ABSTRACT

Cross-cloud data migration is one of the prevailing challenges faced by mobile users, which is an essential process when users change their mobile phones to a different provider. However, due to the insufficient local storage and computational capabilities of the smart phones, it is often very difficult for users to backup all data from the original cloud servers to their mobile phones in order to further upload the downloaded data to the new cloud provider. To solve this problem, an efficient data migration model between cloud providers and a mutual authentication and key agreement scheme is to be constructed based on elliptic curve certificate-free cryptography for peer-to peer cloud. The proposed scheme helps to develop trust between different cloud providers and lays a foundation for the realization of cross-cloud data migration. Mathematical verification and security correctness of the scheme is evaluated against notable existing schemes of data migration, which demonstrate that the proposed scheme exhibits a better performance than other state of- the-art scheme in terms of the achieved reduction in both the computational and communication cost.

1.INTRODUCTION

1.1 PURPOSE

With the rapid development of the smart phone and mobile terminal industries, smart phones have become indispensable for people. People are now increasingly relying on hand-held devices such as smart phones, tablet etc., in an unprecedented number. It is worthy of note that one individual may own and use multiple smart devices. It is also common for people to recycle their smart devices quite frequently, given the fact that new arrivals characterize more attractive inherent features from a variety of manufacturers .When people opt to use a new smart device from a different manufacturer, the data stored in the cloud server of the previous smart device provider should be transferred to the cloud server of the new smart device provider. One of the common ways of accomplishing this transfer is to log onto the original cloud server, download the data onto the smart terminal devices, log onto the new cloud server, and finally upload the data to the new server. To this end, it is essential to develop a more efficient and secure way of data transfer from one cloud server to another.

1.2 SCOPE

The data migration issues between clouds has many unresolved potential problems. For instance, achieving mutual authentication, building communication key securely and protecting the data transfer from potential attacks are some concerns to mention. Herein, authentication and key agreement mechanism can be an effective way to solve these problems.

1.3 MODEL DIAGRAM

It is essential to develop a more efficient and secure way of data transfer from one cloud server to another. An ideal data migration model that can transfer user data directly between cloud servers is shown in Fig. 1.1.Such a model often imposes compatibility issues, since different cloud service providers characterize diverse user functions, mutual distrust and security risks in the process of data transmission, which make this ideal data migration model difficult to implement.

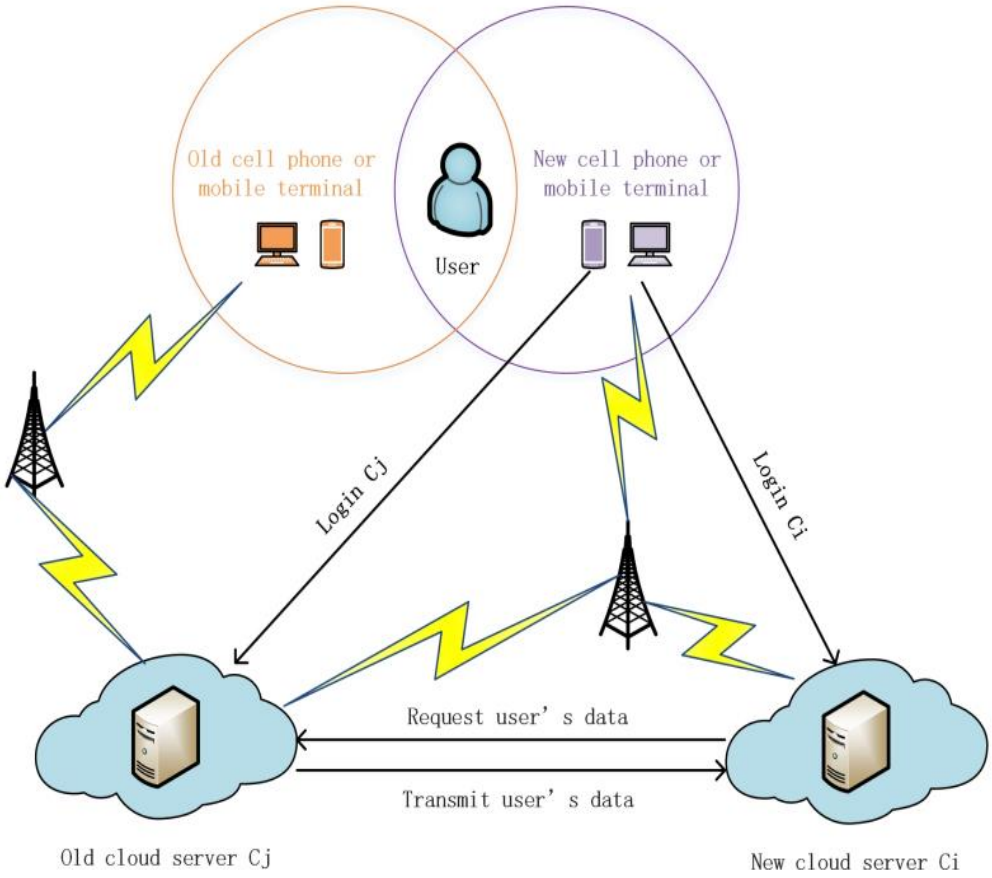

Fig 1.1 Data Migration Model

2.LITERATURE SURVEY

The related work on anonymous authentication systems is widely divided into publicly-constructed cryptosystems (PKCs), identity-based cryptosystems, pseudonym-based systems, combination systems with identity-based encryption and pseudonyms, and application-oriented methods. Anonymous PKC-based authentication systems were not practical on mobile networks because of the computer resources needed for PKC modular exponentiation, which requires more resources than the mobile device can give. The performance of ECC based systems is boosted by ECC-based identity cryptosystems. Unlike the typical PKC, identity based cryptosystems use a public identification like an ID or email address as the public key to the expense elimination of public key certificate management, which is often desirable in mobile situations. However, there are two fundamental disadvantages to the schemes based on identity cryptosystems.

First, identity-based authentication systems are affected by a key problem of escrow, where a key needed to encrypt/decode is kept in a cache to ensure that an authorised person can access the key under particular circumstances. The key escrow systems are considered a safety risk, since the escrow agent holding all the cryptographic keys can be a single point of failure or information leaked. Secondly, it is very difficult to revoke the user once the private key of a user is compromised. Anonymous authentication schemes also emerge by using pseudonyms and other privacy protection mechanisms. Based on the anonymous credentials, pseudorandom number creation and proof knowledge of peer to peer systems, Lin et al. and Sun et al. presented solutions for privacy and emergency response. However, if these approaches are employed directly on the distributed healthcare system, because of the large overhead computing costs. In Lin et al. suggested a strong data protection method (SAGE), using pseudonyms and identity-based encryption to boost security levels. This system provides both contents and context oriented privacy to fight strong global opponents. Two authentication techniques (password and smartcard) have been proposed in recent years. However, there are constraints because the smart card and password can both be lost, stolen or cloned. In addition, several of these systems need the server to keep a verification password table that could cause attacks such as password disclosure attacks and server Spoofing. Authentication systems

have been presented to address the difficulties in two factor-based authentication schemes, three factor (biometric, password and smartcard). Biometric features are a third aspect used to construct a robust authentication system to improve security. These systems are developed for intelligent cards. However, clever cards cannot carry out more advanced activities such as server connection or server communication, thus these schemes are not ideal for authentication with smartphones.

2.1 TECHNOLOGIES USED

- **Elliptic Curve Cryptography**

Elliptic Curve Cryptography (ECC) The Elliptical Curve Cryptography (ECC) was introduced by Victor S. Miller and Neal Koblitz in the 80s, but it was only in the late 90's that it began its application. ECC is based on the usage of finite field elliptic curves. The equation defining the E(Fp) elliptical curve is as follows:

$$\mathbf{E: y^2 = x^3 + a.x + b (mod\ p)}$$

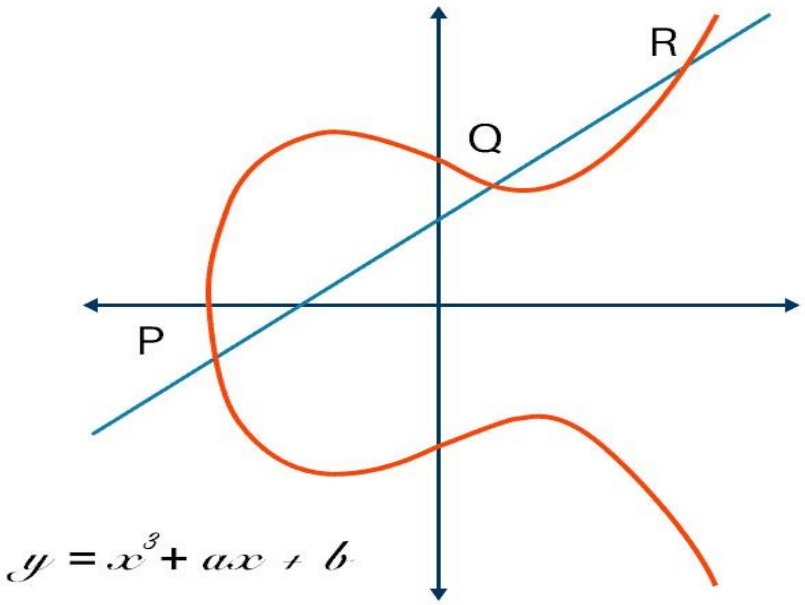

Fig 2.1 Elliptic Curve

Where p is the order of Fp and a, b − as Fp is the order of two elements. If certain relationships are followed by the prior equation, the previous equation describes a number of points in Fp with coordinates (x, y). Moreover, a sum operation is defined over an elliptical curve and a special item O is also defined as the identity element. Then, a point G of the elliptical curve is defined as a base point and an elliptical curve is employed for the sum operation of a number of elements (points). The generator order n is the integer n = O and G 6 = O. Furthermore, G + O = G which is stated to form a cyclic group across the elliptical curve. Elliptical curve cryptosystems are predicated on the intractability of certain mathematical issues, just as any other public key system. Specifically, ECC is based on the ECDLP problem that asserts that it is inoperative to compute the discrete logarithm of a random elliptical curve in relation to a base point of an elliptical curve.

Elliptic curve cryptography is a key-based technique for encrypting data. ECC focuses on pairs of public and private keys for decryption and encryption of web traffic. ECC, an alternative technique to RSA, is a powerful cryptography approach. It generates security between key pairs for public key encryption by using the mathematics of elliptic curves. ECC has gradually been growing in popularity recently due to its smaller key size and ability to maintain security. This trend will probably continue as the demand on devices to remain secure increases due to the size of keys growing, drawing on scarce mobile resources. This is why it is so important to understand elliptic curve cryptography in context.

ECC bases its approach to public key cryptographic systems on how elliptic curves are structured algebraically over finite fields. Therefore, ECC creates keys that are more difficult, mathematically, to crack. For this reason, ECC is considered to be the next generation implementation of public key cryptography and more secure than RSA.It also makes sense to adopt ECC to maintain high levels of both performance and security. That’s because ECC is increasingly in wider use as websites strive for greater online security in customer data and greater mobile optimization, simultaneously.

More sites using ECC to secure data means a greater need for this kind of quick guide to elliptic curve cryptography.

ECC is among the most commonly used implementation techniques for digital signatures in cryptocurrencies. Both Bitcoin and Ethereum apply the Elliptic Curve Digital Signature Algorithm (ECDSA) specifically in signing transactions. However, ECC is not used only in cryptocurrencies. It is a standard for encryption that will be used by most web applications going forward due to its shorter key length and efficiency.

Elliptic curves are applicable for encryption, digital signatures, pseudo-random generators and other tasks. They are also used in several integer factorization algorithms that have applications in cryptography, such as Lenstra elliptic-curve factorization.

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- A property-based proxy re-encryption scheme to enable users to achieve authorization in access control environments. So next introduced a new proxy broadcast repeat encryption (PBRE) scheme and proved its security against selective cipher text attack (CCA) under the decision n-BDHE hypothesis.
- A broadcast agent encryption (RIBBPRE) security concept based on revocable identity to solve the key revocation problem. . In this RIB-BPRE scheme, the agent can undo a set of delegates specified by the principal from the re-encryption key. They also pointed out that the identity-based broadcast agent re-encryption (RIB-BPRE) schemes do not take advantage of cloud computing, thus causes inconvenience to cloud users.
- To increase security, a secure data sharing cloud (SeDaSC) method using a single encryption key to encrypt files. Which provides data confidentiality ,integrity and other functions.

3.1.1 DISADVANTAGES

- The system doesn’t have mote security due to lack of less security cryptography techniques.
- There is no authentication and key agreement for enhancing more security on data

3.2 PROBLEM STATEMENT

- Cross-cloud data migration is one of the prevailing challenges faced by mobile users, which is an essential process when users change their mobile phones to a different provider.
- Due to the insufficient local storage and computational capabilities of the smart phones, it is often very difficult for users to backup all data from the original cloud servers to their mobile phones in order to further upload the downloaded data to the new cloud provider

3.3 PROPOSED SYSTEM

- The system proposes a peer-to-peer cloud authentication and key agreement (PCAKA) scheme based on anonymous identity to solve the problem of trust between cloud servers. Based on the elliptic curve certificate-free cryptography, our scheme can establish secure session keys between cloud service providers to ensure session security.
- The novelty of the proposed scheme lies in the fact that it eliminates the need for trusted authority (TA) and simplifies operations while maintaining security. In our scheme, the cloud servers enable the data owners in need of the data migration services to act as trusted third authority, so that they can verify each other and establish trusted session keys after each of the involved users performs some computation independently.
- The proposed scheme uses server anonymity to protect the privacy of service providers and users. It is worthy of note that both the two cloud servers involved in the migration process use anonymous identities for mutual authentication and key agreement providers to gain unnecessary information such as the brand of the old and new mobile phones belonging to the users respectively. Thus,this methodology maintains the privacy of the users by not revealing his/her personal choice.
- It provides identity traceability to trace malicious cloud servers. If the cloud service providers exhibit any errors or illegal operations in the service

process, users can trace back to the real identity of the corresponding cloud server based on the anonymous identity.

3.3.1 ADVANTAGES

- The system achieves efficient revocation, efficient file access and immediate revocation simultaneously.
- The system stores encrypted data on the cloud, but never reveals the decryption keys to the cloud. This protects the confidentiality of the file data.

3.4 SYSTEM REQUIREMENT SPECIFICATION

3.4.1 FUNCTIONAL REQUIREMENTS

- Monitoring

  These include any product or service health monitoring, failure conditions, error detection, logging, and correction.
- Maintenance

  These requirements include modularity, complexity, or interface design. Requirements should not be placed here simply because they are thought to be good design practices.
- Operations

- periods of interactive operations and periods of unattended operations
- data processing support functions
- backup and recovery operations
- safety considerations and requirements
- disaster recovery and business resumption

3.4.2 NON-FUNCTIONAL REQUIREMENTS

- Usability:

Prioritize the important functions of the system based on usage patterns. Frequently used functions should be tested for usability, as should complex and critical functions. Be sure to create a requirement for this.

- Reliability:

Reliability defines the trust in the system that is developed after using it for a period of time. It defines the likeability of the software to work without failure for a given time period .The number of bugs in the code, hardware failures, and problems can reduce the reliability of the software.

- Performance:

The response times are measured from any point and under particular circumstances. Calculating the specific peak times when load on system is unusually high.

- Supportability:

The system needs to be cost-effective to maintain. Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system.

3.4.3 HARDWARE REQUIREMENTS

- Processor - Intel(R) Core(TM) i5-8265U CPU
- RAM - 4 GB(MIN)
- Hard Disk - 128 GB

3.4.4 SOFTWARE REQUIREMENTS

- Operating System - Windows XP
- Coding Language - Java/J2EE(JSP,Servlet)
- Front End - J2EE
- Back End - MySQL

4.SYSTEM DESIGN

4.1 SYSTEM COMPONENTS

4.1.1 Data Owner

In this module, Data owner has to register to cloud and logs in, the data owner will do the following operations such Upload Data, Search File, Download File, Verify Data, List All Files to Migrate.

4.1.2 Cloud Server

In this module, the cloud will authorize both the owner and the user and do the following operations such as View all Users, View all Data Files ,View all Migrated File, View all transactions, View all attackers, View Time Delay Results, View Throughput Results.

4.1.3 Mobile Terminal

In this module, the terminal has to register to cloud and logs in. before the user can operate for the file details the terminal must do the following operations such as View All Transactions, View All Cloud Files, View All Migrated File, Cloud Tasks.

4.2 DATA FLOW DIAGRAM

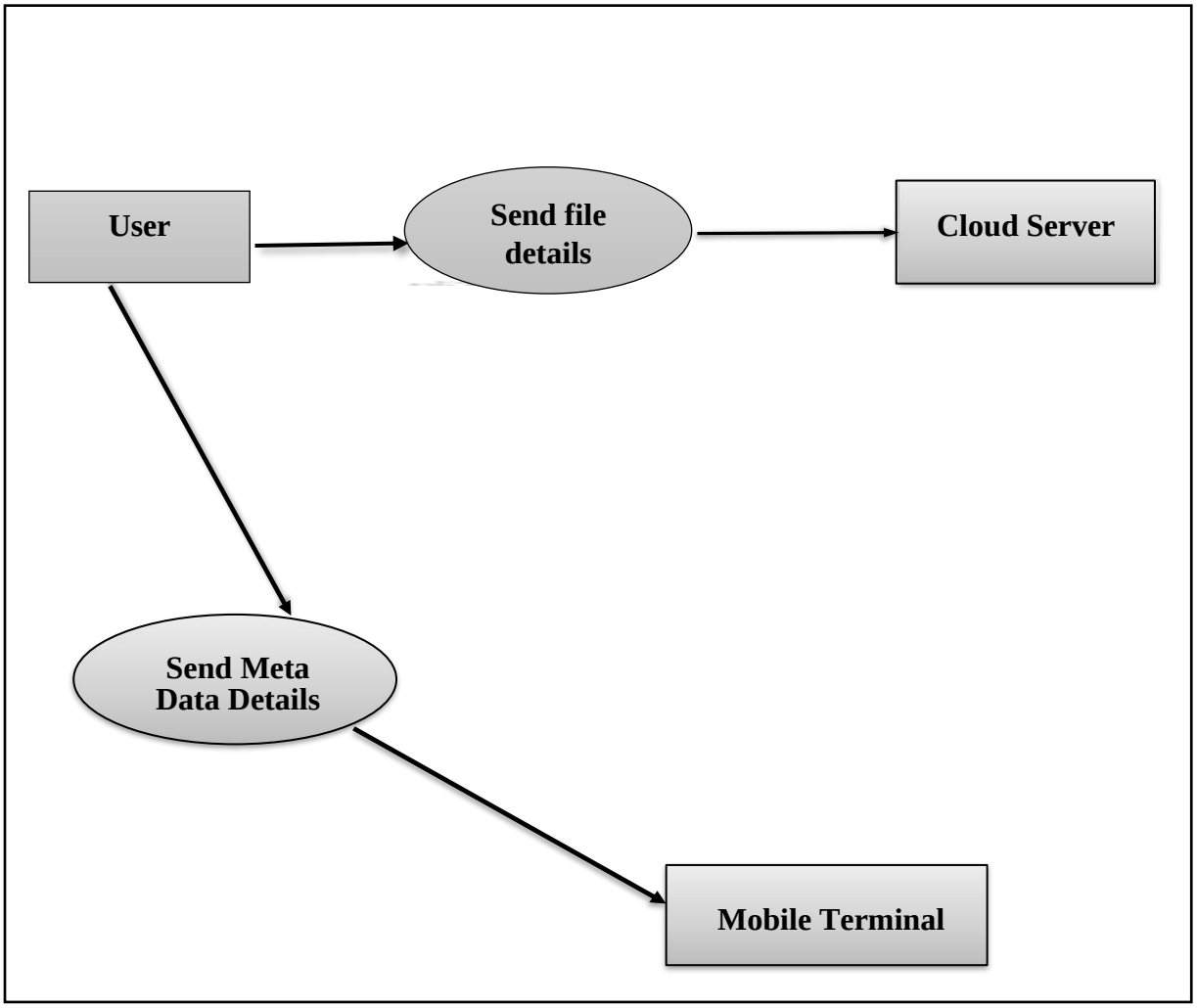

Fig 4.1 Level 0

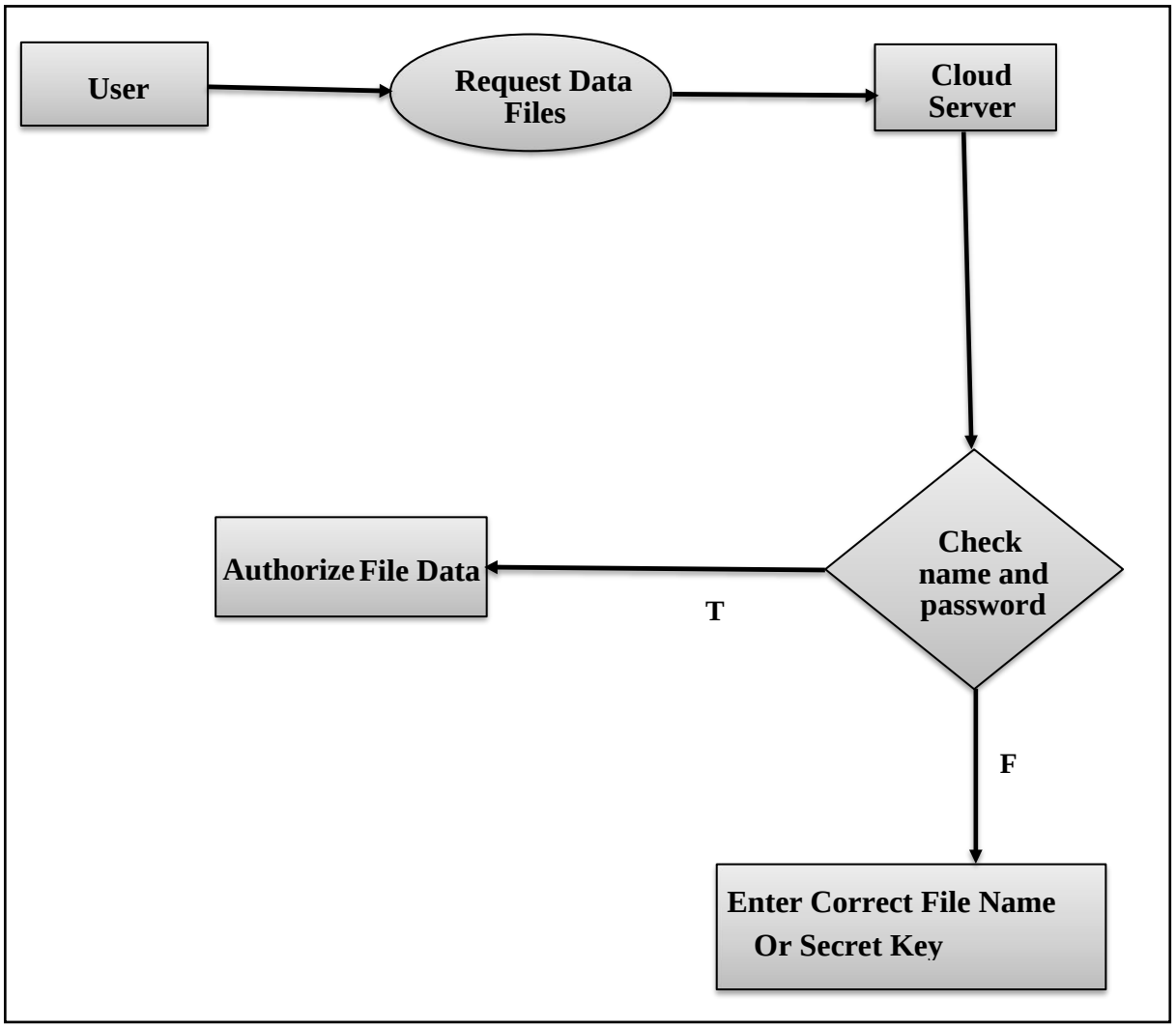

Fig 4.2 Level 1

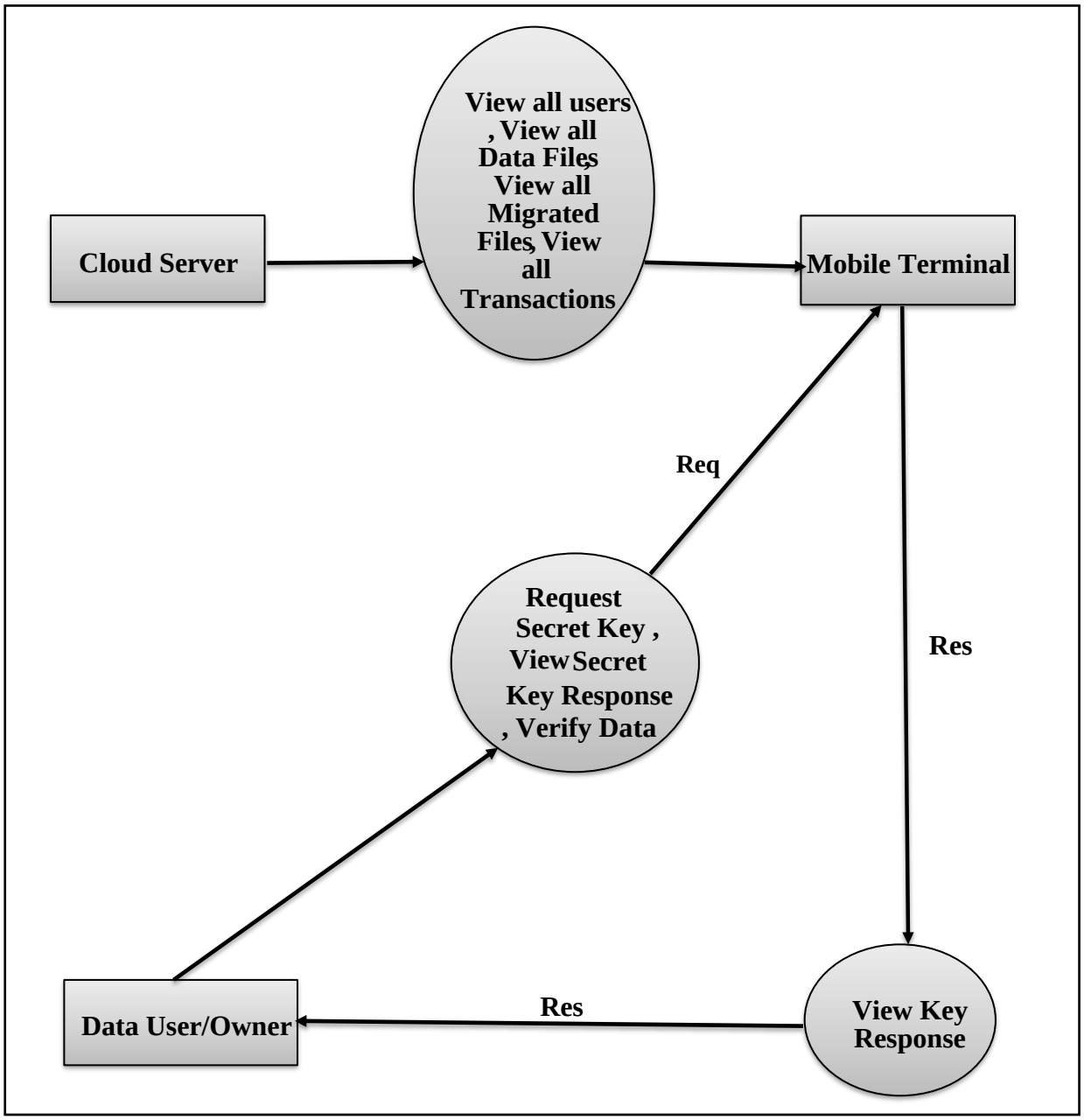

Fig 4.3 Level 2

4.3 UML DIAGRAMS

4.3.1 Use Case

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

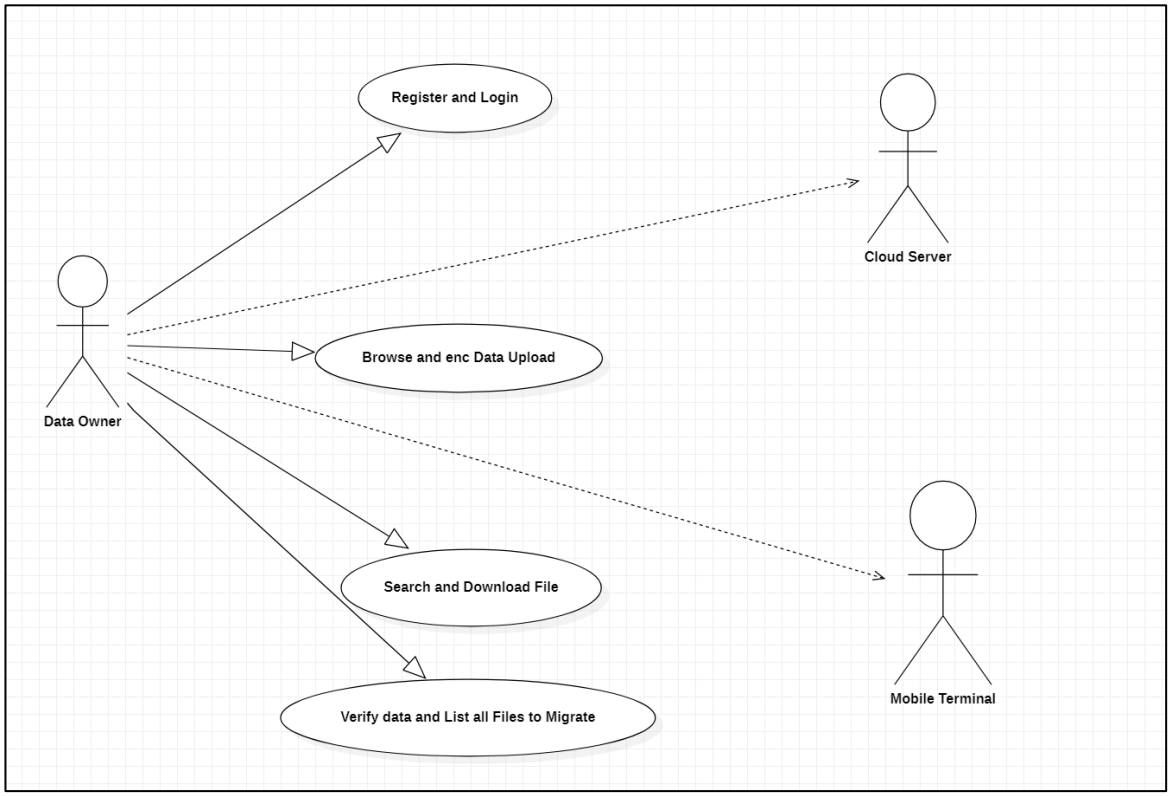

Fig 4.4 Data Owner

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

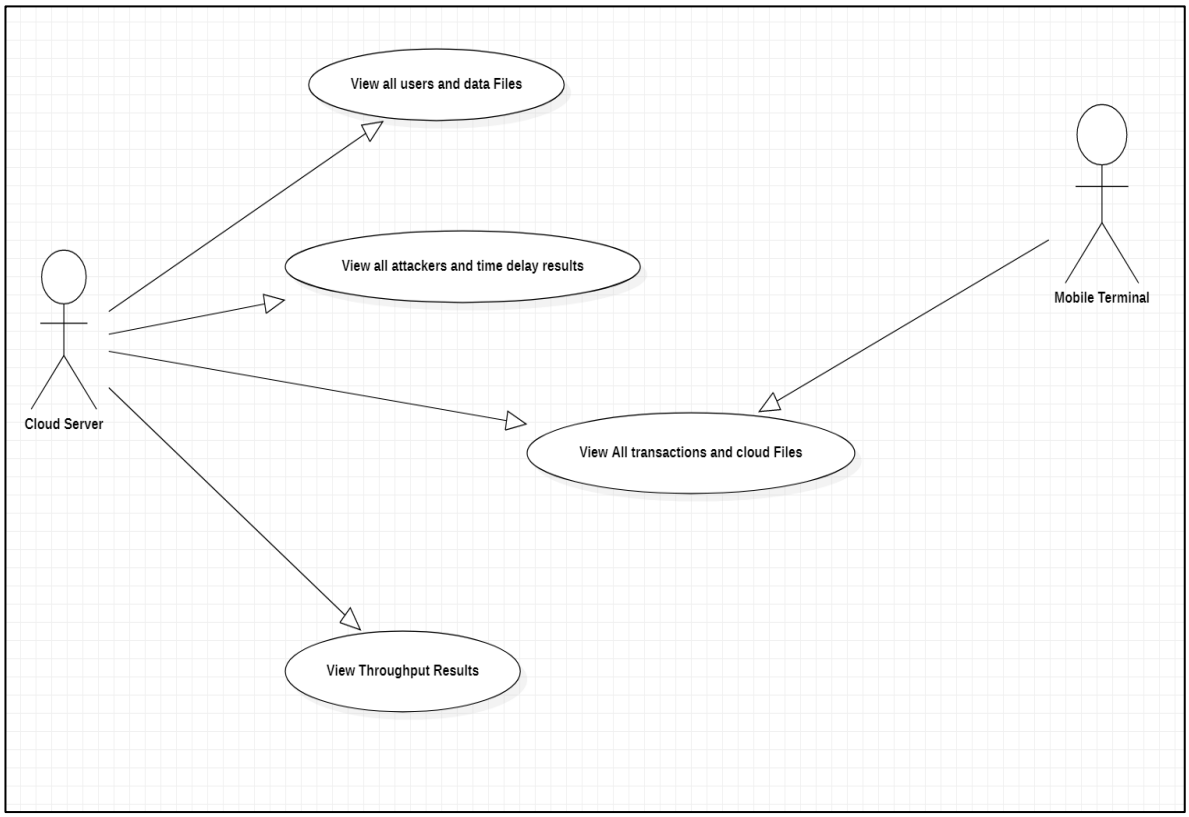

Fig 4.5 Cloud Server

It is essential to analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram.

After that, we will enlist the actors that will interact with the system. The actors are the person or a thing that invokes the functionality of a system. It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.Once both the actors and use cases are enlisted, the relation between the actor and use case/ system is inspected. It identifies the no of times an actor communicates with the system. Basically, an actor can interact multiple times with a use case or system at a particular instance of time.

Following are some rules that must be followed while drawing a use case diagram:

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.
2. The communication of an actor with a use case must be defined in an understandable way.

3. Specified notations to be used as and when required.
4. The most significant interactions should be represented among the multiple no of interactions between the use case and actors.

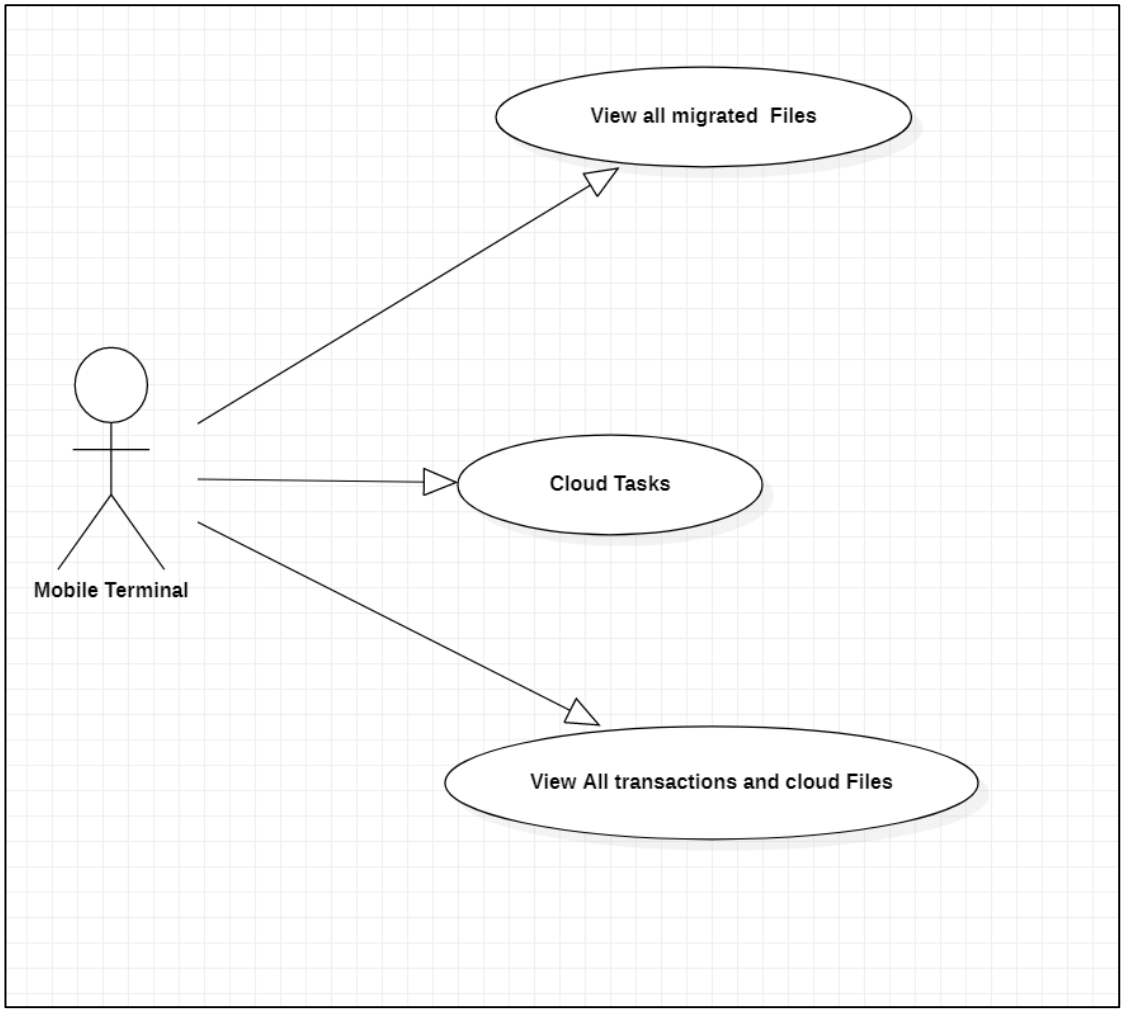

Fig 4.6 Mobile Terminal

4.3.2 Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

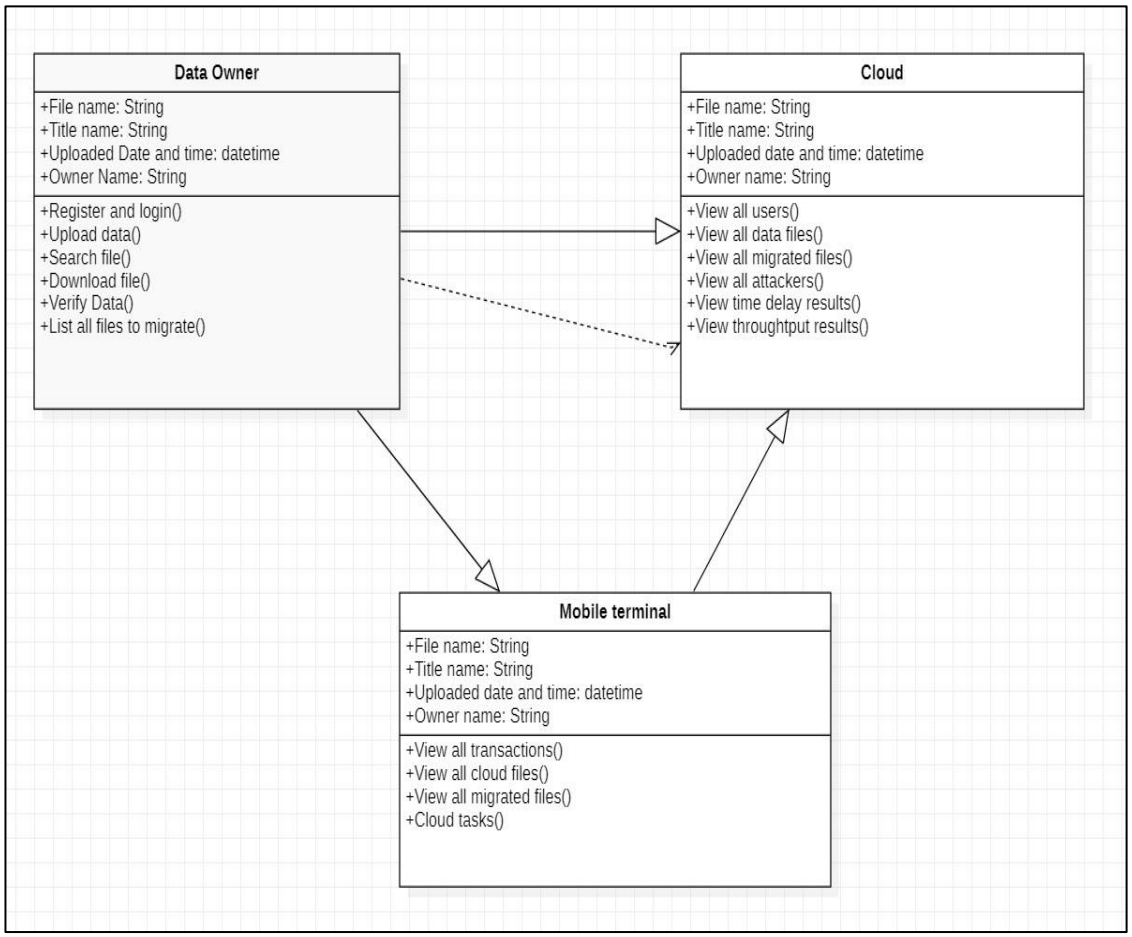

Fig 4.7 Class Diagram

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

4.3.3 Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching. The main purpose of sequence diagram are

- To model high-level interaction among active objects within a system.
- To model interaction among objects inside a collaboration realizing a use case.
- It either models generic interactions or some certain instances of interaction.

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence Diagrams captures:

- The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- High-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

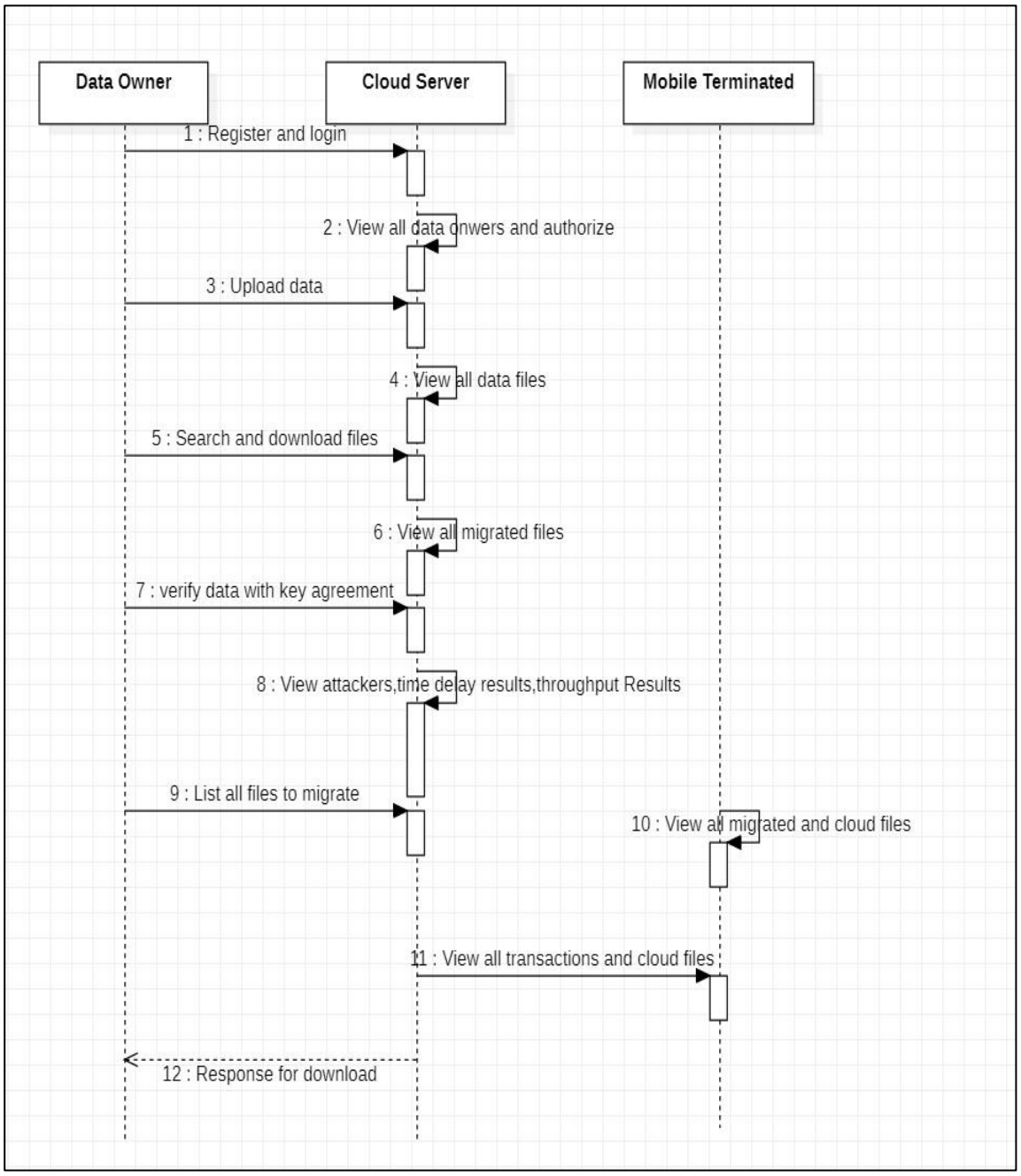

Fig 4.8 Sequence Diagram

4.3.4 ER Diagram

4.3.3.1 User

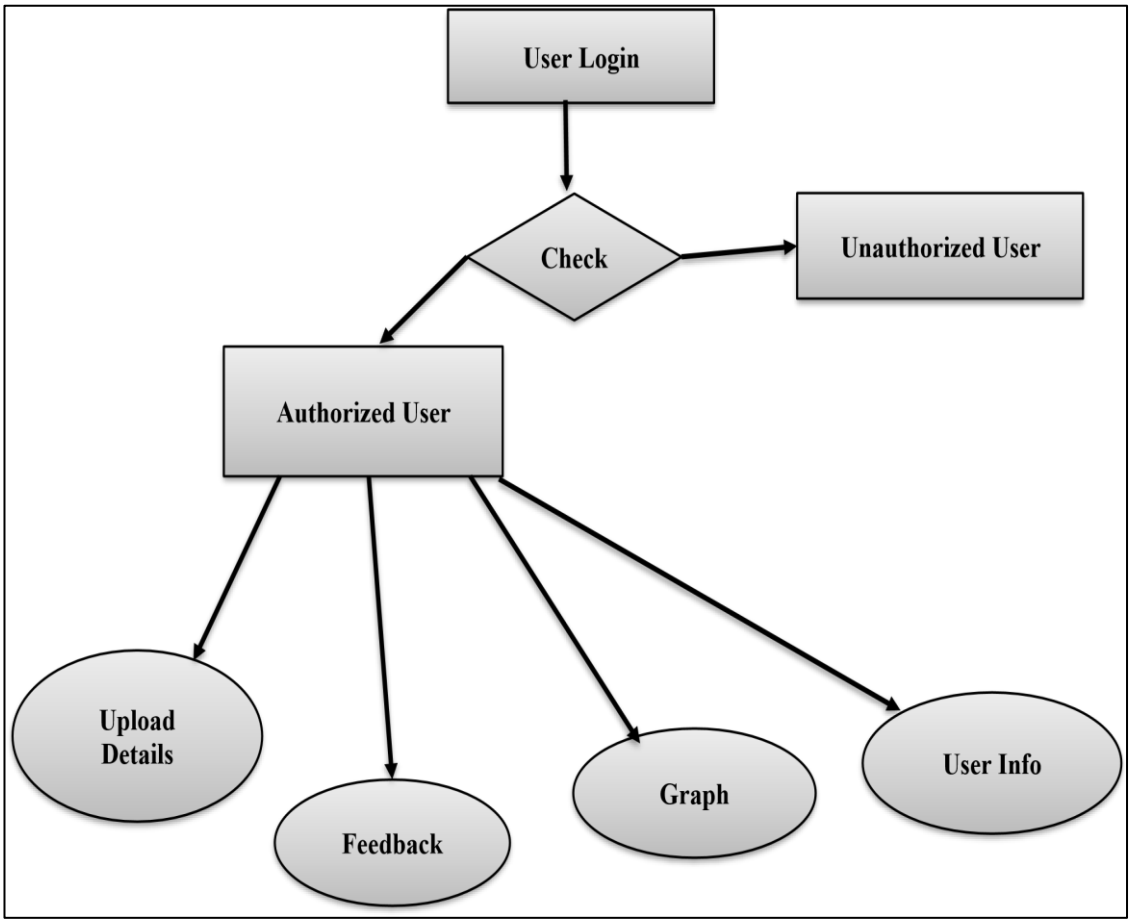

Fig 4.9 User

4.3.3.2 Admin

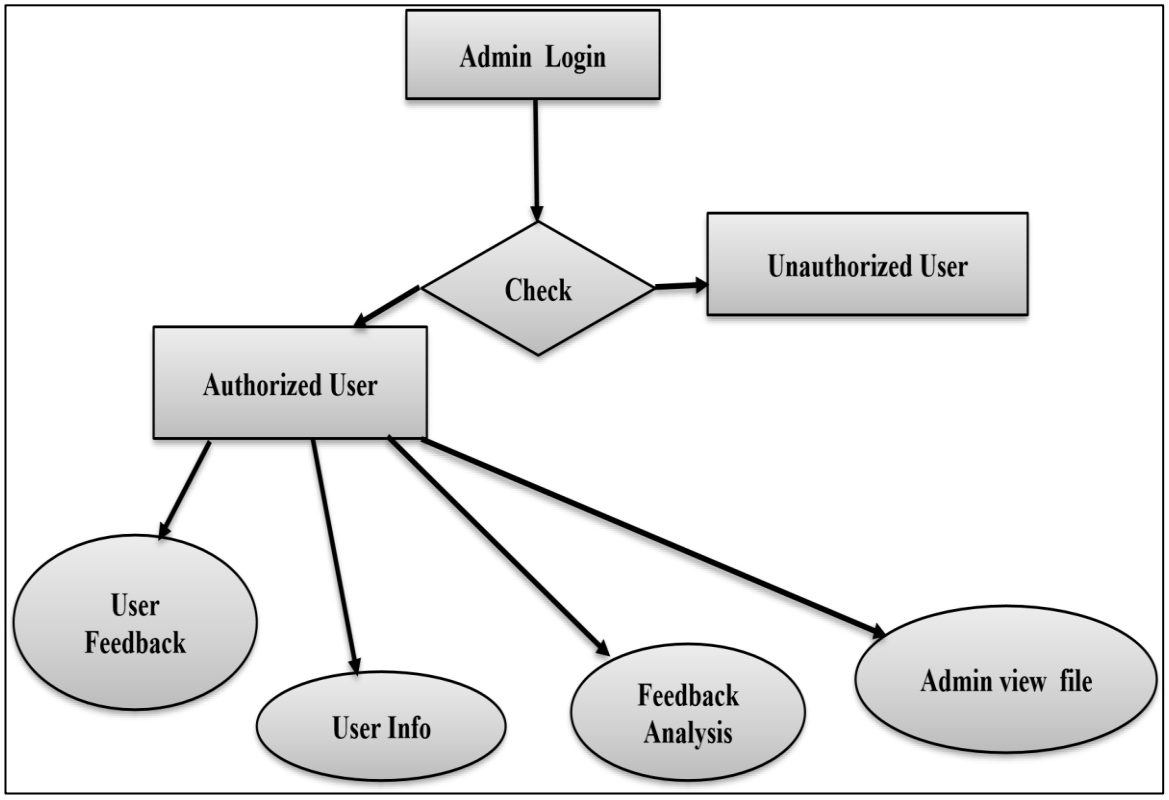

Fig 4.10 Admin

5.IMPLEMENTATION

5.1SAMPLE CODE

- **Connect.jsp**

```
<%@ include file="connect.jsp" %>
<%@ page import="java.util.*" %>
<%@ page import="java.io.*"%>
<%@ page import="com.oreilly.servlet.*"%>
<%
ArrayList list = new ArrayList();
ServletContext context = getServletContext();
String dirName=context.getRealPath("Gallery/");
String paramname=null,uname=null,pass=null,email=null,mno=null,addr=null,dob=null,gender=null,pincode=null,location=null,image=null,cloud=null,bin = "";
FileleInputStream fs=null; File file1 = null; try {
MultipartRequest multi = new MultipartRequest(request, dirName,10 * 1024 *
1024);// 10MB
Enumeration params = multi.getParameterNames();
while (params.hasMoreElements())
{ paramname = (String) params.nextElement();
if(paramname.equalsIgnoreCase("userid")){
 uname=multi.getParameter(paramname);
} if(paramname.equalsIgnoreCase("pass"))
        pass=multi.getParameter(paramname);
} if(paramname.equalsIgnoreCase("email")
{ email=multi.getParameter(paramname);
```

```
} if(paramname.equalsIgnoreCase("mobile"))
{ mno=multi.getParameter(paramname);
} if(paramname.equalsIgnoreCase("address"))
{ addr=multi.getParameter(paramname);
} if(paramname.equalsIgnoreCase("dob"))
{ dob=multi.getParameter(paramname);
} if(paramname.equalsIgnoreCase("gender"))
{ gender=multi.getParameter(paramname);
} if(paramname.equalsIgnoreCase("pincode"))
{ pincode=multi.getParameter(paramname);
} if(paramname.equalsIgnoreCase("cloud"))
{
cloud=multi.getParameter(paramname);
} } int f = 0;
Enumeration files = multi.getFileNames(); while (files.hasMoreElements())
{ paramname = (String) files.nextElement(); if(paramname != null) f = 1;
image = multi.getFilesystemName(paramname); String fPath =
context.getRealPath("Gallery\\"+image); file1 = new File(fPath); fs = new
FileInputStream(file1); list.add(fs);
String ss=fPath;
FileInputStream fis = newFileInputStream(ss); StringBuffer sb1=new StringBuffer(); int i = 0;
 while ((i = fis.read()) != -1)
{ if (i != -1)
{
 String hex=Integer.toHexString(i);
sb1.append(hex); String binFragment = ""; int iHex; for(int i1= 0; i1 < hex.length();  i1++)
```

```
{ iHex =Integer.parseInt(""+hex.charAt(i1),16); binFragment = Integer.toBinaryString(iHex);
while(binFragment.length() < 4)
{ binFragment = "0" + binFrag;
} bin += binFragment;
}
}
}}}
FileInputStream fs1 = null;
String query1="select * from dataowner  where username='"+uname+"' and cloud='"+cloud+"'";
Statement st1=connection.createStatement(); ResultSet rs1=st1.executeQuery(query1); if ( rs1.next() ){
out.print("Owner Name Already Exist in  ");%><%=cloud%><%
%>
<p><a href="dataOwnerRegister.jsp">Back</a></p>
<%
}
else
{
PreparedStatement ps=connection.prepareStatement("insert into dataowner
(username,password,email,mobile,address,dob,gender,pincode,image,cloud)
values(?,?,?,?,?,?,?,?,?,?)");
ps.setString(1,uname);
 ps.setString(2,pass);
 ps.setString(3,email);
ps.setString(4,mno);
ps.setString(5,addr);
 ps.setString(6,dob);
```

```
 ps.setString(7,gender);
 ps.setString(8,pincode);
 ps.setBinaryStream(9, (InputStream)fs, (int)(file1.length())); ps.setString(10,cloud);
 if(f == 0) ps.setObject(9,null);
else if(f == 12){ fs1 = (FileInputStream)list.get(0); ps.setBinaryStream(9,fs1,fs1.available());
}
int x=ps.executeUpdate();
if(x>0)
 {
 out.print("Registered Successfully");
%>
<p><a href="dataOwnerRegister.jsp">Back</a></p>
<%
}
}
}
catch (Exception e)
{
out.println(e.getMessage());
}
%>
```

6.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, sub-assemblies, 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 tests. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTING

6.1.1 Unit Testing

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

6.1.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.1.3 Functional Testing

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.4 System Testing

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

It is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

6.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. The test provides inputs and responds to outputs without considering how the software works.

6.2TESTING STRATEGIES

6.2.1 Integration Testing

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

The task of the integration test is to check that components or software applications,

e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully.

No defects encountered.

- **Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

- **Bottom up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up,

processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality

6.2.2 Acceptance Testing

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

Test Results: All the test cases mentioned above passed successfully.

No defects encountered.

6.3 TEST CASES

Table 6.1 Test Cases

S.NO	TEST CASE	EXPECTED RESULT	RESULT	REMARKS (if fails)
1	Data owner	Searches, Downloads and Uploads files	Pass	Doesn’t download files, if not uploaded.
2	Cloud server	Authorize both user and owner to perform operations	Pass	Doesn’t view the users, if they didn’t login
3	Mobile terminal	Register to cloud and performs	Pass	Displays error if not registered

6.4 RESULTS AND DISCUSSION

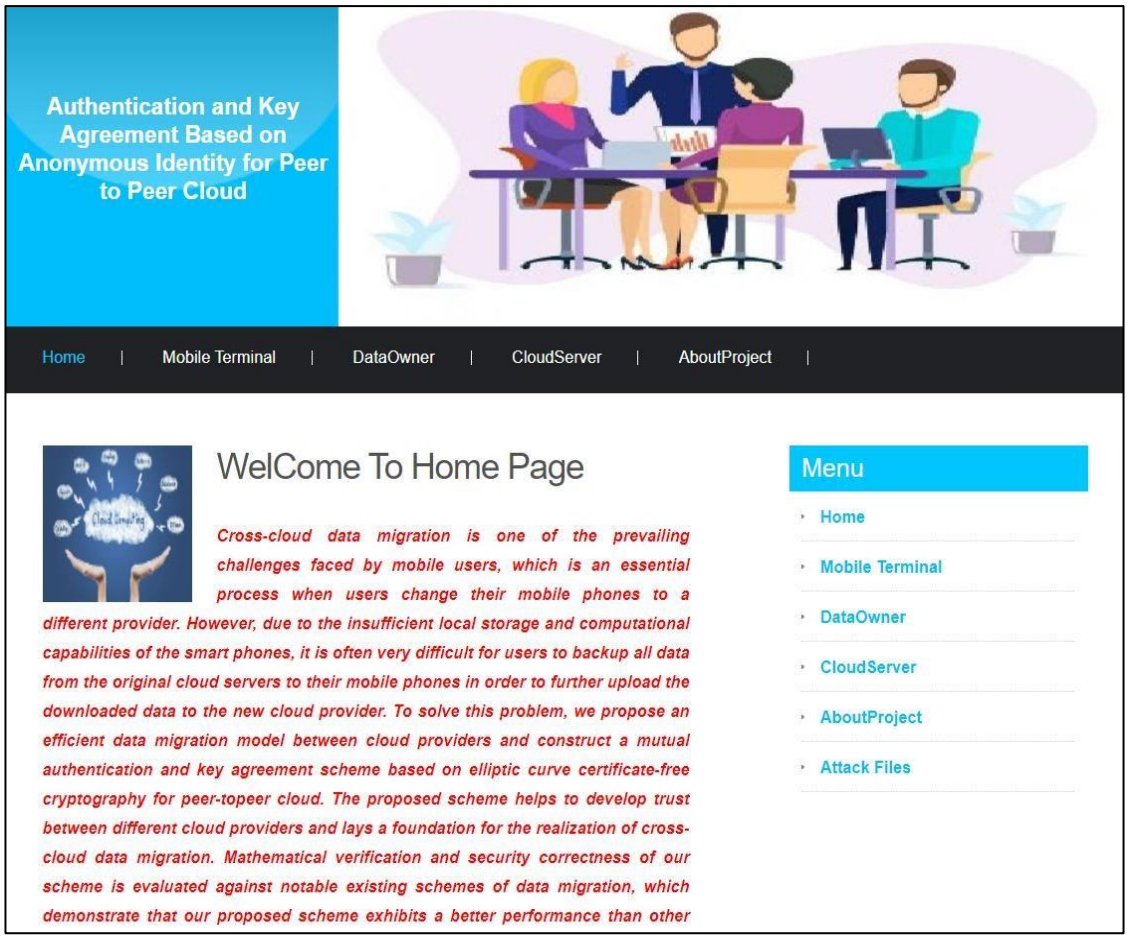

Fig 6.1 Home Page

This is the home page . It contains the menu in which we can log into the different modules

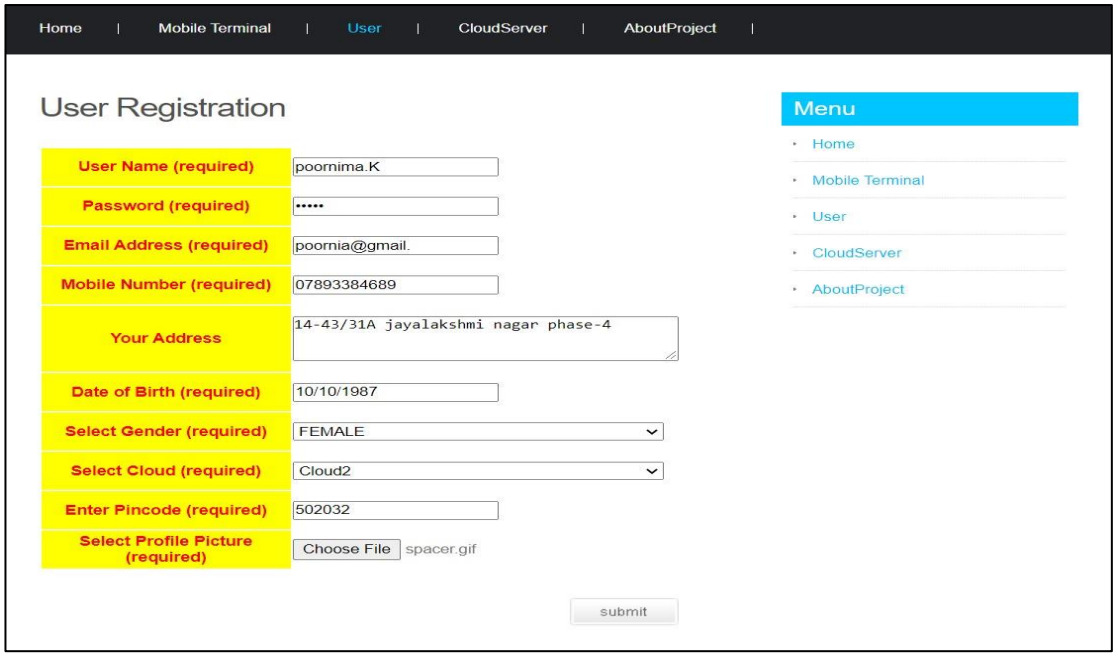

Fig 6.2 User Registration page

This the User registration page where we can fill all the details related to the user to create a profile.

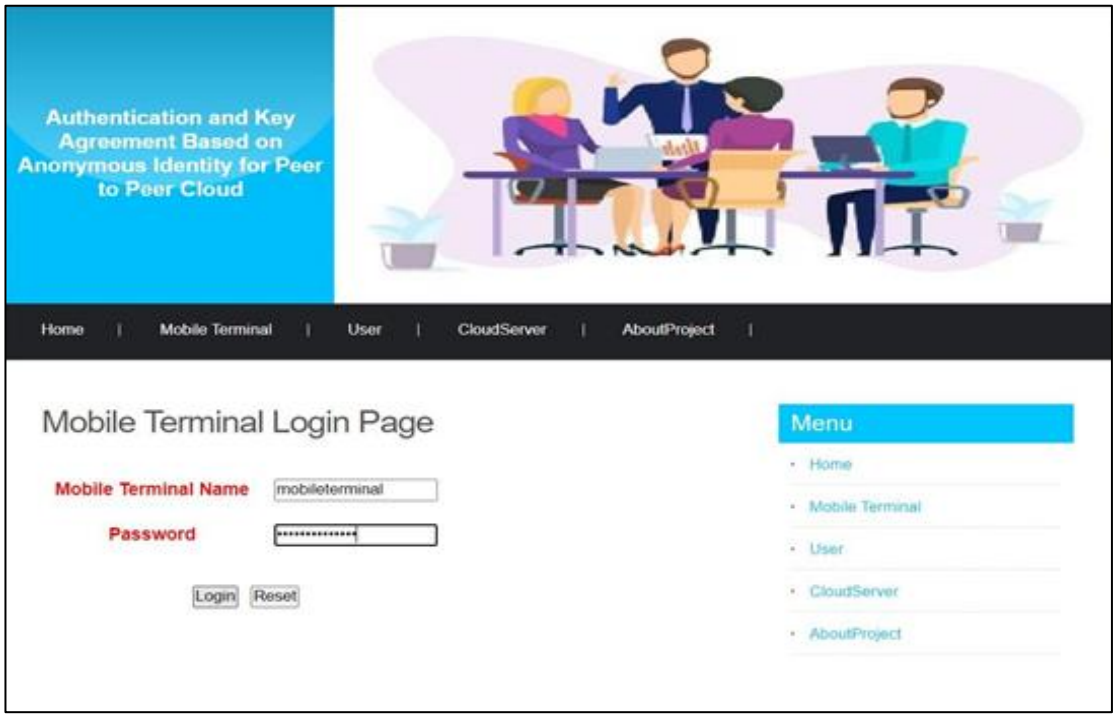

Fig 6.3 Mobile Terminal Login Page

This is the Mobile terminal login page where the credentials need to entered to login.

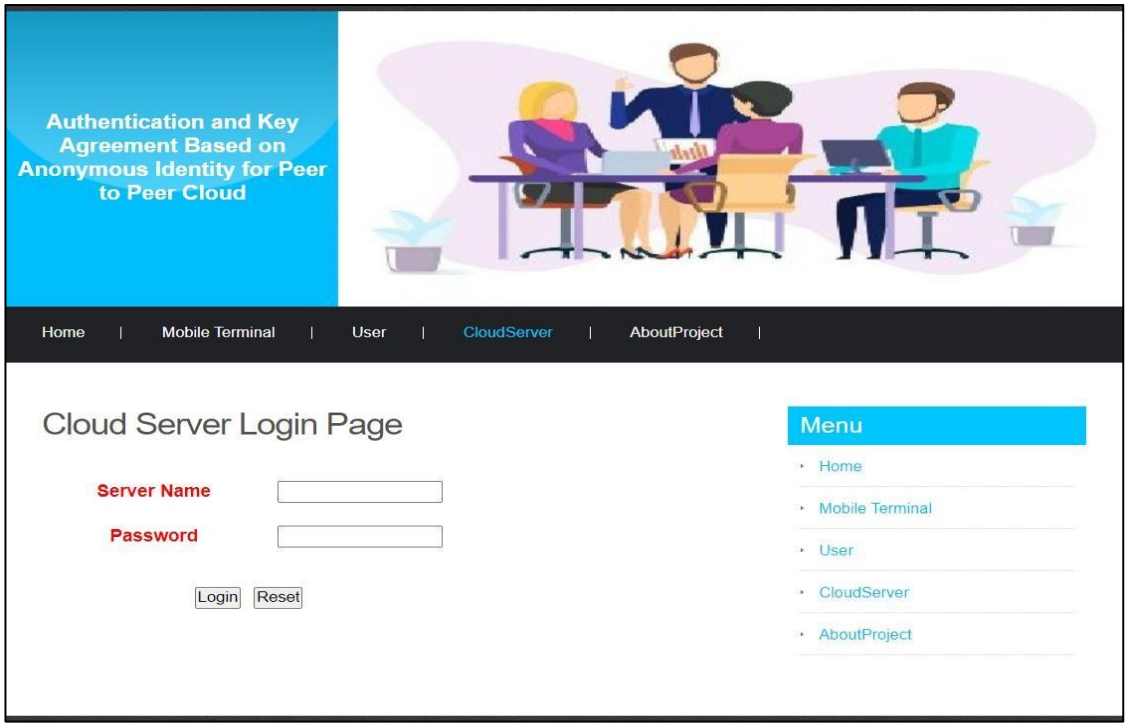

Fig 6.4 Cloud server login page

This is the Cloud Sever login page where the credentials need to be entered to login.

Fig 6.5 User Profile

This is the user profile. The user can upload data, search and download file, verify data and list all files to migrate.

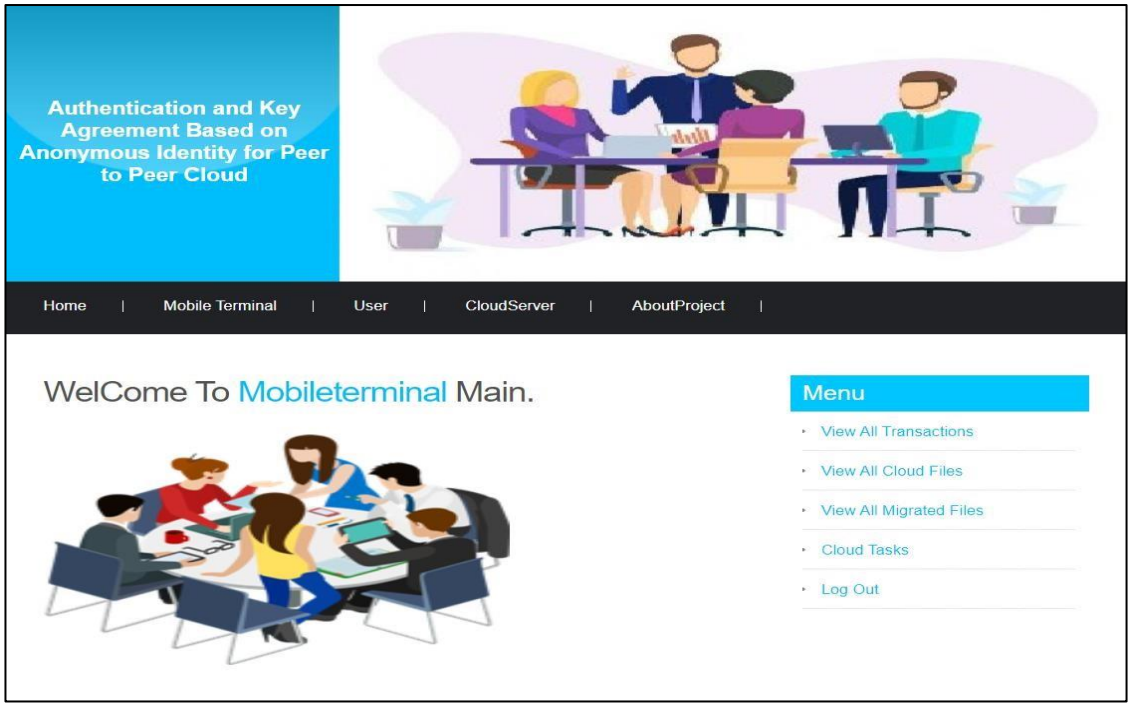

Fig 6.6 Mobile Terminal

This is the Mobile Terminal page. It views all transactions, cloud files and migrated files.

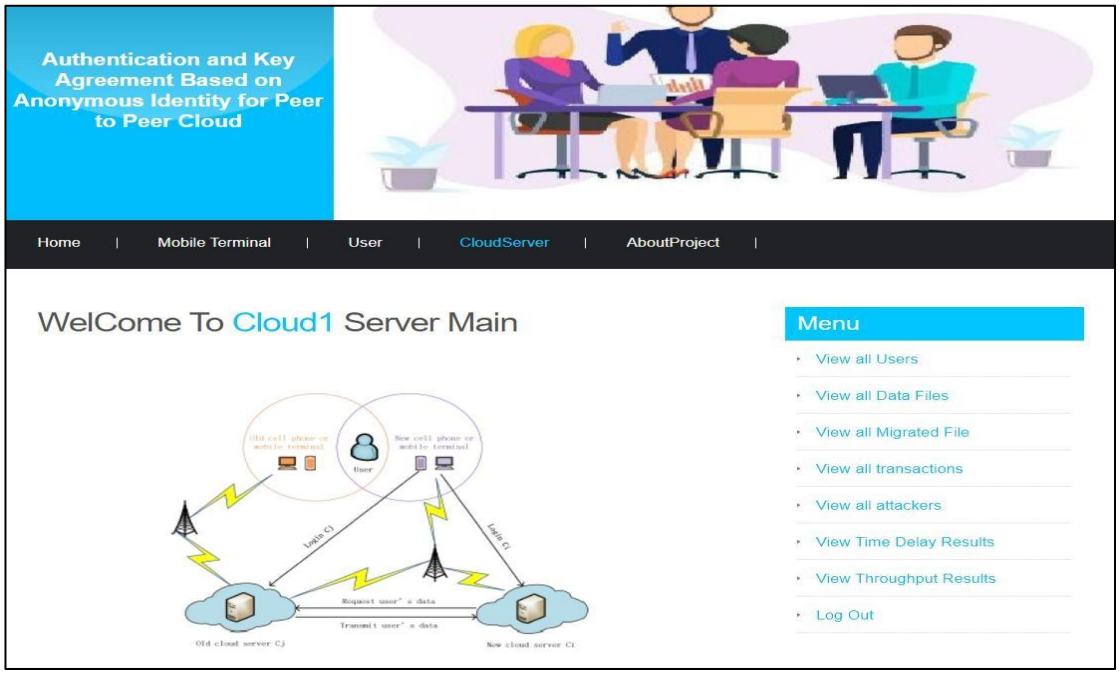

Fig 6.7 Cloud Server

This is the Cloud server page. It views all users,data files,migrated files,attackers,time delay results, throughput results, transactions, cloud files and migrated files.

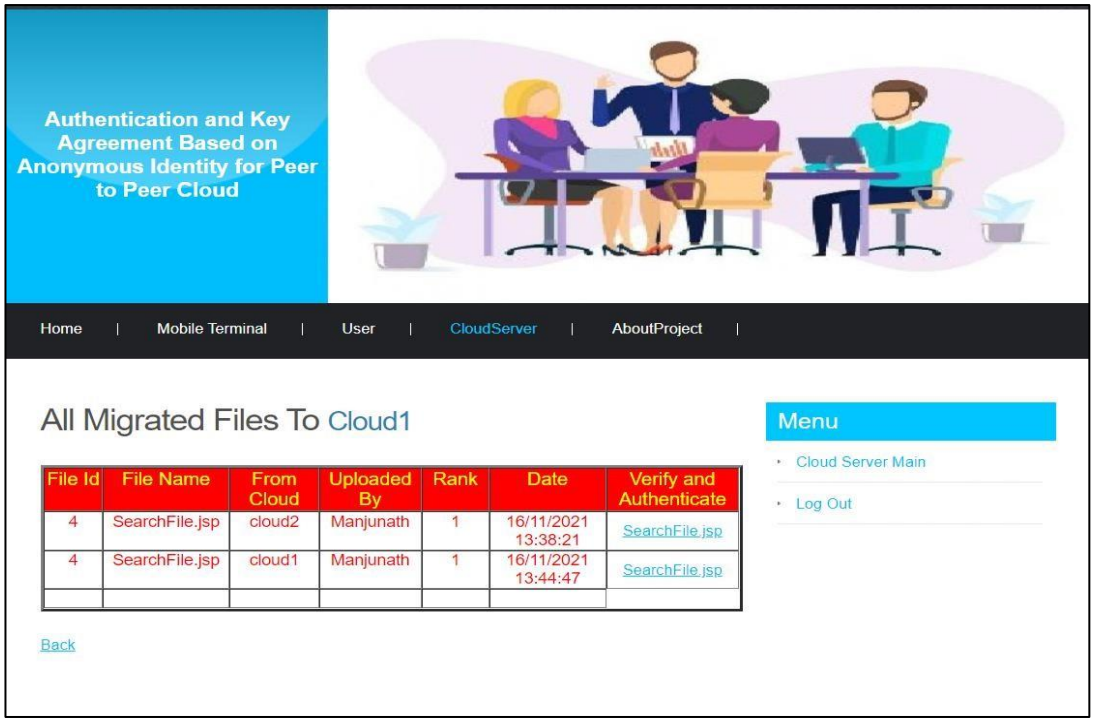

Fig 6.8 Migrated Files

This page shows all the files that are migrated between the clouds.

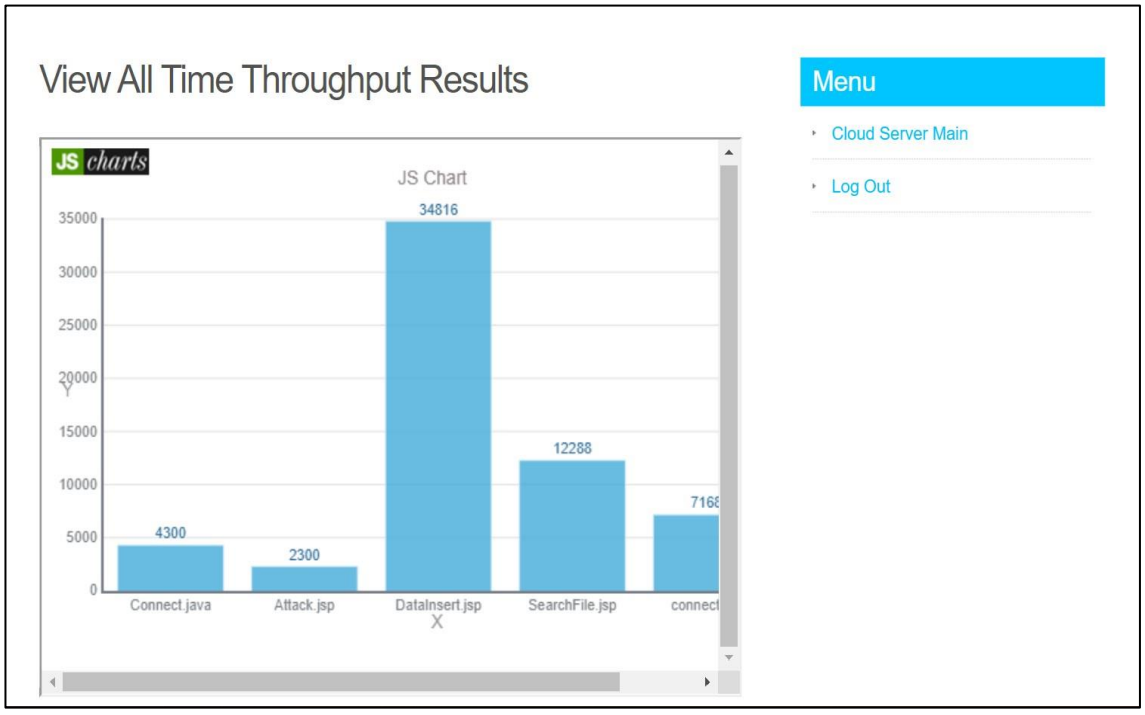

Fig 6.9 View all throughput results

This page shows all the throughput results of data insertion, searching file and uploading data in the form of a bar graph

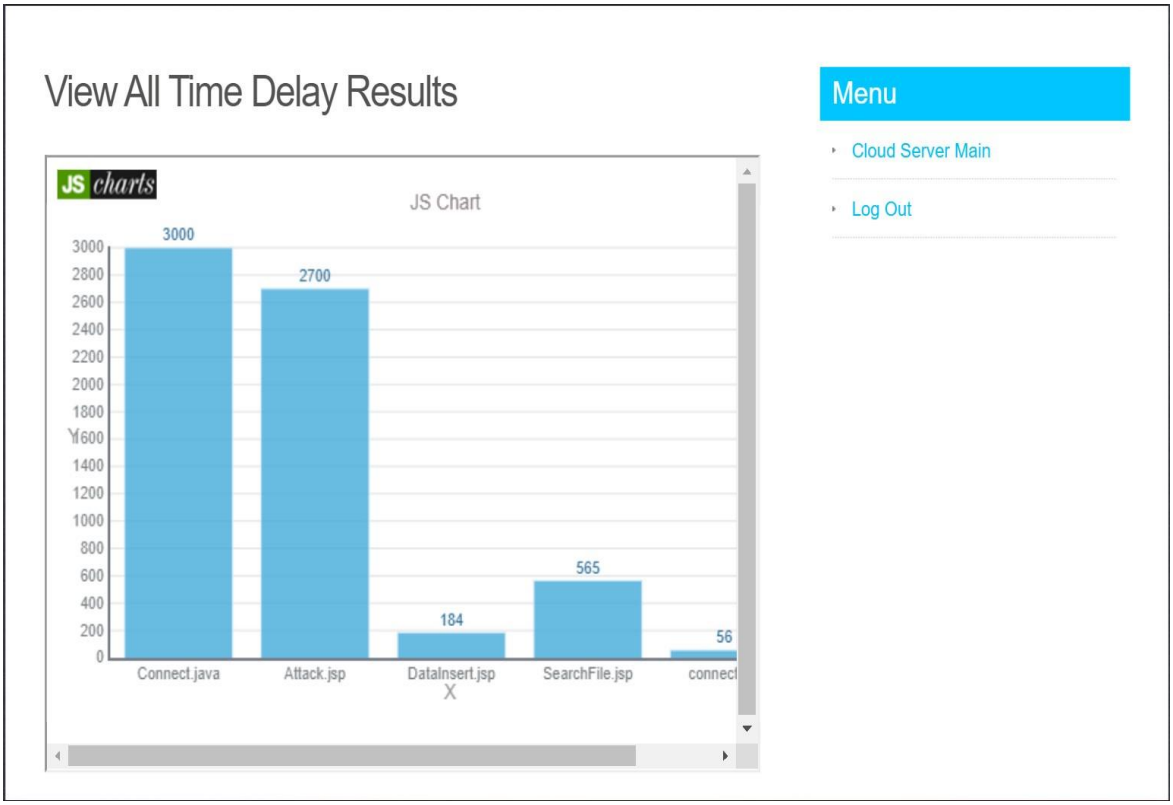

Fig 6.10 View all Time Delay results

This page shows all the results related to time taken during each operations and mentions its delay in the form of bar graph.

7.CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The scheme to transfer user data between different cloud servers based on a key agreement protocol. Through the mathematical analysis and comparative evaluation presented in this paper, the advantages of our scheme are proved from three aspects: security performance, calculation costs and communication costs. The proposed scheme can efficiently solve the primary problem of trust during data migration between cloud servers and further can provide anonymity for the identity of cloud servers. On the premise of protecting the privacy of cloud service providers, our proposed scheme indirectly protects the privacy of users. In addition, the identity traceability provided by our proposed scheme also enables users to effectively constrain the cloud service providers.

7.2 FUTURE ENHANCEMENT

As a future work, we plan to explore and develop a protocol that allows multiple users to share data across different cloud servers, with the motivation of enhancing the efficiency of data sharing among multiple users.

8.REFERENCES

1. C. I. network information center, “The 44th china statistical report on internet development,” http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/ 201908/P020190830356787490958.pdf, 2019.
2. B. Li, J. Li, and L. Liu, “Cloudmon: a resource-efficient iaas cloud monitoring system based on networked intrusion detection system virtual appliances,” Concurrency and Computation: Practice and Experience, vol. 27, no. 8, pp.a.1861–1885, 2015.
3. J. Cui, H. Zhou, H. Zhong, and Y. Xu, “Akser: attribute-based keyword search with efficient revocation in cloud computing,” Information Sciences, vol. 423, pp. 343–352, 2018.
4. J. Cui, H. Zhong, W. Luo, and J. Zhang, “Area-based mobile multicast group key management scheme for secure mobile cooperative sensing,” Science China Information Sciences, vol. 60, no. 9, p. 098104, 2017.
5. J. Cui, H. Zhou, Y. Xu, and H. Zhong, “Ooabks: Online/offline attributebased encryption for keyword search in mobile cloud,” Information Sciences, vol. 489, pp. 63–77, 2019.
6. D. Petcu, “Portability and interoperability between clouds: challenges and case study,” in European Conference on a Service-Based Internet. Springer, 2011, pp. 62–74.
7. T. Binz, F. Leymann, and D. Schumm, “Cmotion: A framework for migration of applications into and between clouds,” in 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA). IEEE, 2011, pp. 1–4.
8. M. N. Shirazi, H. C. Kuan, and H. Dolatabadi, “Design patterns to enable data portability between clouds’ databases,” in 2012 12th International Conference on Computational Science and Its Applications. IEEE, 2012, pp. 117–120.
9. X. Liang, Z. Cao, H. Lin, and J. Shao, “Attribute based proxy reencryption with delegating capabilities,” in Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, 2009, pp. 276–286.

10. K. Liang, M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang, Y. Yu, and A. Yang, “A secure and efficient ciphertext-policy attributebased proxy reencryption for cloud data sharing,” Future Generation Computer Systems, vol. a.52, pp. 95–108, 2015.