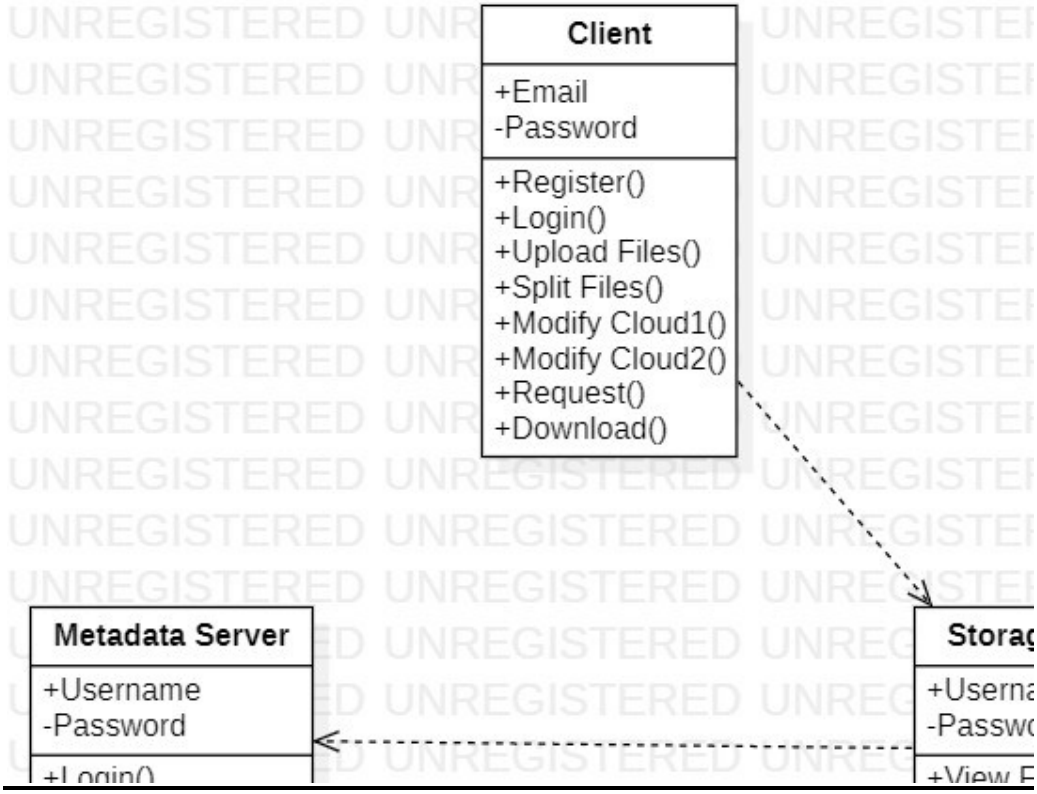
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



4.2.2. 1: Sequence diagram

4.2.3. CLASS DIAGRAM:

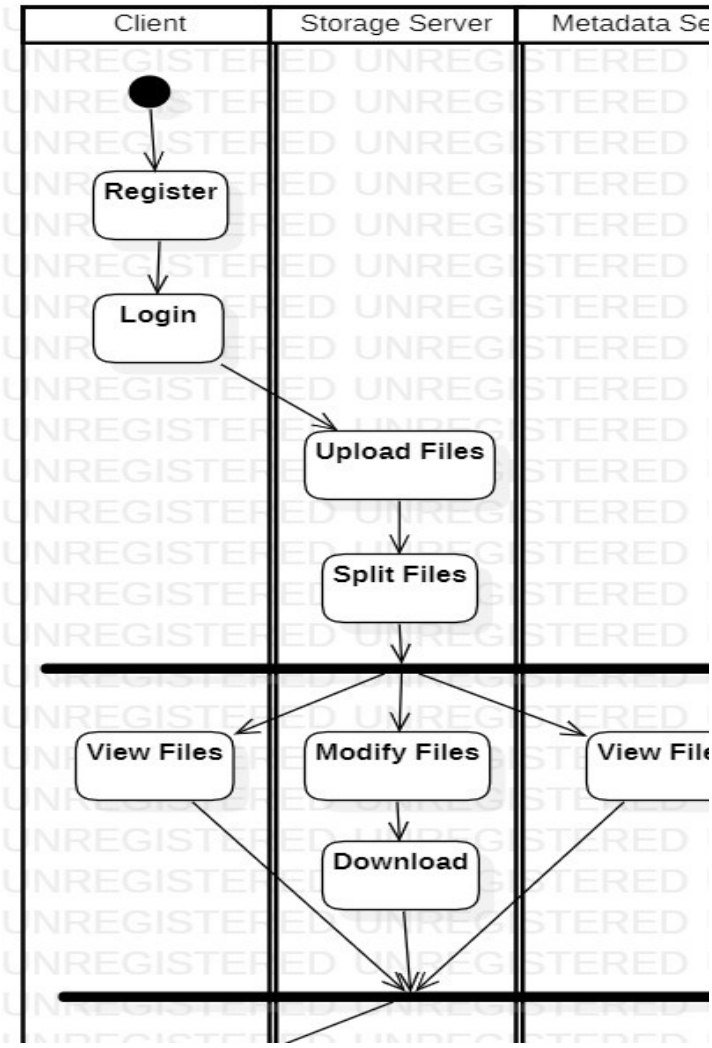
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



4.2.3. 1: Class Diagram

4.2.4 ACTIVITY DIAGRAM:

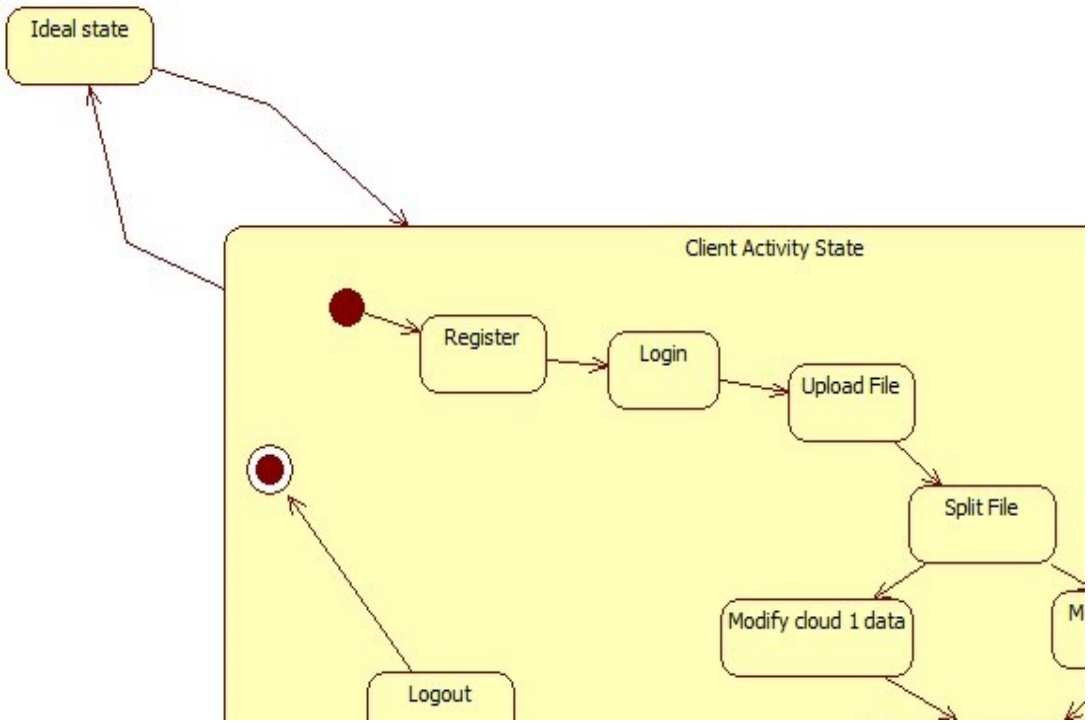
Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



4.2.4. 1: Activity Diagram

4.2.5 STATECHART DIAGRAM:

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination. Statechart diagrams are also used for forward and reverse engineering of a system.



4.2.5. 1: State Chart Diagrm

CHAPTER 5
IMPLEMENTATION

5. IMPLEMENTATION

5.1 MODULES :

There are 4 Modules in the optimization information leakage in multicloud storage.They are as follows:

1. **Data Owner.**
2. **Meta Data Server.**
3. **Cloud Service Provider.**
4. **Data User.**

DATA OWNER:

In this module, we develop the Customer features functionalities. Customer first register his/her details and login. Customer can outsource sensitive and valuable data to cloud by encrypting data and splitting data in to multiple parts.
Data owner has option to modify data which is uploaded to cloud. In this process when user updates data stored in cloud1 with data which is already available in cloud2 then total data will be visible in cloud1 only. In order to solve this problem owner will check data similarity using minhash and data matching percentage is calculated and refer to user where to upload data.

METADATA SERVER :

Metadata servers are used to store the metadata database about the information of files, CSPs and users,which usually are structured data representing the whole cloud file system.

CLOUD SERVICE PROVIDER:

In this module, we design the Cloud functionalities. The Cloud entity can view all customer details, file upload details and customer file download details. In this module, we use the DriveHQ Cloud Service API for the Cloud Integration and develop the project.

We consider a system of s storage servers S1, . . . , Ss, which stores part of data uploaded by data owner. We assume that each server appropriately authenticates user. For simplicity and without loss of generality, we focus on the read/update storage abstraction of which exports two operations.

DATA OWNER:

Data user module is used to check user data download by requesting file from cloud.The read routine fetches the stored value from the servers.

CHAPTER 6

OUTPUT AND SCREENSHOTS

7.OUTPUT AND SCREENSHOTS

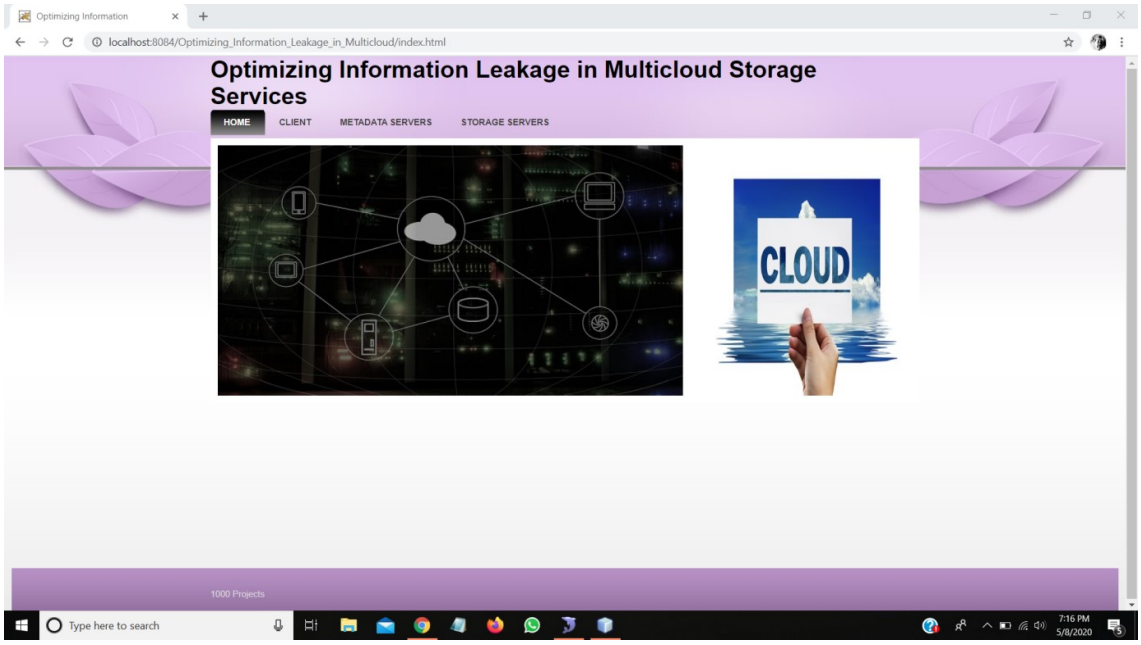


Figure 6. 1: HOMEPAGE

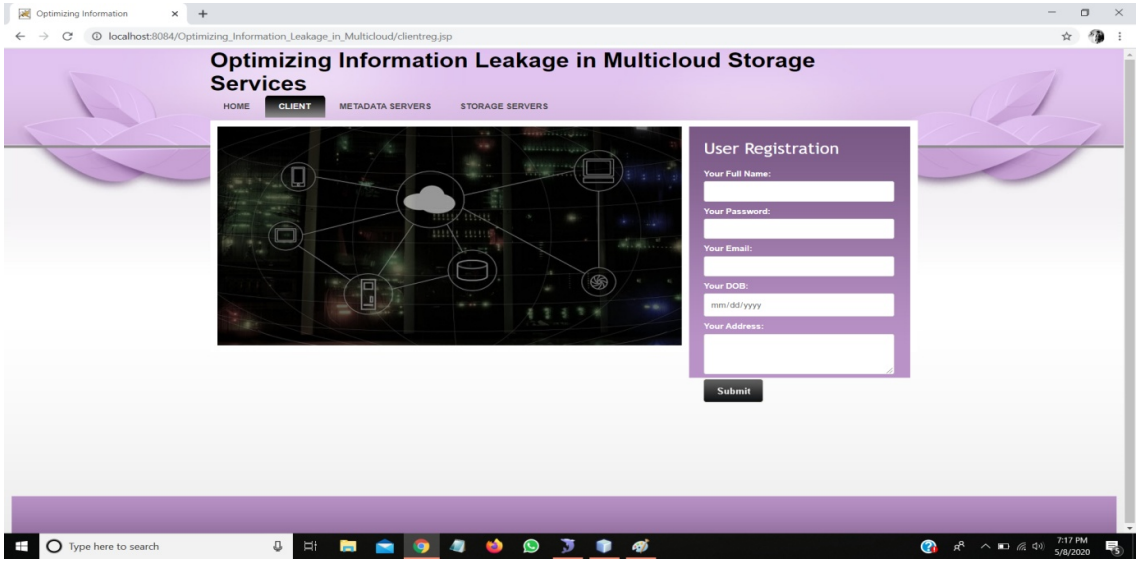


Figure 6. 2: CLIENT REGISTRATION

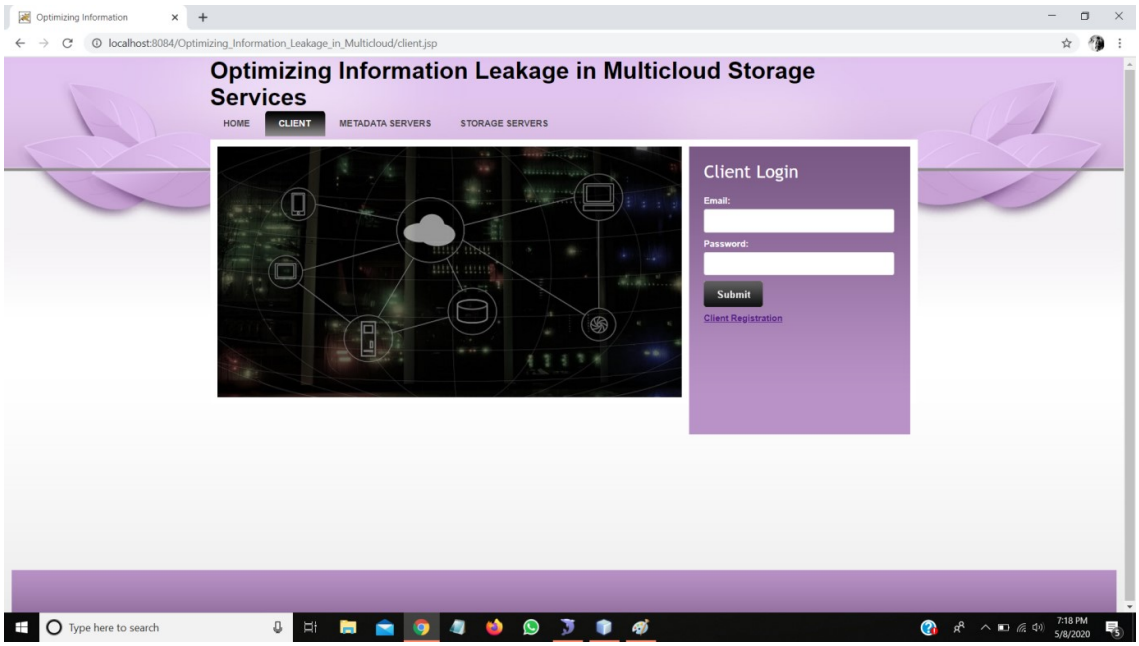


Figure 6. 3: CLIENT LOGIN PAGE

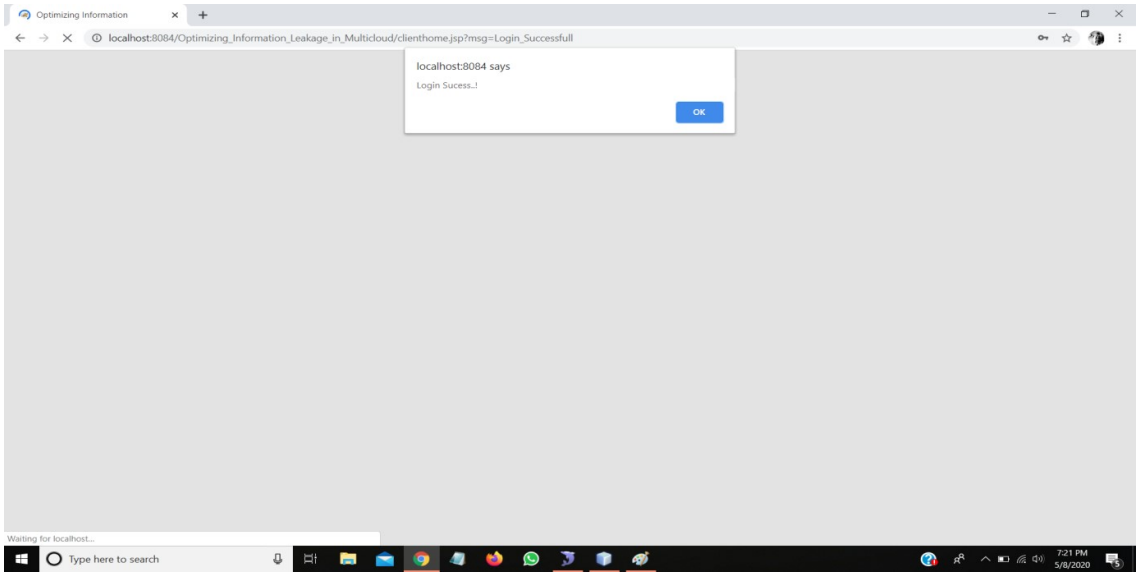


Figure 6. 4: CLIENT LOGIN SUCCESSFUL

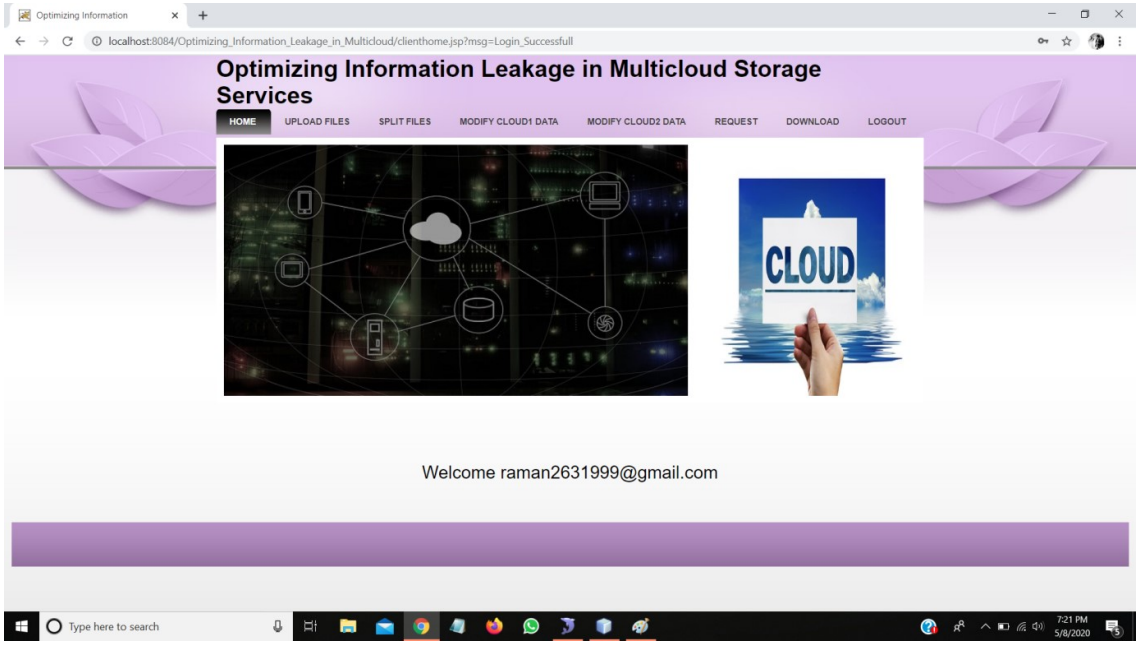


Figure 6. 5: CLIENT HOME PAGE

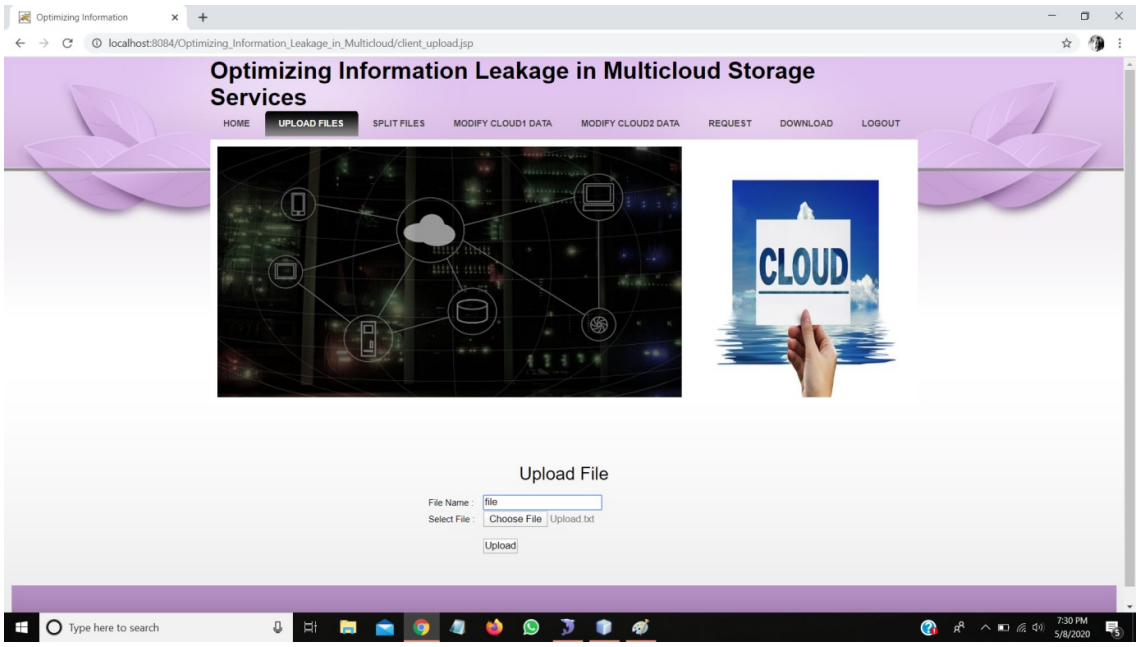


Figure 6. 6: CLIENT UPLOAD FILE PAGE

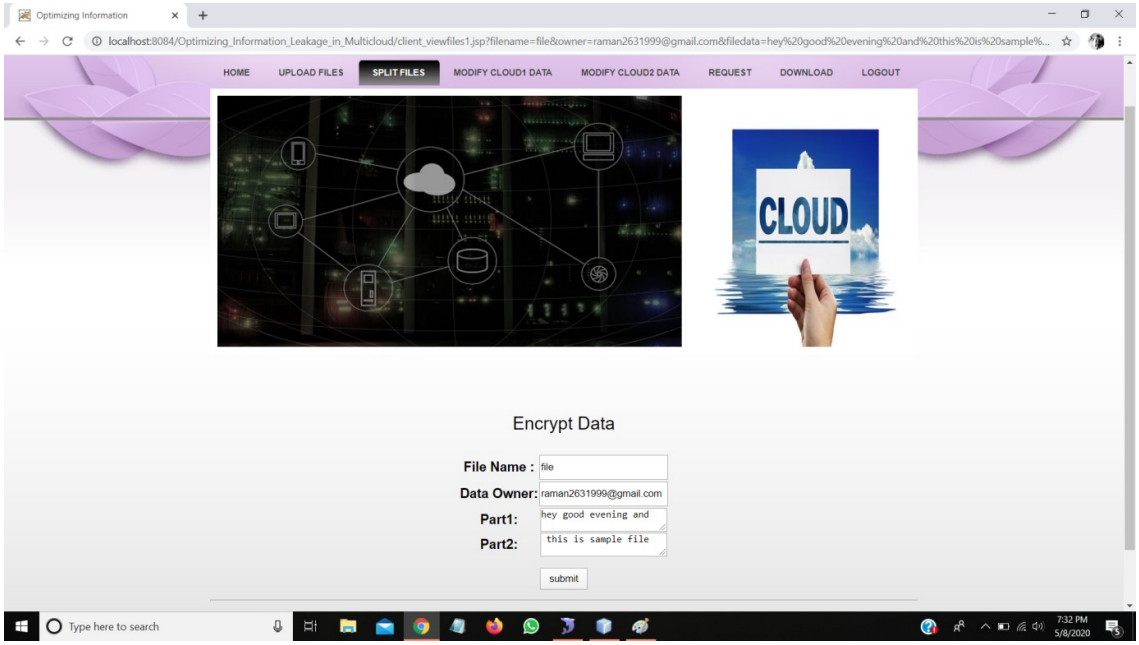


Figure 6. 7: SPLIT FILE PAGE

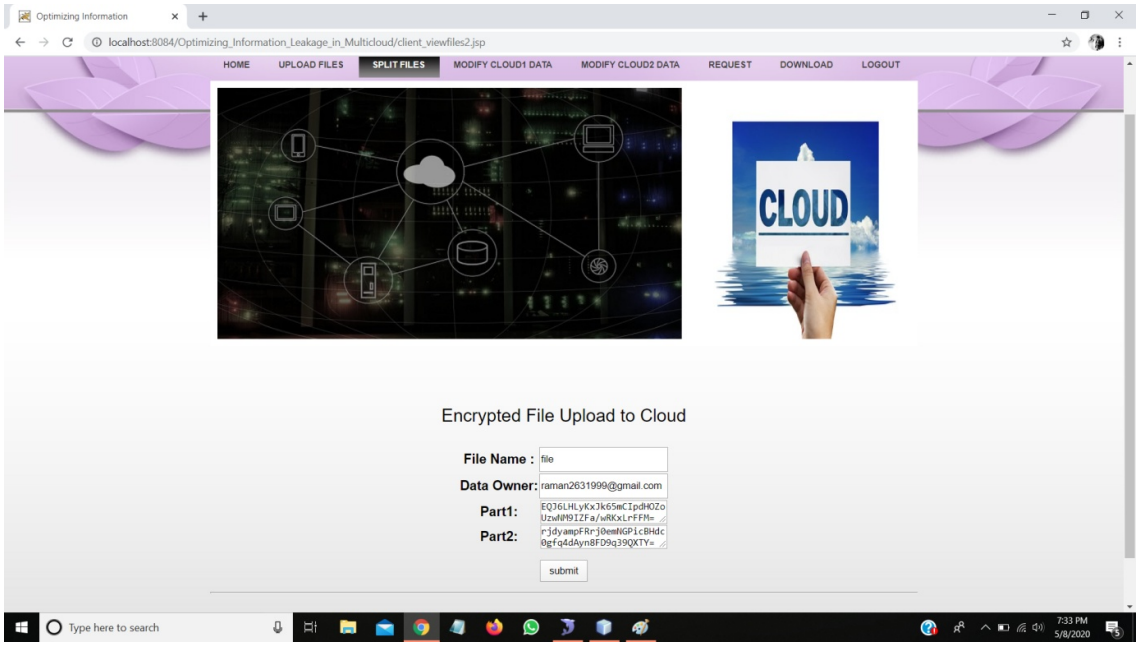


Figure 6. 8: SPLIT FILE ENCRYPTED FORMAT

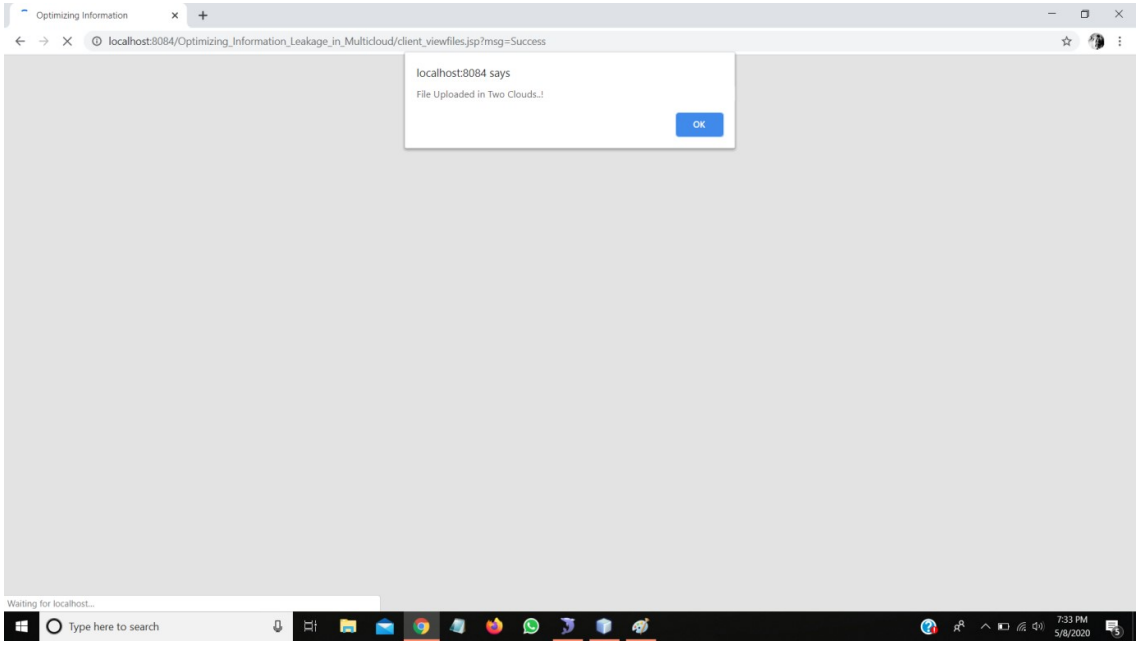


Figure 6. 9:UPLOAD SUCCESSFUL PROMPT

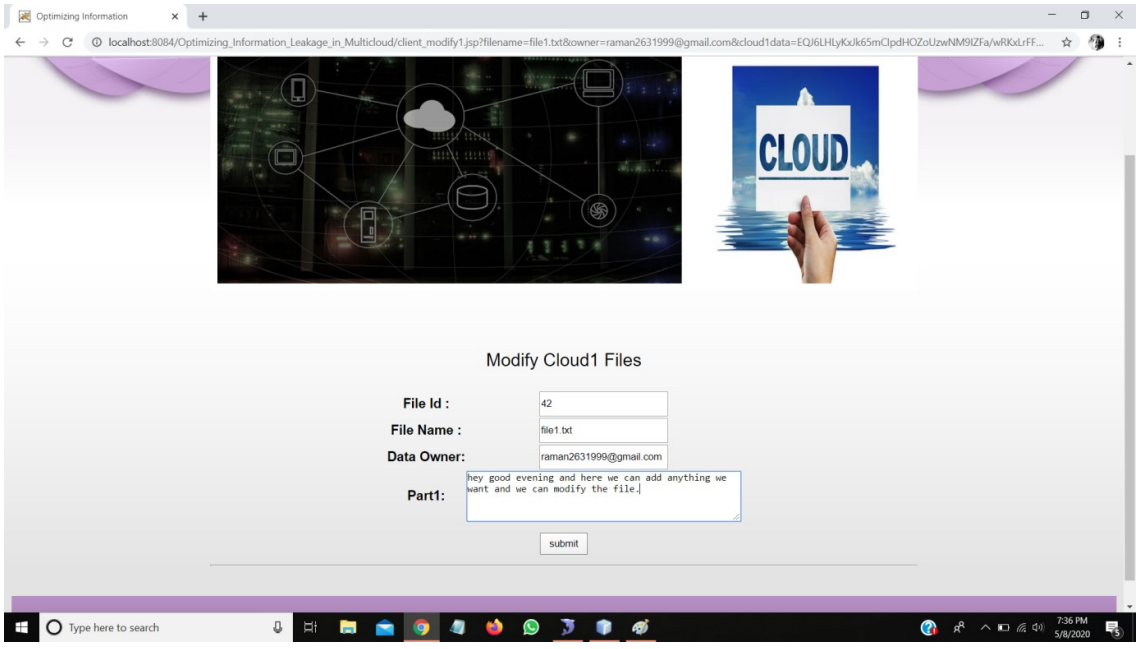


Figure 6. 10: MODIFY CLOUD 1 PAGE

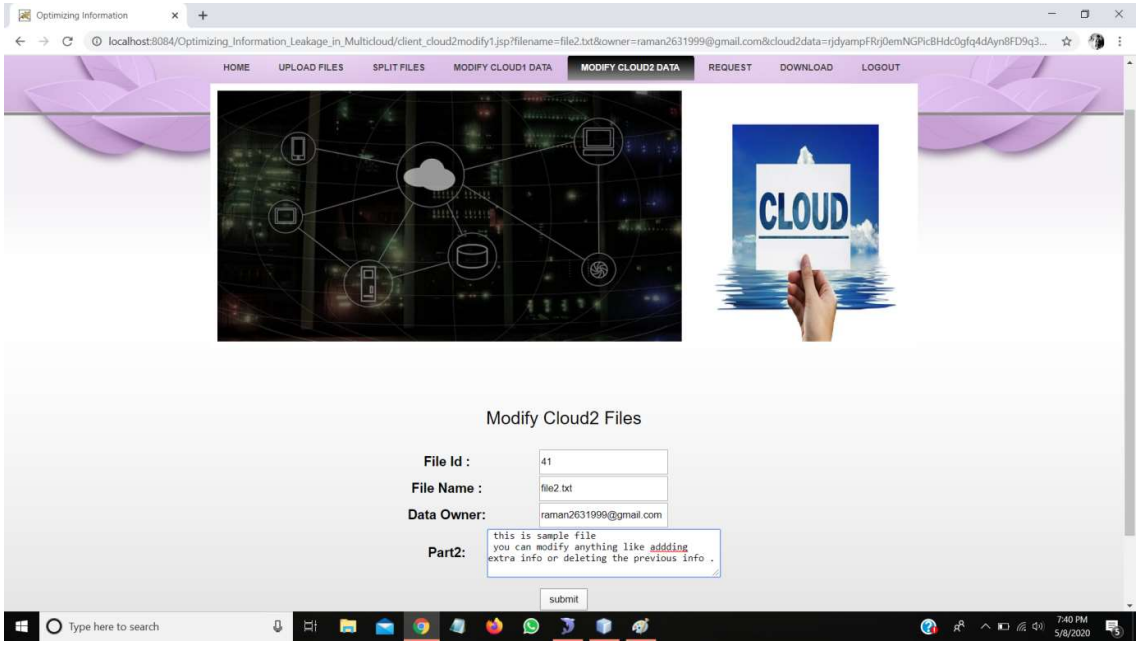


Figure 6. 11: MODIFY CLOUD 2 PAGE

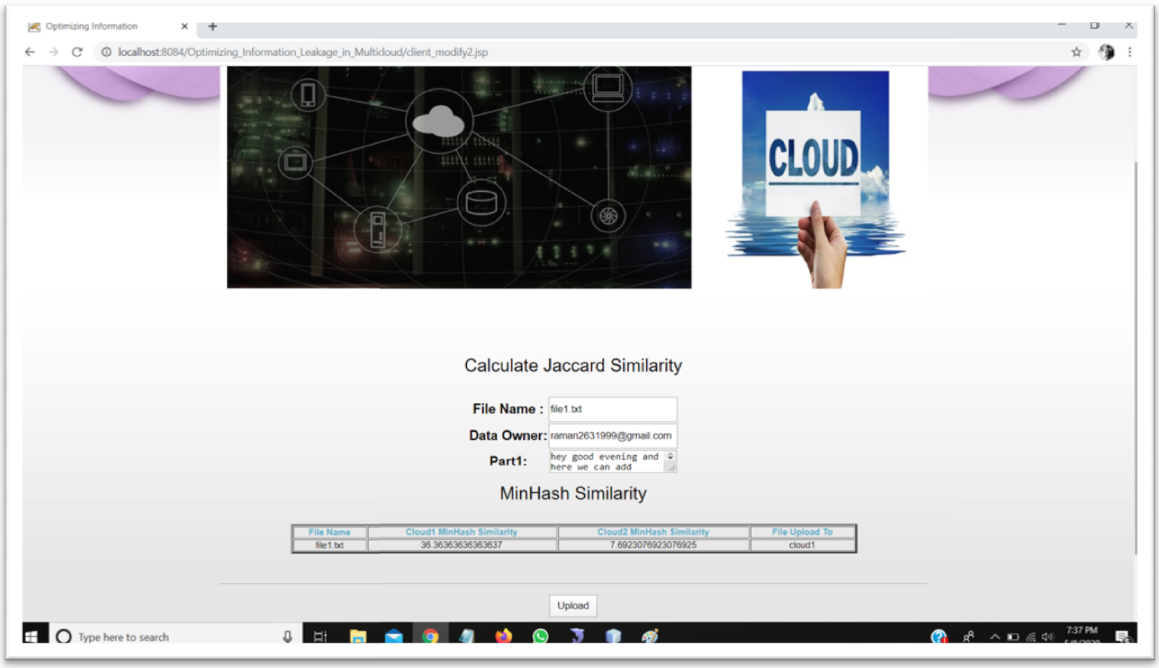


Figure 6. 12: CALCULATION OF MINHASH SIMILARITY CLOUD 1

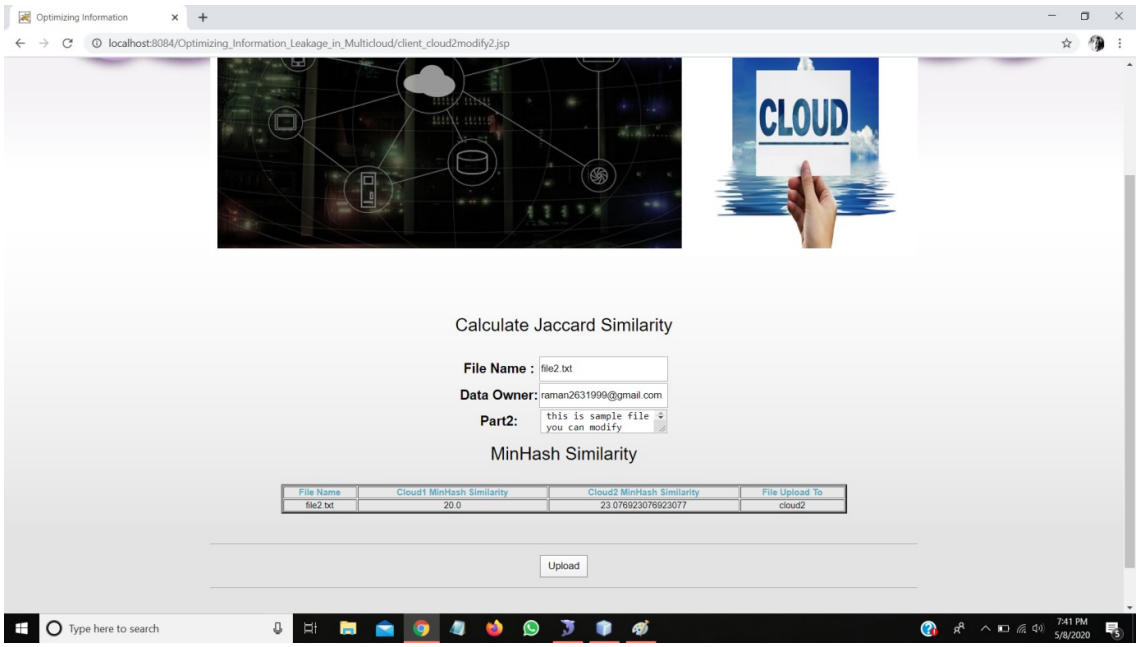


Figure 6. 13:CALCULATING MINHASH SIMILARITY CLOUD 2

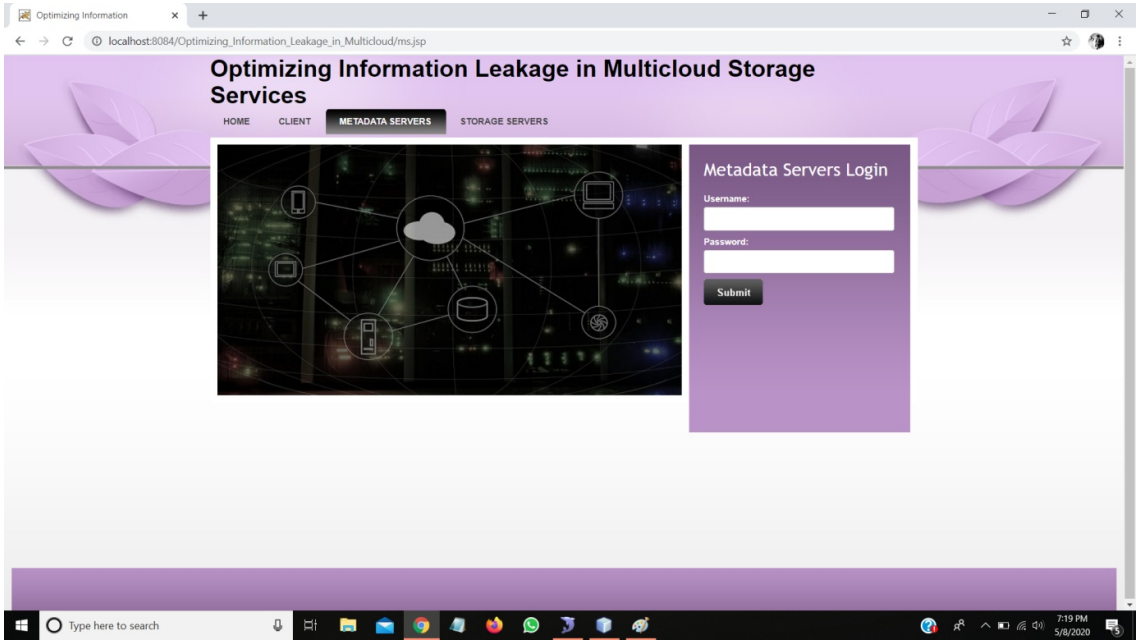


Figure 6. 14:METADATA SERVER LOGIN PAGE

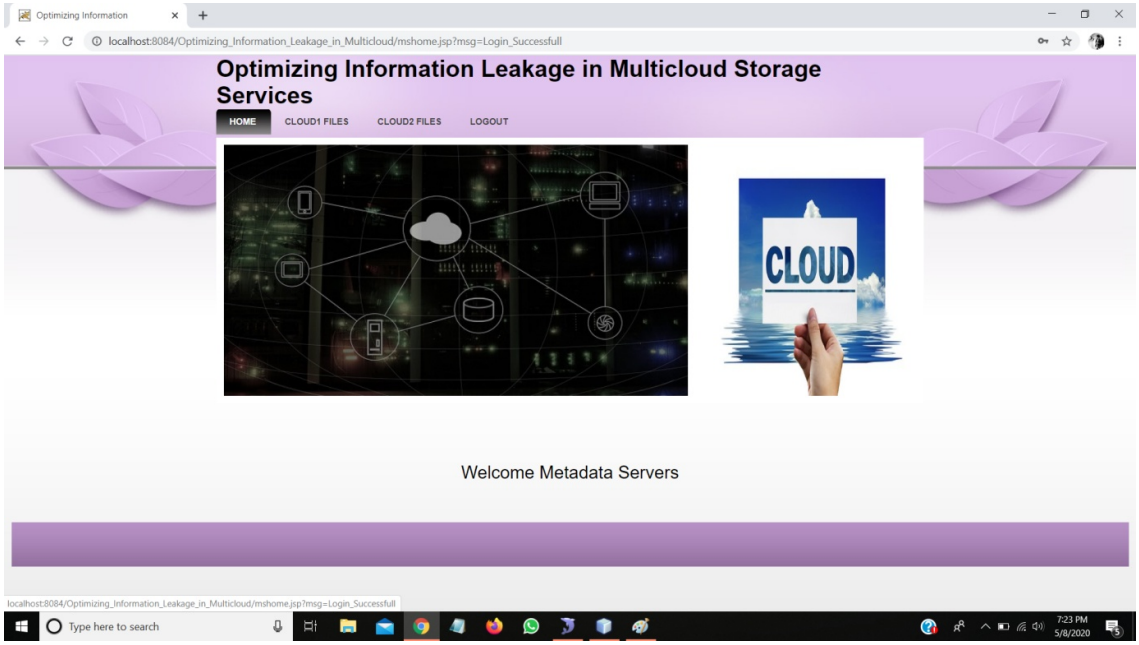


Figure 6. 15:METADATA SERVER HOMEPAGE

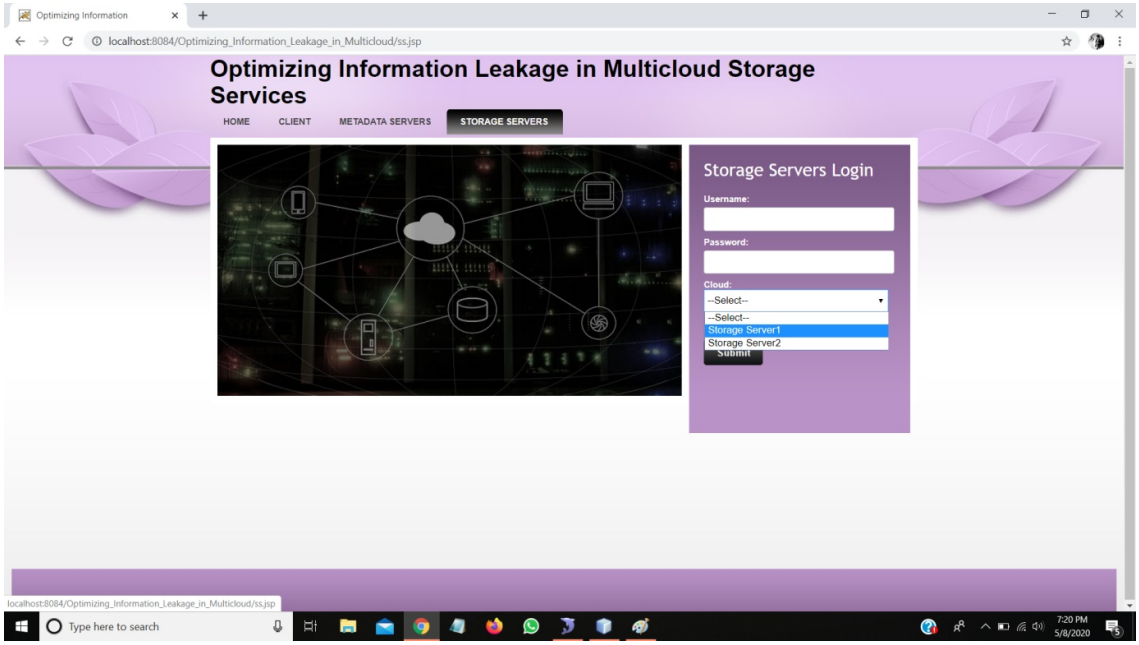


Figure 6. 16:STORAGE SERVER LOGIN PAGE

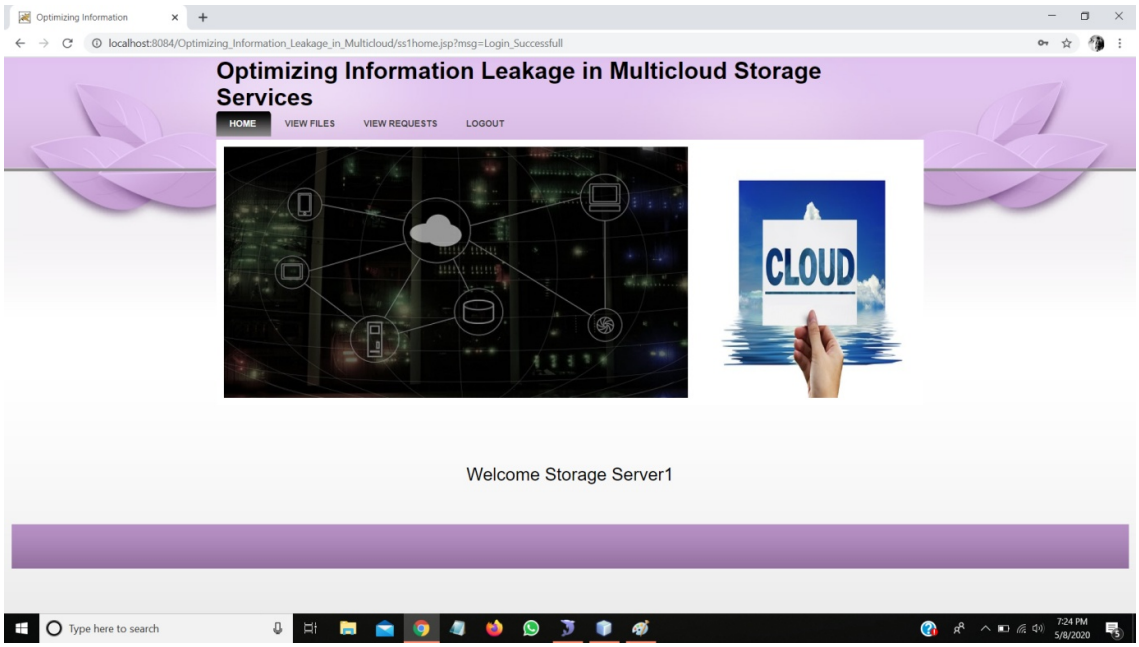


Figure 6. 17:STORAGE SERVER 1 HOME PAGE

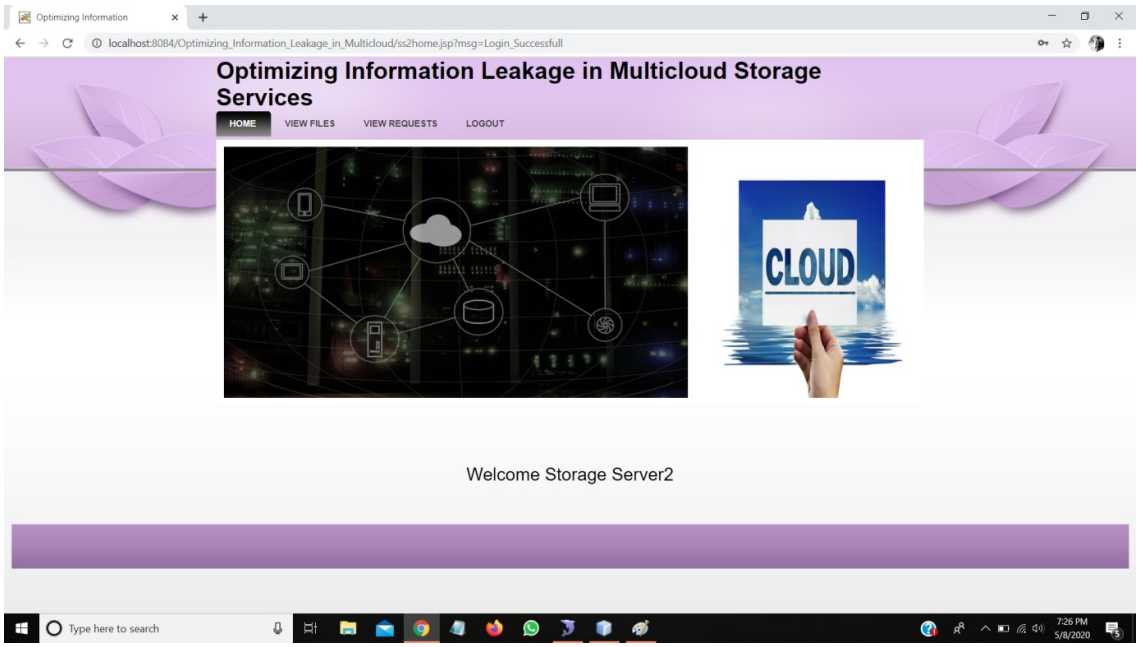


Figure 6. 18:STORAGE SERVER 2 HOMEPAGE

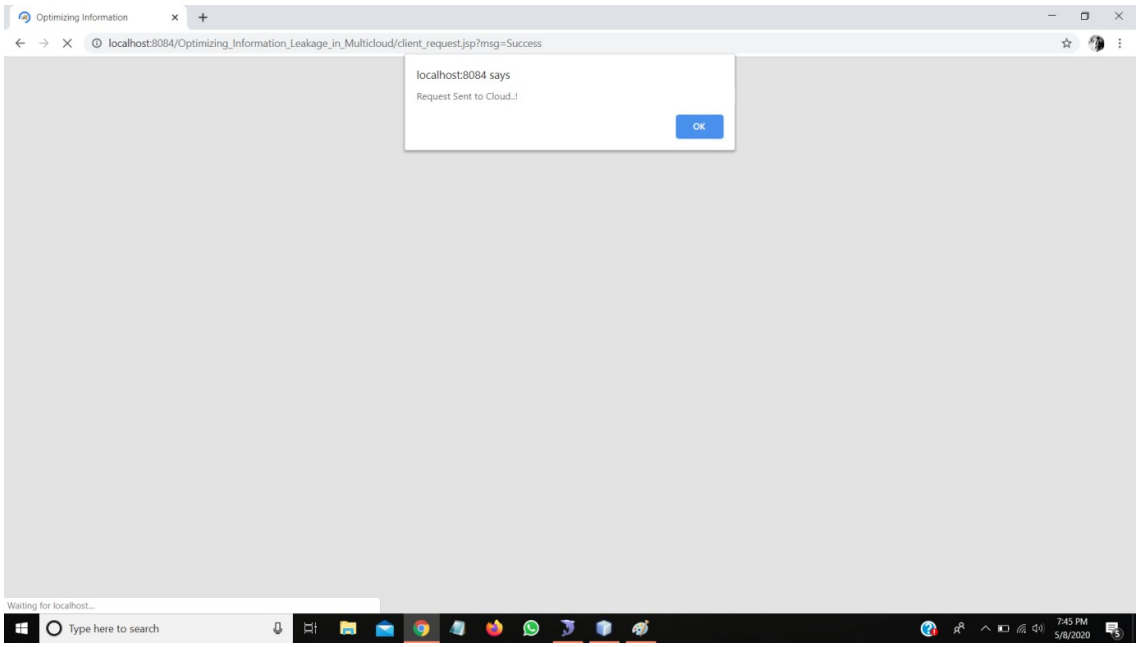


Figure 6. 19:CLIENT REQUEST SENT TO STORAGE SERVER

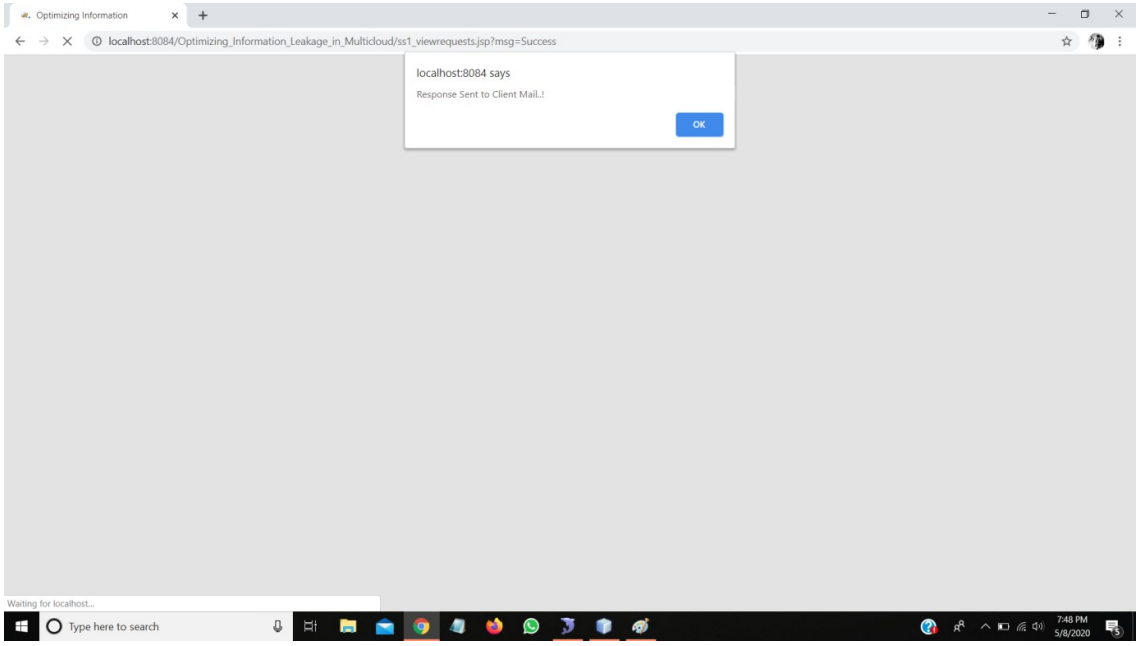


Figure 6. 20:AFTER RESPONDING TO CLIENT REQUESTS

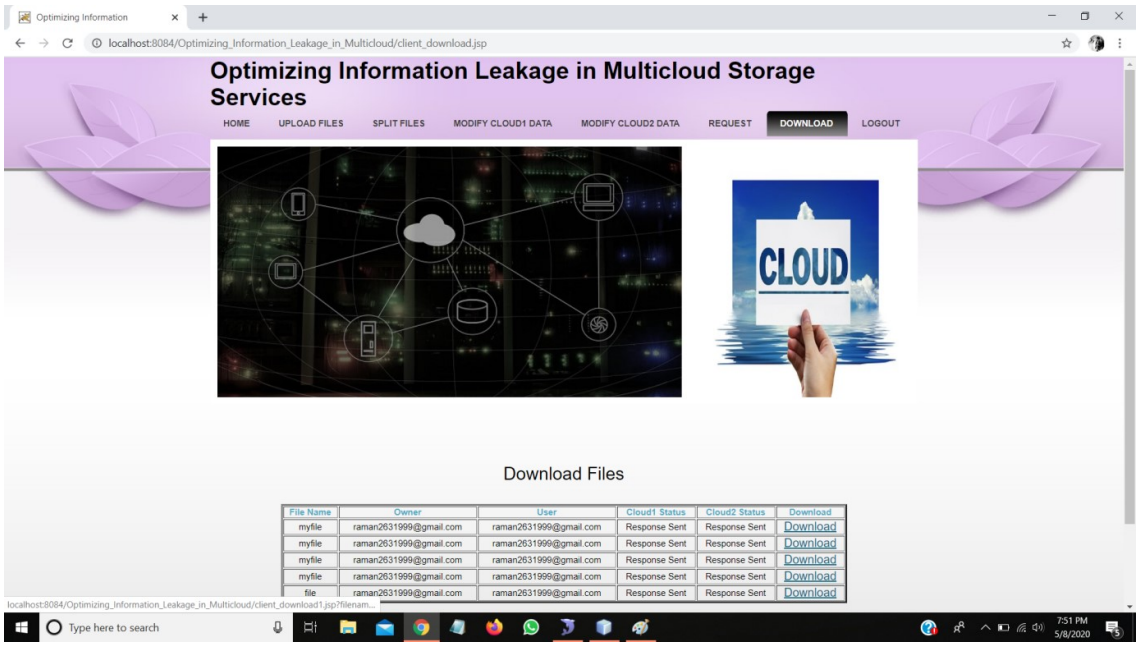


Figure 6. 21: CLIENTS DOWNLOAD PAGE

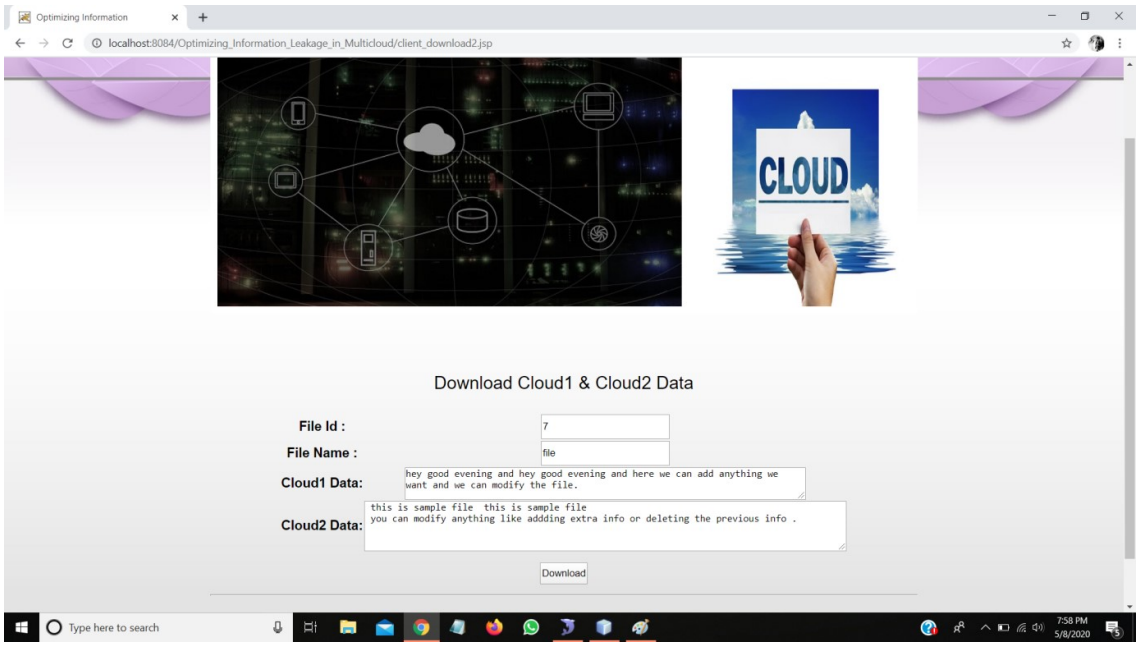


Figure 6. 22:AFTER ENTERING SUCCESSFUL SECRET KEYS

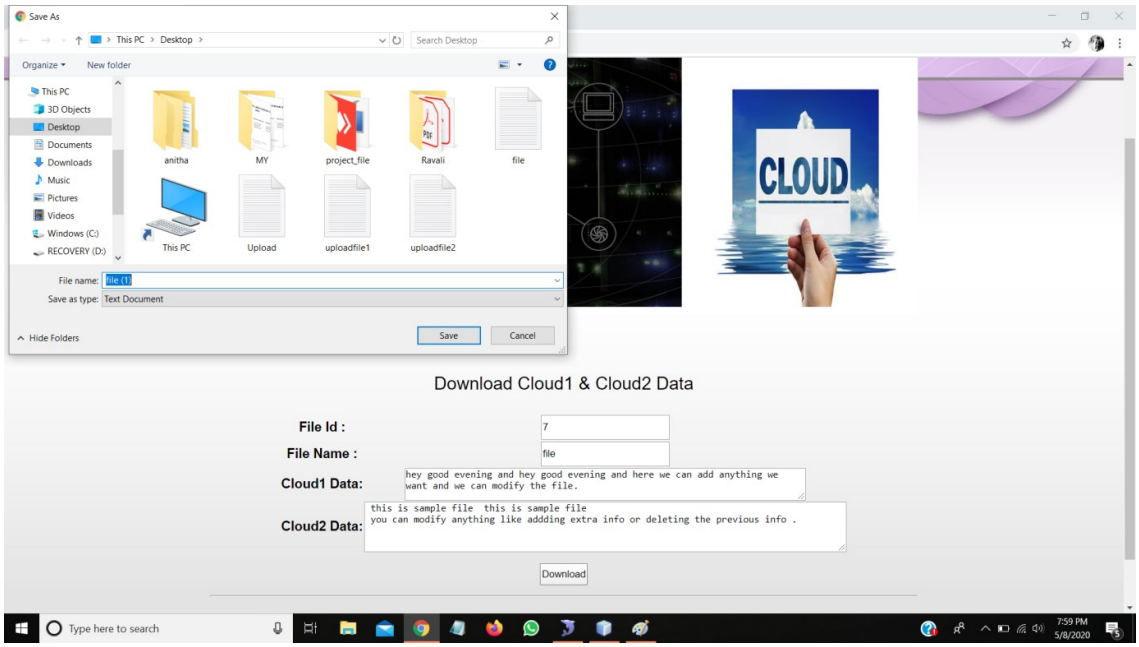


Figure 6. 23:CHOOSING DOWNLOAD PATH

CHAPTER 7

TESTING

7. TESTING

7.1 SYSTEM TESTING :

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.

7.2 TESTING METHODOLOGIES :

The following are the Testing Methodologies:

• Unit Testing.

• Integration Testing.

7.2.1 Unit Testing :

- The purpose is to validate that each unit of the software code performs as expected. Modules and methods in the project have been individually tested. Edge cases were also covered. We carried out both functional and non functional tests. For the demonstration purpose we consider the example of project module.

Examples :

1. Verify if a user will be able to login with a valid username and valid password.
2. Verify if a user cannot login with a valid username and an invalid password.
3. User enters the invalid email address & clicks the Next button, Verify if the user will get the alert message.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the

software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- **Functional testing** – This black box testing type is related to functional requirements of a system; it is done by software testers.
- **Non-functional testing** – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** – Regression testing is done after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure.It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

What do you verify in White Box Testing ?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

STEP- 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

STEP-2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

7.2.2 Integration Testing :

- Software modules have been integrated logically and tested as a group i.e modules have been tested for interoperability with other modules in the project.

- The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

Test Case Objective	**Test Case Description**	**Tested Result**
Check the file uploaded is .txt .	Checking the file given as input is .txt	Verified in the Test file.

Table 7.2.2.1: Integration Testing Table

CONCLUSION AND FUTURE SCOPE

- Distributing data on multiple clouds provides users with a certain degree of information leakage control in that no single cloud provider is privacy to all the user's data. We show that distributing data chunks in a round robin way can leak user's data as high as 80% of the total information with the increase in the number of data synchronization. To optimize the information leakage, we presented the StoreSim, an information leakage aware storage system in the multicloud.We demonstrate that StoreSim is both effective and efficient in minimizing information leakage during the process of synchronization in multicloud. We show that our StoreSim can achieve near-optimal performance and reduce information leakage up to 80% compared to unplanned placement.
- We can improve the file formats like other than text documents.Now we are doing only with text documents .
- We can make sure that information leakage should be reduced to 100% compared to unplanned placement.

REFERENCES

[1]. J. Crowcroft, "On the duality of resilience and privacy," in Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 471, no. 2175. The Royal Society, 2015, p. 20140862.

[2]. Bessani, M. Correia, B. Quaresma, F. Andr´e, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," ACM Transactions on Storage (TOS), vol. 9, no. 4, p. 12, 2013.

[3]. H. Chen, Y. Hu, P. Lee, and Y. Tang, "Nccloud: A network-coding-based storage system in a cloud-of-clouds," 2013.

[4]. T. G. Papaioannou, N. Bonvin, and K. Aberer, "Scalia: an adaptive scheme for efficient multi-cloud storage," in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press, 2012, p. 20.

[5]. Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services,"in Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 2013, pp. 292–308.

[6] G. Greenwald and E. MacAskill, "Nsa prism program taps in to user data of apple, google and others," The Guardian, vol. 7, no. 6, pp. 1–43, 2013.

[7] T. Suel and N. Memon, "Algorithms for delta compression and remote file synchronization," 2002.

[8] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking personal cloud storage," in Proceedings of the 2013 conference on Internet
measurement conference. ACM, 2013, pp. 205–212.

[9] I. Drago, M. Mellia, M.MMunafo, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: understanding personal cloud storage services," in Proceedings of the 2012 ACM conference on Internet measurement conference. ACM, 2012, pp. 481–494.

[10] U. Manber et al., "Finding similar files in a large file system." in Usenix Winter, vol. 94, 1994, pp. 1–10.

[11] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M.Walfish, "Depot: Cloud storage with minimal trust," ACM Transactions on Computer Systems (TOCS), vol. 29, no. 4, p. 12, 2011.

[12] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten, "Sporc: Group collaboration using untrusted cloud resources." in OSDI, vol. 10, 2010, pp. 337–350.

[13] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," in ACM SIGOPS Operating Systems Review, vol. 35, no. 5. ACM, 2001, pp. 202–215.

[14] L. P. Cox and B. D. Noble, “Samsara: Honor among thieves in peer-to-peer storage,” ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 120–132, 2003.

[15] H. Zhuang, R. Rahman, and K. Aberer, “Decentralizing the cloud: How can small data centers cooperate?” in Peer-to-Peer Computing (P2P), 14-th IEEE International Conference on. Ieee, 2014, pp. 1–10.

[16] H. Zhuang, R. Rahman, P. Hui, and K. Aberer, “Storesim: Optimizing information leakage in multicloud storage services,” in Cloud Computing
Technology and Science (CloudCom), 2015 IEEE 7th International Conference on. IEEE, 2015, pp. 379–386.

[17] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “A hybrid edge-cloud architecture for reducing on-demand gaming latency,” Multimedia Systems,
pp. 1–17, 2014.

[18] T. Zou, R. Le Bras, M. V. Salles, A. Demers, and J. Gehrke, “Cloudia: a deployment advisor for public clouds,” in Proceedings of the VLDB Endowment, vol. 6, no. 2. VLDB Endowment, 2012, pp. 121–132.

[19] J.-M. Bohli, N. Gruschka, M. Jensen, L. L. Iacono, and N. Marnau, “Security and privacy-enhancing multicloud architectures,” Dependable and Secure Computing, IEEE Transactions on, vol. 10, no. 4, pp. 212–224, 2013.

[20] H. Harkous, R. Rahman, and K. Aberer, “C3p: Context-aware crowdsourced cloud privacy,” in 14th Privacy Enhancing Technologies Symposium (PETS 2014), 2014.

Appendix A : SAMPLE CODE

Databaseconnection

```
package databaseconnection;
import java.sql.*;

public class databasecon
{
	static Connection co;
	public static Connection getconnection()
	{


		try
		{
			Class.forName("com.mysql.jdbc.Driver");
			co =
DriverManager.getConnection("jdbc:mysql://localhost:3306/securedata","root","root");
		}
		catch(Exception e)
		{
			System.out.println("Database Error"+e);
		}
		return co;
	}

}
```

Decryption

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package action;

/**
 *
 * @author java2
 */
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JOptionPane;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class decryption {
//public static void main(String args[])
//{
//          Scanner s=new Scanner(System.in);
//           System.out.println("Enter encrypted Text and key");
//           String text=s.next();
//           String key=s.next();
//          new decryption().decrypt(text,key);
//}

   public String decrypt(String txt, String skey) {
      String decryptedtext = null;
      try {

         //converting string to secretkey
         byte[] bs = Base64.decode(skey);
         SecretKey sec = new SecretKeySpec(bs, "AES");
         System.out.println("converted string to seretkey:" + sec);

         System.out.println("secret key:" + sec);

         Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
         aesCipher.init(Cipher.ENCRYPT_MODE, sec);//initiating ciper encryption using secretkey

         byte[] byteCipherText = new BASE64Decoder().decodeBuffer(txt); //encrypting data

         //  System.out.println("ciper text:"+byteCipherText);
```

```
            aesCipher.init(Cipher.DECRYPT_MODE, sec, aesCipher.getParameters());//initiating ciper
decryption

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

            System.out.println("Decrypted Text:" + decryptedtext);
        } catch (Exception e) {
            System.out.println(e);
        }
        return decryptedtext;
    }

    String decrypt(String str, SecretKey sec) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```

Download

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package action;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author java2
 */
public class download extends HttpServlet {
String part1_name="";

   /**
    * 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, SQLException {
       response.setContentType("text/html;charset=UTF-8");
       try (PrintWriter out = response.getWriter()) {
           /* TODO output your page here. You may use following sample code. */
           String[] filedetails = request.getQueryString().split(",");
           String filename = null, dkey = null;

           InputStream is = null;
           FileInputStream fis = null;
           Connection con = Dbconnection.getConnection();
           Statement st = con.createStatement();
           ResultSet rt = st.executeQuery("select * from request where filename='" + filedetails[0] + "'
AND  owner='" + filedetails[1] + "'");
           if (rt.next()) {
               filename = rt.getString("filename");
               dkey = rt.getString("dkey1");
               //is = (InputStream) rt.getAsciiStream("data");
           } else {
               out.println("error while retreiving data");
           }

          setName(filename+".part0");
             InputStream in = new
FileInputStream("C:\\Users\\admin\\Desktop\\down\\"+filename+".part0");
           BufferedReader br = new BufferedReader(new InputStreamReader(in));
           String temp = null;
           StringBuffer sb = new StringBuffer();
           while ((temp = br.readLine()) != null) {
               sb.append(temp + "\n");
           }
           //String content = new decryption().decrypt(sb.toString(), skey);
           response.setHeader("Content-Disposition", "attachment;filename=\"" + filename + "\"");
```

```
            out.write(sb.toString());
        }
    }

    public void setName(String Part_name){

      part1_name = Part_name;

    }// <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 {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(download.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * 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 {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(download.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
```

```
    public String getName(){

     return part1_name;
  }
}
```

encrypt1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package action;

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.BufferedReader;
import java.io.File;
import action.Ftpcon;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.Iterator;
import java.util.List;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
```

```
/**
 *
 * @author java2
 */
public class encrypt1 extends HttpServlet {

    //private static java.sql.Date getCurrentDate() //{
        //java.util.Date today = new java.util.Date();
        //return new java.sql.Date(today.getTime());
    //}

    /**
     * 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");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            Connection con;
            PreparedStatement pstm = null;
            PreparedStatement pstm1 = null;
             PreparedStatement pstm2 = null;
             String filename = "";
             String owner = request.getParameter("owner");

            String fname1 = "";
             //String date = "";
             //String gname = "";
             //String subject = "";
            // String keyword = "";
            String cd = "";
            String str = "";



             String privatekey = "";
             String policy = "";
             String role = "";
             String exp = "";
            String department = "";
            String skey="8962"  ;
         String securitylevel="";
        byte PART_SIZE = 2;

            try {
                boolean isMultipartContent = ServletFileUpload.isMultipartContent(request);
                if (!isMultipartContent) {
```

```
    return;
}
FileItemFactory factory = new DiskFileItemFactory();
ServletFileUpload upload = new ServletFileUpload(factory);
try {
    List<FileItem> fields = upload.parseRequest(request);
    Iterator<FileItem> it = fields.iterator();
    if (!it.hasNext()) {
        return;
    }
    while (it.hasNext()) {
        FileItem fileItem = it.next();

        if (fileItem.getFieldName().equals("privatekey")) {
            privatekey = fileItem.getString();
            System.out.println("File Name" + privatekey);

        }

            if (fileItem.getFieldName().equals("policy")) {
            policy = fileItem.getString();
            System.out.println("File Name" + policy);

        }

              if (fileItem.getFieldName().equals("role")) {
            role = fileItem.getString();
            System.out.println("File Name" + role);

        }

                if (fileItem.getFieldName().equals("exp")) {
            exp = fileItem.getString();
            System.out.println("File Name" + exp);

        }

                    if (fileItem.getFieldName().equals("department")) {
            department = fileItem.getString();
            System.out.println("File Name" + department);

        }


                        if (fileItem.getFieldName().equals("securitylevel")) {
            securitylevel = fileItem.getString();
            System.out.println("File Name" + securitylevel);

        }


        if (fileItem.getFieldName().equals("filename")) {
            filename = fileItem.getString();
            System.out.println("File Name" + filename);
```

```
                }
                if (fileItem.getFieldName().equals("owner")) {
                    owner = fileItem.getString();
                    System.out.println("File Keyword" + owner);

                }
                if(fileItem.getFieldName().equals("modify")) {
                    str = getStringFromInputStream(fileItem.getInputStream());
                    System.out.println("datazzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz" + str);
                }

            }

                try {
                    con = Dbconnection.getConnection();
                    pstm = con.prepareStatement("insert into upload (filename, data,
owner,privatekey,skey)values(?,?,?,?,?)");
                  pstm1 = con.prepareStatement("insert into cloudadata (filename,newfilename,
owner,privatekey,dkey)values(?,?,?,?,?)");
                    pstm2 = con.prepareStatement("insert into cloudbdata (filename,newfilename,
owner,privatekey,dkey)values(?,?,?,?,?)");
                    // String str = getStringFromInputStream(fileItem.getInputStream());
                     //secretkey generating
                //  KeyGenerator keyGen = KeyGenerator.getInstance("AES");
                     //  keyGen.init(128);
                    //  SecretKey secretKey = keyGen.generateKey();
                    //   System.out.println("secret key:" + secretKey);
                     //converting secretkey to String
                      //byte[] be = secretKey.getEncoded();//encoding secretkey
                    //  String skey = Base64.encode(be);
                    //   System.out.println("converted secretkey to string:" + skey);

                      byte[] bs = Base64.decode(privatekey);
                     SecretKey sec = new SecretKeySpec(bs, "AES");

                      String cipher = new StringXORer().encode(str,privatekey);

                    // String cipher = new encryption().encrypt(str, sec);
                      System.out.println("aaaaaaaaaaaaaaaaaaaaa"+str);

                      //for get extension from given file
                      //String b = fileItem.getName().substring(fileItem.getName().lastIndexOf('.'));
                      //System.out.println("File Extension" + b);
                      String skey1="7895";
                    pstm.setString(1, filename);
                     // pstm.setString(4, date);
                     // pstm.setDate(4, getCurrentDate());

                      pstm.setString(2, cipher);
                     // pstm.setString(6, name);
                       pstm.setString(3, owner);
                        pstm.setString(4, privatekey);
                         pstm.setString(5, skey1);

                      /*Cloud Start*/
```

```
  File f = new File("C:\\Users\\admin\\Desktop\\upoladfiles\\"+filename);
 FileWriter fw = new FileWriter(f);
fw.write(cipher);
    fw.close();

    Ftpcon ftpcon = new Ftpcon();
    File inputFile = new File("C:\\Users\\admin\\Desktop\\upoladfiles\\"+filename);
    FileInputStream inputStream;
    Random r = new Random();
  String newFileName;
  FileOutputStream filePart;
  int fileSize = (int) inputFile.length();
  int nChunks = 0, read = 0, readLength = fileSize/2;
  byte[] byteChunkPart;
    inputStream = new FileInputStream(inputFile);
          while (fileSize > 0) {
                if (fileSize <= 2) {
                      readLength = fileSize;
                }
                byteChunkPart = new byte[readLength];
                read = inputStream.read(byteChunkPart, 0, readLength);
                fileSize -= read;
                assert (read == byteChunkPart.length);
                nChunks++;

                newFileName = filename + ".part"
                              + Integer.toString(nChunks - 1);

       File inputFile1 = new File("C:\\Users\\admin\\Desktop\\down\\"+newFileName);
                filePart = new FileOutputStream(inputFile1);
                filePart.write(byteChunkPart);
                filePart.flush();
                filePart.close();
                byteChunkPart = null;
                filePart = null;

      if (nChunks==1){

         int i = r.nextInt(10000 - 5000) + 5000;
         String dkey = i+"";
          pstm1.setString(1, filename);
          pstm1.setString(2, newFileName);
          pstm1.setString(3, owner);
          pstm1.setString(4, privatekey);
          pstm1.setString(5, dkey);
          pstm1.execute();
      ftpcon.upload(inputFile1, newFileName);
      }
      else if(nChunks==2){

         int i = r.nextInt(10000 - 4000) + 5000;
         String dkey = i+"";
          pstm2.setString(1, filename);
          pstm2.setString(2, newFileName);
          pstm2.setString(3, owner);
```

```
                    pstm2.setString(4, privatekey);
                    pstm2.setString(5, dkey);
                    pstm2.execute();
                 ftpcon.upload1(inputFile1, newFileName);
                }
                    }
                    inputStream.close();


                int i = pstm.executeUpdate();
                if (i == 1) {
                   response.sendRedirect("upload.jsp?msg=success");
                } else {
                   response.sendRedirect("upload.jsp?m1=failed");
                }
                con.close();
              } catch (Exception e) {
                out.println(e.toString());
              }


          } catch (FileUploadException e) {
          } catch (Exception ex) {
             Logger.getLogger(encrypt1.class.getName()).log(Level.SEVERE, null, ex);
          }
       } finally {
          out.close();
       }
    }


}

private static String getStringFromInputStream(InputStream is) {
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();
    String line;
    try {
       br = new BufferedReader(new InputStreamReader(is));
       while ((line = br.readLine()) != null) {
          sb.append(line + "\n");
       }
    } catch (IOException e) {
    } finally {
       if (br != null) {
          try {
             br.close();
          } catch (IOException e) {
          }
       }
    }
    return sb.toString();
}
```

```
    // <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>

}
```

Encryption

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package action;
```

```
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JOptionPane;
import sun.misc.BASE64Encoder;

public class encryption {

   public String encrypt(String text, SecretKey secretkey) {
      String plainData = null, cipherText = null;
      try {
         plainData = text;

         Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
         aesCipher.init(Cipher.ENCRYPT_MODE, secretkey);//initiating ciper encryption using
secretkey

         byte[] byteDataToEncrypt = plainData.getBytes();
         byte[] byteCipherText = aesCipher.doFinal(byteDataToEncrypt);//encrypting data

         cipherText = new BASE64Encoder().encode(byteCipherText);//converting encrypted data to
string

         System.out.println("\n Given text : " + plainData + " \n Cipher Data : " + cipherText);

      } catch (Exception e) {
         System.out.println(e);
      }
      return cipherText;
   }

}
```

Ftpcon

```
package action;

import java.io.File;
import java.io.FileInputStream;
```

```
import org.apache.commons.net.ftp.FTPClient;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author java2
 */
public class Ftpcon {

    FTPClient client = new FTPClient();
    FileInputStream fis = null;
    boolean status;

    public boolean upload(File file,String filename) {
        try {
            client.connect("ftp.drivehq.com");
            client.login("1000p", "1000p");
            client.enterLocalPassiveMode();
            fis = new FileInputStream(file);
            status = client.storeFile("/shiva/" + filename, fis);
            client.logout();
            fis.close();

        } catch (Exception e) {
            System.out.println(e);
        }


        if (status) {
            System.out.println("success");
            return true;
        } else {
            System.out.println("failed");
            return false;
        }
    }
    public boolean upload1(File file,String filename) {
         try {


            client.connect("ftp.drivehq.com");
            client.login("Rajakishan", "Raju123");
            client.enterLocalPassiveMode();
            fis = new FileInputStream(file);
            status = client.storeFile("/c/" + filename, fis);
            client.logout();
            fis.close();

        } catch (Exception e) {
            System.out.println(e);
        }
```

```
    if (status) {
          System.out.println("success");
          return true;
       } else {
          System.out.println("failed");
          return false;
       }
    }


}
```

Mail

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package action;

import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

      import javax.mail.Authenticator;
/**
 *
 * @author java2
 */
public class Mail {

   public static boolean secretMail(String msg, String userid, String to) {
      Properties props = new Properties();
      props.put("mail.smtp.host", "smtp.gmail.com");
      props.put("mail.smtp.socketFactory.port", "465");
      props.put("mail.smtp.socketFactory.class",
            "javax.net.ssl.SSLSocketFactory");
      props.put("mail.smtp.auth", "true");
      props.put("mail.smtp.port", "465");
      // Assuming you are sending email from localhost
```

```
        Session session = Session.getDefaultInstance(props,
                new javax.mail.Authenticator() {
                    @Override
                    protected PasswordAuthentication getPasswordAuthentication() {
                        return new PasswordAuthentication("nikilp306@gmail.com", "9032101994");
                    }
                });

        System.out.println("Message   " + msg);
        try {
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(userid));
            message.setRecipients(Message.RecipientType.TO,
                    InternetAddress.parse(to));
            message.setSubject("Secret key ");
            message.setText(msg);

            Transport.send(message);

            System.out.println("Done");
            return true;

        } catch (MessagingException e) {
            System.out.println(e);
            e.printStackTrace();
            return false;
            // throw new RuntimeException(e);
        }
    }

}
```