INTRODUCTION

1.1 Overview

Cloud computing, also on-demand computing, is a kind of Internet-based computing that provides shared processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources, which can be rapidly provisioned and released with minimal management effort. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centres. It relies on sharing of resources to achieve coherence and economy of scale, similar to over a network.

Recent years have witnessed a "gold rush" of online data hosting services such as Amazon S3, Windows Azure, Google Cloud Storage, Aliyun OSS, and so forth. These services provide customers with reliable, scalable, and low-cost data hosting functionality. More and more enterprises and organizations are hosting all or part of their data into the cloud, in order to reduce the IT maintenance cost and enhance the data reliability. Existing clouds exhibit great heterogeneities in terms of both working performances and pricing policies. Different cloud vendors build their respective infrastructures and keep upgrading them with newly emerging gears. They also design different system architectures and apply various techniques to make their services competitive. Such system diversity leads to observable performance variations across cloud vendors. Pricing policies of existing storage services provided by different cloud vendors are distinct in both pricing levels and charging items. Vendor lock-in risk: Facing numerous cloud vendors as well as their heterogeneous performances/policies, customers may be perplexed with which cloud(s) are suitable for storing their data and what hosting strategy is cheaper. The general status quo is that customers usually put their data into a single cloud and then simply trust to luck. This is subject to the so-called "vendor lock-in risk", because customers would be confronted with a dilemma if they want to switch to other cloud venders risk first lies in that data migration inevitably generates considerable expense. For example, moving 100 TB of data from Amazon S3 (California datacentre) to Aliyun OSS (Beijing datacenter) would consume as much as 12,300 (US) dollars. The vendor lock-in risk makes customers suffer from price adjustments of cloud vendors which are not uncommon. For example, the fluctuation of electricity bills in a region will affect the prices of cloud services in this region. Cloud Storage have been adjusting their pricing terms

Nirvanix, which has thousands of customers including top 500 companies, suddenly shut down its cloud storage service in Sep. 2013. Though the service quality is formally guaranteed by service level agreements (SLA), failures and outages do occur. Almost all the major cloud vendors experienced service outages. Some outages even lasted for several hours. Multi-cloud data hosting. Recently, multi-cloud data hosting has received wide attention from researchers, customers, and startups.

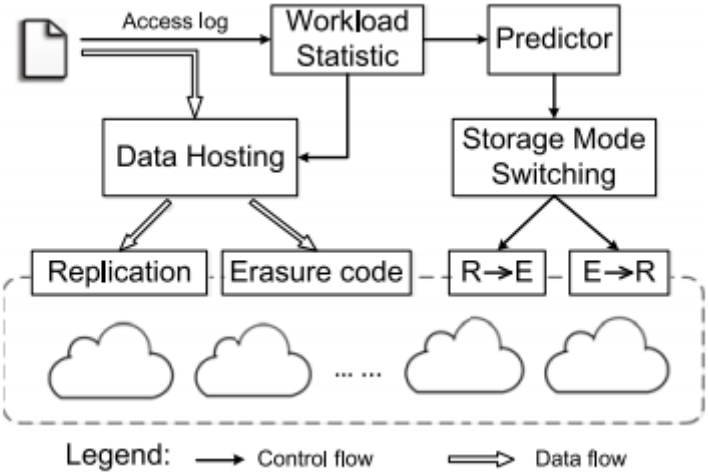

Fig 1.1: Architecture of CHARM

The whole model is located in the proxy in Figure 1.1. There are four main components in CHARM: Data Hosting, Storage Mode Switching (SMS), Workload Statistic, and Predictor. Workload Statistic keeps collecting and tackling access logs to guide the placement of data. It also sends statistic information to Predictor which guides the action of SMS. Data Hosting stores data using replication or erasure coding, according to the size and access frequency of the data. SMS decides whether the storage mode of certain data should be changed from replication to erasure coding or in reverse, according to the output of Predictor. The implementation of changing storage mode runs in the background, in order not to impact online service. Predictor is used to predict the future access frequency of files. The time interval for prediction is one month, that is, we use the former months to predict access frequency of files in the next month. A very simple predictor, which uses the weighted moving average approach, works well in data hosting model. Data Hosting and SMS are two important modules in CHARM. Data Hosting decides storage mode and the clouds that the data should be stored in. This is a complex integer programming problem demonstrated in the following subsections.

1.2 Problem Statement

Multi-cloud users encounter the two critical problems:

- To choose appropriate clouds to minimize monetary cost in the presence of heterogeneous pricing policies which mainly depends on the data-level usage, particularly storage capacity consumption and network bandwidth consumption.

- To meet the different availability requirements of different services where, the major concern lies in which redundancy mechanism is more economical based on specific data access patterns.

1.3 Objectives

1) To propose and implement CHARM, a novel, efficient, and heuristic-based data hosting scheme for heterogeneous multi-cloud environments is. CHARM accommodates different pricing strategies, availability requirements, and data access patterns. It selects suitable clouds and an appropriate redundancy strategy to store data with minimized monetary cost.
2) To design and implement a flexible transition scheme for CHARM. It keeps monitoring the variations of pricing policies and data access patterns, and adaptively triggers the transition process between different data storage modes.
3) To evaluate the performance of CHARM using two typical real-world traces and prototype experiments.
Both trace-driven simulation and experiment results confirm the efficacy of CHARM.

1.4 Motivation

Information processed by businesses, government organizations and individuals often comes with confidentiality and integrity requirements that the processing party must adhere to. As a result, data processors must deploy security controls for their ICT infrastructure, protecting it against external as well as internal attackers. This is relatively easy when this infrastructure is local and controlled by the processing party, but much harder when it is provided by an external service provider. Cloud services promise great benefits in terms of financial savings, easy and convenient access to data and services, as well as business agility. Organizations and individuals therefore choose to outsource their data to the cloud, where an untrusted party is in

charge of storage and computation. A major concern for the adoption of cloud computing is the inability of the cloud to build user trust in the information security measures deployed in cloud services. Common computing techniques cannot be applied on encrypted data, and therefore the data and the programs that compute on the data must be decrypted before being run on the cloud infrastructure. A comprehensive solution for securing the cloud computing infrastructure can be based on cryptographic mechanisms of secure computation. These mechanisms allow for distributed computation of arbitrary functions of private (secret) inputs, while hiding any information about the inputs to the functions. Put differently, these mechanisms support computation on encrypted data.

LITERATURE SURVEY

2.1Towards Network-level Efficiency for Cloud Storage Service

Cloud storage services such as Dropbox, Google Drive, and Microsoft OneDrive provide users with a convenient and reliable way to store and share data from anywhere, on any device, and at any time. The cornerstone of these services is the data synchronization (sync) operation which automatically maps the changes in users' local filesystems to the cloud via a series of network communications in a timely manner. If not designed properly, however, the tremendous amount of data sync traffic can potentially cause (financial) pains to both service providers and users. This paper addresses a simple yet critical question: Is the current data sync traffic of cloud storage services efficiently used? First define a novel metric named TUE to quantify the Traffic Usage Efficiency of data synchronization. Based on both real-world traces and comprehensive experiments, study and characterize the TUE of six widely used cloud storage services and results demonstrate that a considerable portion of the data sync traffic is in a sense wasteful, and can be effectively avoided or significantly reduced via carefully designed data sync mechanisms. All in all, the study of TUE of cloud storage services not only provides guidance for service providers to develop more efficient, traffic economic services, but also helps users pick appropriate services that best fit their needs and budgets.

Cloud storage services such as Dropbox, Google Drive, and Microsoft OneDrive (renamed from SkyDrive since Feb. 2014) provide users with a convenient and reliable way to store and share data from anywhere, on any device, and at any time. The users' data (e.g., documents, photos, and music) stored in cloud storage are automatically synchronized across all the designated devices (e.g., PCs, tablets, and smartphones) connected to the cloud in a timely manner. With multiplicity of devices – especially mobile devices – that users possess today, such "anywhere, anytime" features significantly simplify data management and consistency maintenance, and thus provide an ideal tool for data sharing and collaboration. In a few short years, cloud storage services have reached phenomenal levels of success, with the user base growing rapidly.

Despite the late entry into this market (in Apr. 2012), Google Drive obtained 10 million users just in its first two months. The key operation of cloud storage services is data

synchronization (sync) which automatically maps the changes in users' local filesystems to the cloud via a series of network communications. In a cloud storage service, the user usually needs to assign a designated local folder (called a "sync folder") in which every file operation is noticed and synchronized to the cloud by the client software developed by the service provider.

2.2 Efficient Batched Synchronization in Dropbox-like Cloud Storage Services

As tools for personal storage, file synchronization and data sharing, cloud storage services such as Dropbox have quickly gained popularity. These services provide users with ubiquitous, reliable data storage that can be automatically synced across multiple devices, and also shared among a group of users. To minimize the network overhead, cloud storage services employ binary diff, data compression, and other mechanisms when transferring updates among users. Through comprehensive measurements and detailed analysis, we demonstrate that many cloud storage applications generate session maintenance traffic that far exceeds the useful update traffic. This behaviour as the traffic overuse problem. To address this problem, we propose the update-batched delayed synchronization (UDS) mechanism. Acting as a middleware between the user's file storage system and a cloud storage application, UDS batches updates from clients to significantly reduce the overhead caused by session maintenance traffic, while preserving the rapid file synchronization that users expect from cloud storage services.

As tools for personal storage, file synchronization and data sharing, cloud storage services such as Dropbox, Google Drive, and SkyDrive have become extremely popular. These services provide users with ubiquitous, reliable data storage that can be synchronized ("synced") across multiple devices, and also shared among a group of users. Dropbox is arguably the most popular cloud storage service, reportedly hitting more than 100 million users who store or update one billion files per day. Cloud storage services are characterized by two key components: a (front-end) client application that runs on user devices, and a (back-end) storage service that resides within the "cloud," hosting users' files in huge data centres. A user can "drop" files into or directly modify files in a special "sync folder" that is then automatically synchronized with cloud storage by the client application. Cloud storage applications typically use two

algorithms to minimize the amount of network traffic that they generate. First, the client application computes the binary diff of modified files and only sends the altered bits to the cloud. Second, all updates are compressed before they are sent to the cloud. As a simple example, if we append 100 MB of identical characters (e.g. "a") to an existing file in the Dropbox sync folder (thus the binary diff size is 100 MB), the resulting network traffic is merely 40 KB. This amount of traffic is just slightly more than the traffic incurred by appending a single byte "a" (i.e. around 38 KB, including meta-data overhead).

2.3 CloudCmp: Comparing Public Cloud Providers

While many public cloud providers offer pay-as-you-go computing, their varying approaches to infrastructure, virtualization, and software services lead to a problem of plenty. To help customers pick a cloud that fits their needs, we develop CloudCmp, a systematic comparator of the performance and cost of cloud providers. CloudCmp measures the elastic computing, persistent storage, and networking services offered by a cloud along metrics that directly reflect their impact on the performance of customer applications. CloudCmp strives to ensure fairness, representativeness, and compliance of these measurements while limiting measurement cost. Applying CloudCmp to four cloud providers that together account for most of the cloud customers today, find that their offered services vary widely in performance and costs, underscoring the need for thoughtful provider selection. From case studies on three representative cloud applications, we show that CloudCmp can guide customers in selecting the best-performing provider for their applications. Internet-based cloud computing has gained tremendous momentum in recent years. Cloud customers outsource their computation and storage to public providers and pay for the service usage on demand. Compared to the traditional computing model that uses dedicated, in-house infrastructure, cloud computing offers unprecedented advantages in terms of cost and reliability. A cloud customer need not pay a large upfront cost (e.g., for hardware purchase) before launching services, or over-provision to accommodate future or peak demand. Instead, the cloud's pay-as-you-go charging model enables the customer to pay for what she actually uses and promises to scale with demand. Moreover, the customer can avoid the cost of maintaining an IT staff to manage her server and network infrastructure.

A growing number of companies are riding this wave to provide public cloud computing services, such as Amazon, Google, Microsoft, Rackspace, and GoGrid. These cloud providers offer a variety of options in pricing, performance, and feature set. For instance, some offer platform as a service (PaaS), where a cloud customer builds applications using the APIs provided by the cloud; others offer infrastructure as a service (IaaS), where a customer runs applications inside virtual machines (VMs), using the APIs provided by their chosen guest operating systems. Cloud providers also differ in pricing models. For example, Amazon's AWS charges by the number and duration of VM instances used by a customer, while Google's AppEngine charges by the number of CPU cycles consumed by a customer's application.

The diversity of cloud providers leads to a practical question: how well does a cloud provider perform compared to the other providers? Answering this question will benefit both cloud customers and providers. For a potential customer, the answer can help it choose a provider that best fits its performance and cost needs. For instance, it may choose one provider for storage intensive applications and another for computation intensive applications. For a cloud provider, such answers can point it in the right direction for improvements. For instance, a provider should pour more resources into optimizing table storage if the performance of its store lags behind competitors.

2.4 DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds

The increasing popularity of cloud storage services has lead companies that handle critical data to think about using these services for their storage needs. Medical record databases, power system historical information and financial data are some examples of critical data that could be moved to the cloud. However, the reliability and security of data stored in the cloud still remain major concerns. 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 and deployed the system using four commercial clouds and used PlanetLab to run clients accessing the service from different countries and 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 on this scenario is twice the cost of using a single cloud, which is optimal and seems to be a reasonable cost, given the benefits.

The increasing maturity of cloud computing technology is leading many organizations to migrate their IT infrastructure and/or adapting their IT solutions to operate completely or partially in the cloud. Even governments and companies that maintain critical infrastructures (e.g., healthcare, telcos) are adopting cloud computing as a way of reducing costs. Nevertheless, cloud computing has limitations related to security and privacy, which should be accounted for, especially in the context of critical applications. A dependable and secure storage system that leverages the benefits of cloud computing by using a combination of diverse commercial clouds to build a cloud-of-clouds. In other words, DEPSKY is a virtual storage cloud, which is accessed by its users by invoking operations in several individual clouds.

SYSTEM ANALYSIS

3.1 Existing System

In existing industrial data hosting systems, data availability (and reliability) are usually guaranteed by replication or erasure coding. In the multi-cloud scenario, we also use them to meet different availability requirements, but the implementation is different. For replication, replicas are put into several clouds, and a read access is only served (unless this cloud is unavailable then) by the "cheapest" cloud that charges minimal for out-going bandwidth and GET operation. For erasure coding, data is encoded into n blocks including m data blocks and n□m coding blocks, and these blocks are put into n different clouds. In this case, though data availability can be guaranteed with lower storage space (compared with replication), a read access has to be served by multiple clouds that store the corresponding data blocks. Consequently, erasure coding cannot make full use of the cheapest cloud as what replication does. Still worse, this shortcoming will be amplified in the multi-cloud scenario where bandwidth is generally (much) more expensive than storage space.

3.2 Proposed System

The proposed CHARM scheme**.** The system proposed is a novel cost-efficient data hosting scheme with high availability in heterogeneous multi-cloud, named "CHARM". It intelligently puts data into multiple clouds with minimized monetary cost and guaranteed availability. Specifically, we combine the two widely used redundancy mechanisms, i.e., replication and erasure coding, into a uniform model to meet the required availability in the presence of different data access patterns. Next, we design an efficient heuristic-based algorithm to choose proper data storage modes (involving both clouds and redundancy mechanisms). Moreover, we implement the necessary procedure for storage mode transition (for efficiently re-distributing data) by monitoring the variations of data access patterns and pricing policies. We evaluate the performance of CHARM using both trace driven simulations and prototype experiments.

3.3 Proposed architecture

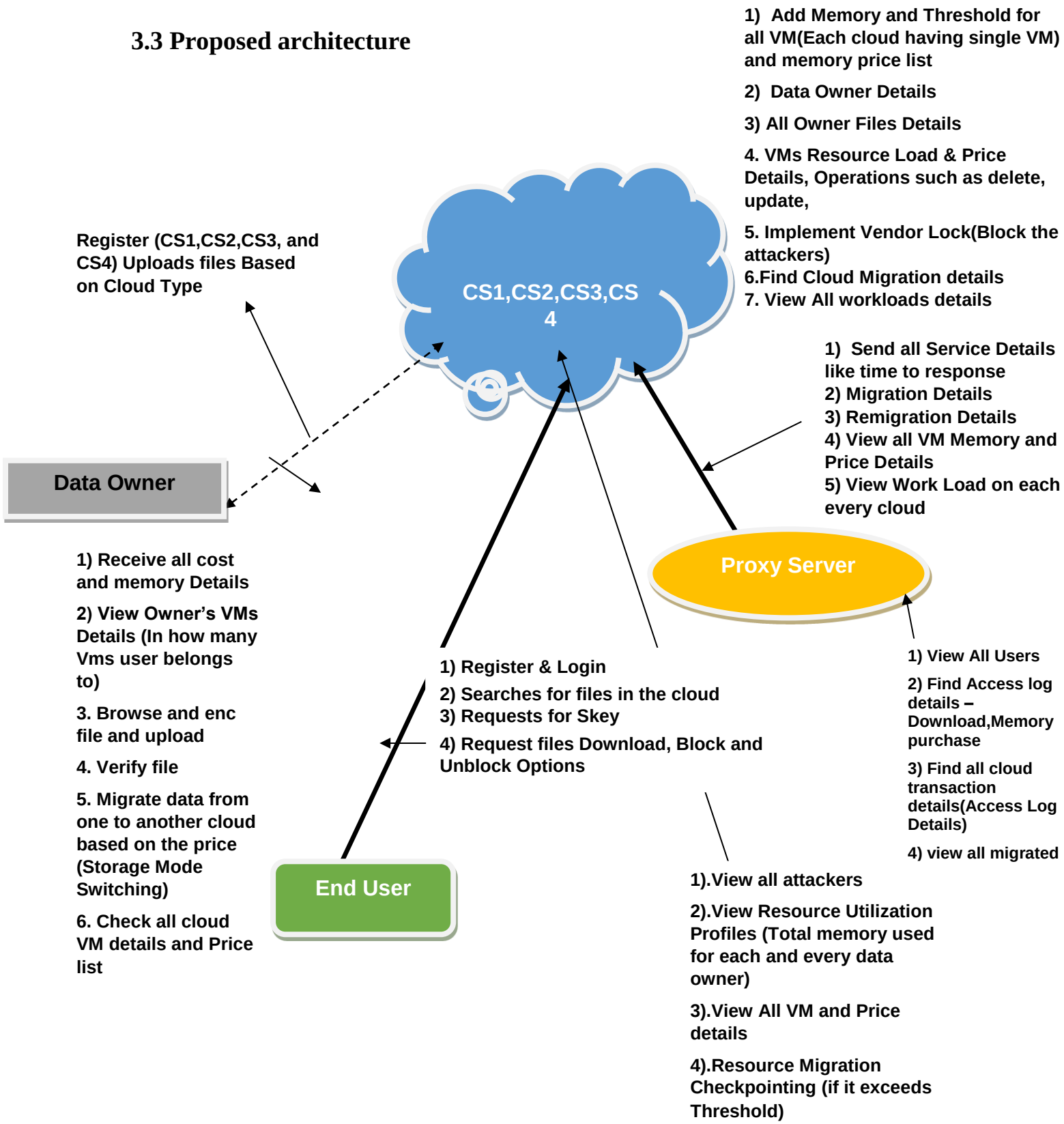

Fig.3.1: Proposed architecture of Charm

CS1 ----Rackspace

CS2 ----Amazon S3

CS3 ----Windows Azure

CS4 ----Aliyun OSS

SYSTEM REQUIREMENTS

Analysis is the process of determining the system's goal, services and constraints by consultation with system users. They are then defined in detail and serve as a system specification. In other words, this is the process of deriving the system requirements through observation of existing systems, discussion with potential users and procurer, task analysis etc. This may involve development of one or more different system models and prototypes. These help the analyst to understand the system to be specified.

Understanding the nature of the problems can be very difficult, especially if the system is new. Consequently, it is difficult to establish exactly what system should do. The descriptions of the services and constraints are the requirements for the system and process of finding out, analyzing, documenting and checking these services and constraint is called requirements engineering. The term requirement is not used throughout the software industry in a consistent way. In some cases the requirements is seen as a high-level, abstract statement of a service that the system should provide or a constraint on the system. At the other, it is a detailed, mathematically formal definition of a system function. The term user requirements means the high level, abstract requirements and system requirements means the detailed description of what the system should do. As well as these two levels of detail, a more detailed description may be produced to bridge the requirements engineering and design activities.

4.1 Hardware Requirements

- System : Pentium IV 3.5 GHz or Latest Version.
- Hard Disk : 40 GB.
- Monitor : 14’ Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 1 GB.

4.2 Software Requirements:

- Operating system : Windows XP or Windows 7, Windows 8.
- Coding Language : Java – AWT,Swings,Networking, Servlets.
- Data Base : My Sql / MS Access, SQLYog.
-
- Documentation : MS Office

- IDE : Eclipse Mars 2.1
- Development Kit : JDK 1.6

4.3 Functional Requirements

- The amount of space available in every cloud has to be calculated accurately.
- If the space available is almost full, the load from one cloud server has to be migrated to another cloud server.
- If any exception occurs, it has to be handled by the system carefully.
- The system has to check for any intruders or hackers.

4.4 Non-functional requirements

Availability : Should be available 24*7

Reliability : Ability of a system to perform its required functions under the specified conditions for a period of time.

Security : Protection from intruders such as hackers using strong encryption system.

Robustness : Ability of the system to run on any platform.

Efficiency : Resource consumption for given load.

Scalability : Ability to extend the system as required.

Response time: Time a system or functional unit takes to react to a particular input.

Performance: Amount of useful work accomplished compared to the time and resources used.

SYSTEM DESIGN

5.1 Data Flow Diagrams

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

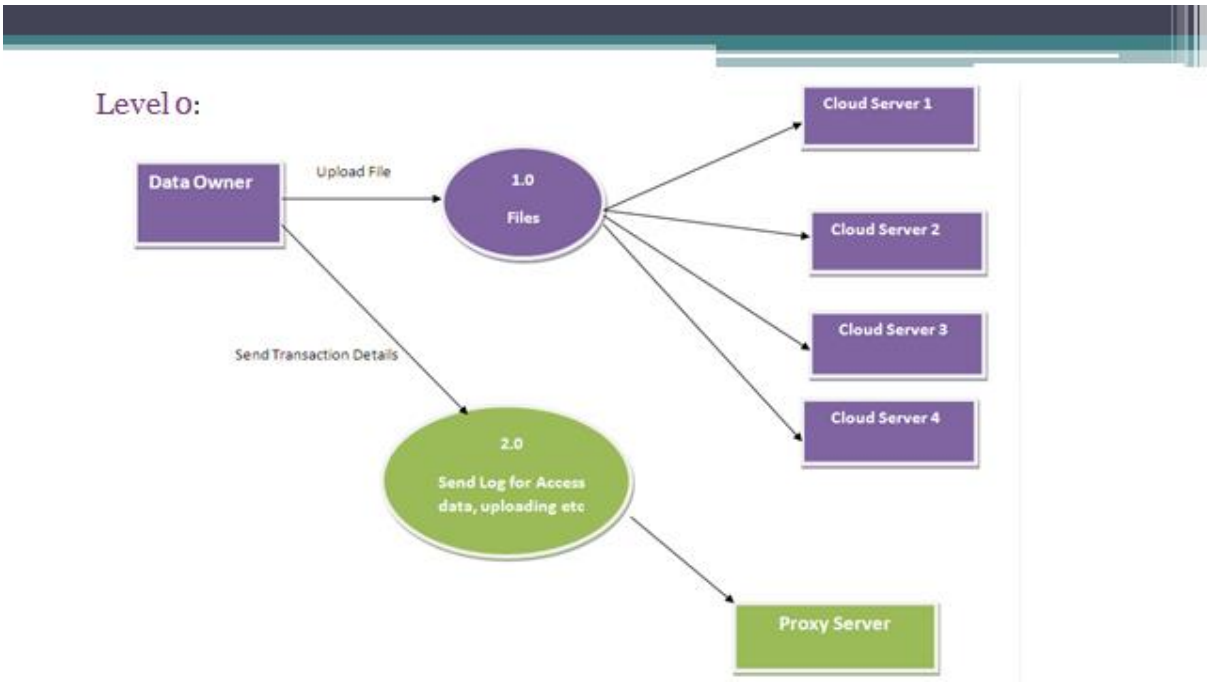

Fig. 5.1.1 : DFD Level 0 for CHARM

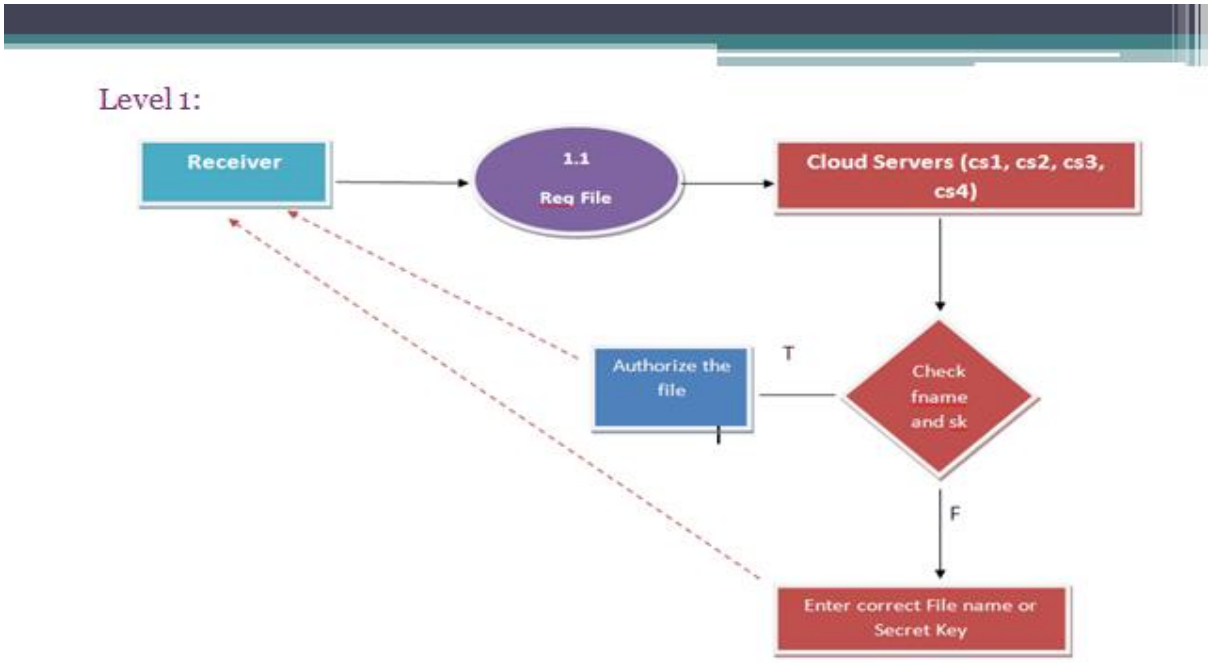

Fig.5.1.2: Level 1 DFD for CHARM

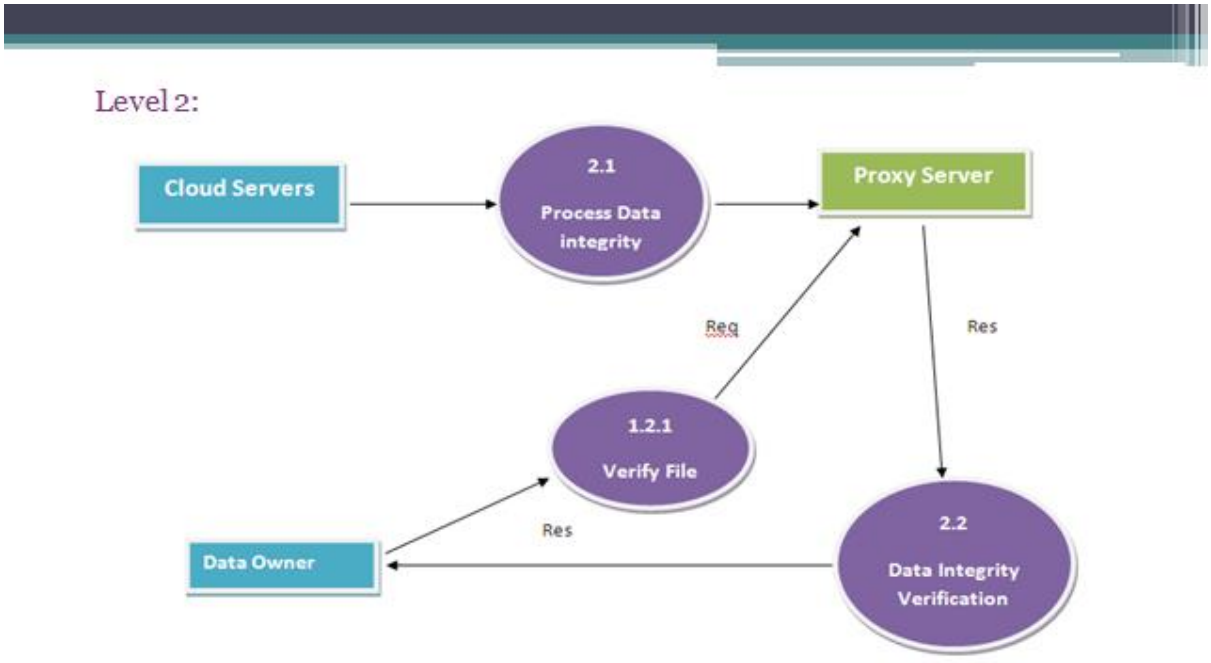

Fig.5.1.3: Level 2 DFD for CHARM

5.2 Use Case Diagram

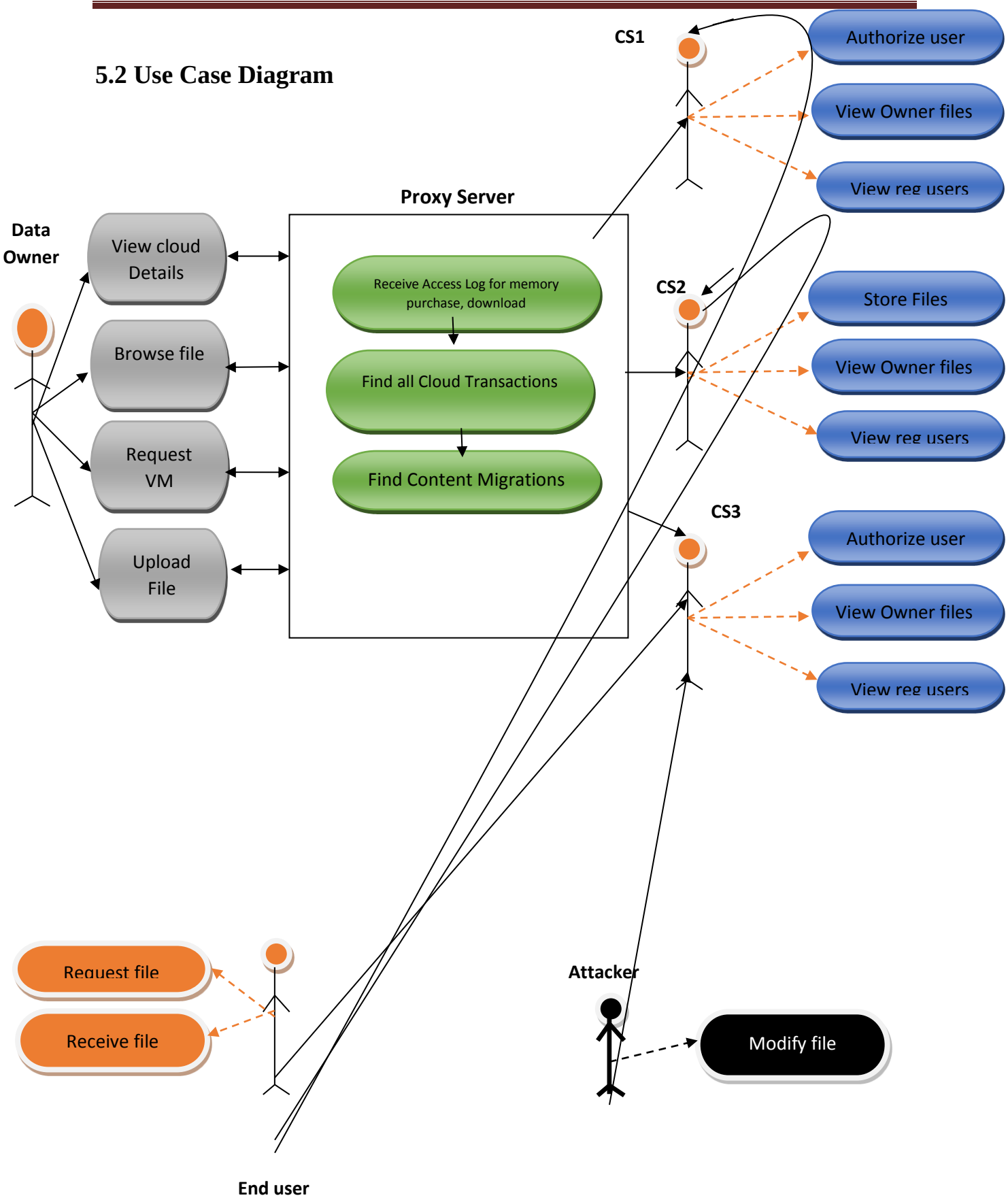

Fig.5.4: Use case diagram for CHARM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

5.3 Sequence Diagram

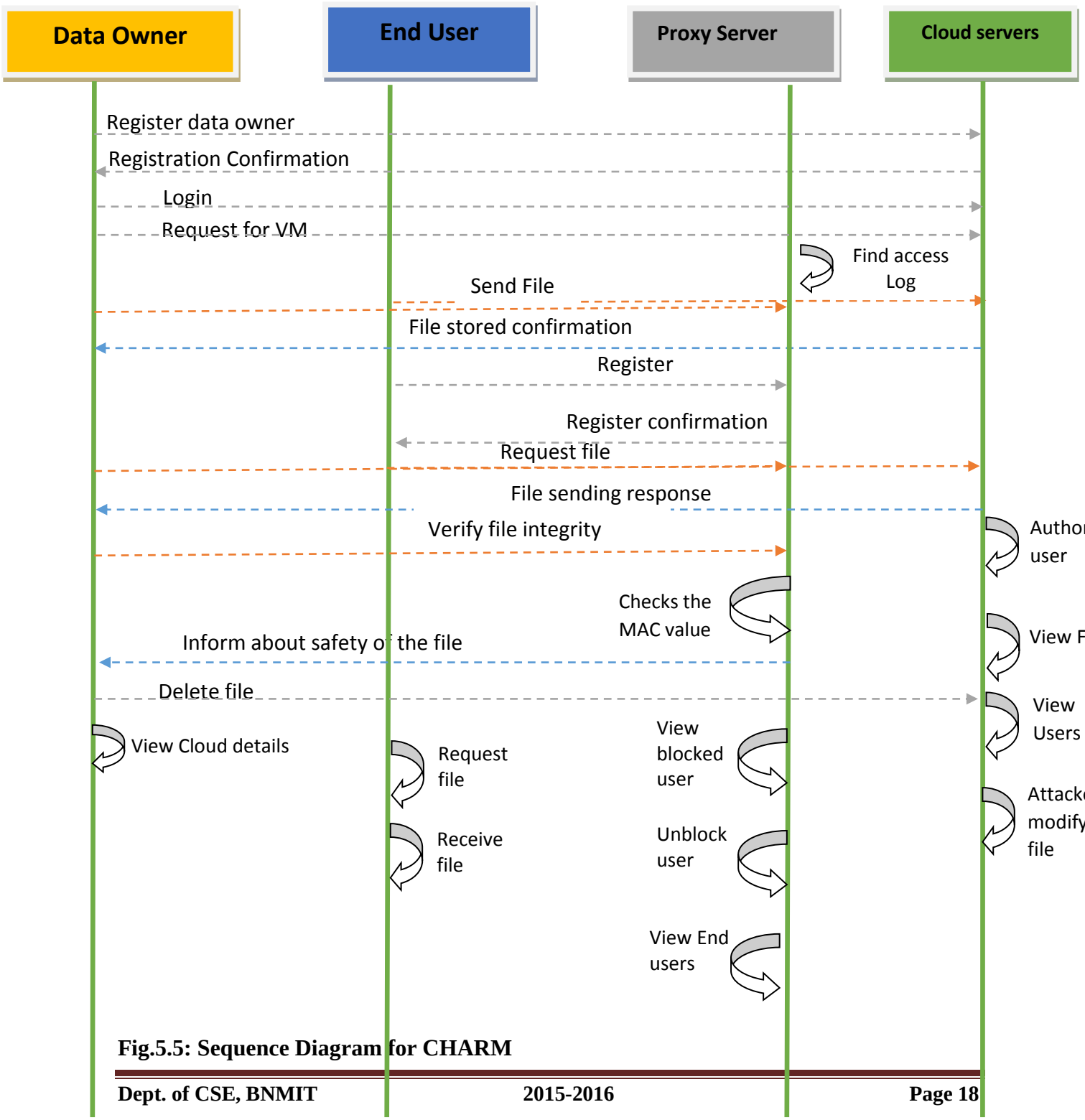

Fig.5.5: Sequence Diagram for CHARM

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

IMPLEMENTATION

6.1 Algorithm

6.1.1 Storage Mode Transition Process

The implementation of storage mode transition is as follows: the proxy gets the data from the clouds where the data is originally stored, and puts it into the newly selected clouds using new storage mode. The implementation consumes out-going bandwidth, in-going bandwidth, and several operations (i.e., GET, DELETE, and PUT). Since DELETE and in-going bandwidth are free, the transition cost T is composed of out-going bandwidth, GET, and PUT. Out-going bandwidth is more expensive thanstorage, so we have to make sure that the cost of transition can be earned back by the new storage mode.The storage mode table can be calculated in advance, because it is only affected by the available clouds, their pricing policies, and availabilities. When deciding the storage mode for each file, we use the read frequency and the size of the file to look up the table for the corresponding storage mode.

Input: the generated table Γ, the ith file's current storage mode $M[i]$, current read frequency $R[i]$, file size $S[i]$
Output: void

```
1  dSize ← the size dimension of Γ
2  dRead ← the read frequency dimension of Γ
3  for each file i do
4    for j in len(dSize) do
5      if S[i] ≥ dSize[j] then
6        dS ← j
7      else
8        break
9      end
10   end
11   for j in len(dRead) do
12     if R[i] ≥ dRead[j] then
13       dR ← j
14     else
15       break
16     end
17   end
18   if M[i] ≠ Γ[dS][dR] then
19     T ← monetary cost of transiting from M[i]
20     if M[i] > Γ[dS][dR] + T then
21       transit from M[i] to Γ[dS][dR]
22     end
23   end
24 end
```

6.1.2 Heuristic Algorithm of Data Placement

The key idea of this heuristic algorithm can be described as follows:

First assign each cloud a value which is calculated based on four factors (i.e., availability, storage, bandwidth, and operation prices) to indicate the preference of a cloud and choose the most preferred n clouds, and then heuristically exchange the cloud in the preferred set with the cloud in the complementary set to search better solution. This is similar to the idea of Kernighan-Lin heuristic algorithm, which is applied to effectively partition graphs to minimize the sum of the costs on all edges cut. The preference of a cloud is impacted by the four factors, and they have different weights. The availability is the higher the better, and the price is the lower the better.

Algorithm 1: Heuristic algorithm of data placement

Input: file size S, read frequency c_r, n's upper limit ξ
Output: minimal cost C_{sm}, the set ψ of the selected clouds

```
C_sm ← inf; ψ ← {}
L_s ← sort clouds by normalized αa_i + β/P_i from high to slow
for n = 2 to ξ do
    G_s ← the first n clouds of L_s
    G_c ← L_s − G_s
    for m = 1 to n do
        A_cur ← calculate the availability of G_s
        if A_cur ≥ A then
            C_cur ← calculate the minimal cost
            if C_cur < C_sm then
                C_sm ← C_cur
                ψ ← G_s
            end
        else
            /*heuristically search better solution*/
            G_s ← sort G_s by a_i from low to high
            G_c ← sort G_c by P_i from low to high
            for i = 1 to n do
                flag ← 0
                for j = 1 to N − n do
                    if a_Gc[j] > a_Gs[i] then
                        swap G_s[i] and G_c[j]
                        flag ← 1
                        break
                    end
                end
                if flag = 0 then
                    break
                end
                A_cur ← calculate the availability of G_s
                if A_cur ≥ A then
                    C_cur ← calculate the minimal cost
                    if C_cur < C_sm then
                        C_sm ← C_cur
                        ψ ← G_s
                    end
                    break
                end
            end
        end
0 in    end
```

6.2 High-level Code (Main Function)

6.2.1 Owner Module

Owner module is to upload their files using some access policy. First they get the public key for particular upload file after getting this public key owner request the secret key for particular upload file. Using that secret key owner upload their file.

Code Snippet:

```
try

{

MultipartRequest multi=new MultipartRequest(request, dirName, 10 * 1024 * 1024);
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( "location")) { location=multi.getParameter(paramname); }
if(paramname.equalsIgnoreCase( "cname")) { cloud=multi.getParameter(paramname); }
if(paramname.equalsIgnoreCase( "pic")) { image=multi.getParameter(paramname);} }

int f=0 ;

Enumeration files=multi.getFileNames();
```

```
while (files.hasMoreElements()) {
param name=( String) files.nextElement();
if(paramname.equals( "d1")) { paramname=n ull; }
if(paramname !=n ull) { 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;
FileInputStreamfis=new FileInputStream(ss); StringBuffer sb1=new StringBuffer(); inti=0 ;
while ((i=f is.read()) !=- 1)
{
if (i !=- 1)
{
//System.out.println(i); String hex=Integer.toHexString(i); // session.put( "hex",hex);
sb1.append(hex); // sb1.append( ",");
String binFragment="" ;
intiHex;
for(inthex.length();i1++)
{
iHex=Integer.parseInt( ""+hex.charAt(i1),16); binFragment=Integer.toBinaryString(iHex);
while(binFragment.length() < 4)
{
binFragment="0" + binFragment;
}
```

```
bin +=b inFragment; //System.out.print(bin);

} } } } }

FileInputStream fs1=null; //name=dirName+ "\\Gallery\\"+image;

intlyke=0; //String as="0" ;

//image=image.replace( ".", "_b.");

String query1="select * from charm_owner  where name='" +uname+ "'"; Statement
st1=connection.createStatement(); ResultSet rs1=st1.executeQuery(query1);

if ( rs1.next() ) {

out.print("User Name Already Exist");<% }
else

{
PreparedStatementps=connection.prepareStatement( "insert into
charm_owner(name,pass,email,mobile,addr,dob,location,cloudname,imagess)
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,location);
ps.setString(8,cloud); ps.setBinaryStream(9, (InputStream)fs, (int)(file1.length())); //
ps.setString(11,sk); if(f==0 ) ps.setObject(9,null); else if(f==1 2) {
fs1=(FileInputStream)list.get(0);

ps.setBinaryStream(9,fs1,fs1.available());

 }

int x=ps.executeUpdate();

if(x>0)

{

request.setAttribute("msg","successfull");

out.print("Registered Successfully"); %>

<% } } } catch (Exception e)
```

```
{
out.println(e.getMessage());
e.printStackTrace();
} %>
```

6.2.2User Module

This module is used to help the client to search the file using the file id and file name. If the file id and name is incorrect means the user does not get the file, otherwise server ask the secret key and get the encryption file. If the user wants the decryption file means user have the secret key.

Code Snippet:

```
   try{
application.setAttribute("cloudName",clname);
%>
<h2><%=clname  %></h2>
<%
session.setAttribute("cnames",clname);
String sql="SELECT * FROM charm_cloud where username='"+name+"' and
password='"+pass+"' and cname='"+clname+"'";
Statement stmt = connection.createStatement();
ResultSetrs =stmt.executeQuery(sql);
String utype="";
if(rs.next()){
if(clname.equalsIgnoreCase("Rackspace")){
application.setAttribute("cnames1",clname);
response.sendRedirect("CloudServerMain.jsp");
}else if(clname.equalsIgnoreCase("Amazon S3")){
application.setAttribute("cnames2",clname);
response.sendRedirect("CloudServerMain.jsp");
}else if(clname.equalsIgnoreCase("Windows Azure")){
application.setAttribute("cnames3",clname);
```

```
response.sendRedirect("CloudServerMain.jsp");
}else if(clname.equalsIgnoreCase("Aliyun OSS")){
application.setAttribute("cnames4",clname);
response.sendRedirect("CloudServerMain.jsp");
}
response.sendRedirect("CloudServerMain.jsp");
%>
<form action="CloudServerMain.jsp" method="post" id="clouddet">
<input id="cname" name="cname" width="300" type="text" value="<%=clname %>" />
</form>
<%
}
else
{
response.sendRedirect("wronglogin.html");
}
}
catch(Exception e)
{
out.print(e.getMessage());
e.printStackTrace();
}
%>
```

6.2.3 Cloud Server

Cloud storage services have become increasingly popular. Because of the importance of privacy, many cloud storage encryption schemes have been proposed to protect data from those who do not have access. All such schemes assumed that cloud storage providers are safe and cannot be hacked; however, in practice, some authorities (i.e., coercers) may force cloud storage providers to reveal user secrets or confidential data on the cloud, thus altogether circumventing storage encryption schemes. In this paper, we present our design for a new cloud storage encryption scheme that enables cloud storage providers to create convincing fake user secrets to protect user privacy. Since coercers cannot tell if obtained secrets are true or not, the cloud storage providers ensure that user privacy is still securely protected.Most of the proposed

schemes assume cloud storage service providers or trusted third parties handling key management are trusted and cannot be hacked; however, in practice, some entities may intercept communications between users and cloud storage providers and then compel storage providers to release user secrets by using government power or other means. In this case, encrypted data are assumed to be known and storage providers are requested to release user secrets.we aimed to build an encryption scheme that could help cloud storage providers avoid this predicament. In our approach, we offer cloud storage providers means to create fake user secrets. Given such fake user secrets, outside coercers can only obtained forged data from a user's stored ciphertext. Once coercers think the received secrets are real, they will be satisfied and more importantly cloud storage providers will not have revealed any real secrets. Therefore, user privacy is still protected. This concept comes from a special kind of encryption scheme called deniable encryption.

Code Snippet:

```
<%
String name = null;
        String a = (String) application.getAttribute("cloudName");
        String s = a, usr2 = "";
        //Rackspace
//Amazon S3
//Windows Azure
//Aliyun OSS
        if (!(a.equalsIgnoreCase("Amazon S3") || a.equalsIgnoreCase("Windows Azure") || a
                        .equalsIgnoreCase("Aliyun OSS"))) {
                usr2 = a;
                application.setAttribute("ocn", usr2);
%>

<%
        }

        if (!(a.equalsIgnoreCase("Rackspace") || a.equalsIgnoreCase("Windows Azure") || a
                        .equalsIgnoreCase("Aliyun OSS"))) {
```

```
            String b=(String)application.getAttribute("cnames2");
            usr2 = b;
            application.setAttribute("ocn", usr2);
            %>
            <h2>Welcome To <%=usr2%> Control Panel</h2>

            <%
    }
    if (!(a.equalsIgnoreCase("Rackspace") || a.equalsIgnoreCase("Amazon S3") || a
                    .equalsIgnoreCase("Aliyun OSS"))) {
            String c=(String)application.getAttribute("cnames3");
            usr2 = c;
            application.setAttribute("ocn", usr2);
            %>
            <h2>Welcome To <%=usr2%> Control Panel</h2>
            <%
    }
    if (!(a.equalsIgnoreCase("Rackspace") || a.equalsIgnoreCase("Amazon S3") || a
                    .equalsIgnoreCase("Windows Azure"))) {
            String d=(String)application.getAttribute("cnames4");
            usr2 = d;
            application.setAttribute("ocn", usr2);
            %>
```

SOFTWARE 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.

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.Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

7.1 TYPES OF TESTS

7.1.1 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.

7.1.2 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.

7.1.3 Functional test

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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.

7.1.4 System Test

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

7.2Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.2.1 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.

- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.3 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.

7.4 Module wise testing

7.4.1 Data owner module

Sl No.of Test Cases	Test case 1
Name of Test	Testing the Login Credentials

Feature being Tested	Login Credentials
Description	Testing to check if the login credentials are working and the login is successful
Sample Input	User Name: sharan@gmail.com Password: sharan
Expected Output	User should be able to Login
Actual Output	Login Successful
Remarks	Test Case pass

Table 7.4.1.1 Login Verification

SlNo.of Test Cases	Test case 2
Name of Test	Owner Registration
Feature being Tested	Registration Process

Description	The process in which a user enters information to be registered as the data owner.
Sample Input	Username: sharan Password: sharan Mail id: sharan@gmail.com Mobile number: 9876543210 Address: Jayanagar Location: Banaglore Cloud Server: Aliyun OSS Profile Photo: browse and upload
Expected Output	User should be able to register and navigated to data owner page
Actual Output	Registration Successful
Remarks	Test Case pass

Table 7.4.1.2 Owner registration

SlNo.of Test Cases	Test case 3
Name of Test	Uploading a file

Feature being Tested	Encryption and uploading
Description	The file is encrypted before being uploaded.
Sample Input	Encrypt file: Select file: demo.txt File Name: demo.txt Upload file: File name: demo.txt Select cloud server: Ailyun OSS
Expected Output	The data must be successfully uploaded and MAC address of cloud server is displayed.
Actual Output	Data uploaded successful
Remarks	Test Case pass

Table 7.4.1.3 Encryption and Upload

SlNo.of Test Cases	Test case 4
Name of Test	Migration of file
Feature being Tested	Migration process

Description	Moving of a file from one cloud to another cloud to main the load
Sample Input	Provide your Email id: sharan@gamil.com Provide your reg Username: sharan Migrating from : Alliyun OSS Migrating to: Amazon S3
Expected Output	Migration should be successful.
Actual Output	You are not registered on Amazon S3
Remarks	Test Case fail Failed because we are not registered to Amazon S3

Table 7.4.1.4 Migration Process

SlNo.of Test Cases	Test case 5
Name of Test	Verify file
Feature being Tested	Verification process
Description	Checking to see if the uploaded file is protected

Sample Input	File Name: demo.txt Select cloud server: Aliyun OSS
Expected Output	Uploaded file should be secure
Actual Output	Demo.txt in Aliyun OSS is secure
Remarks	Test Case pass

Table 7.4.1.5 Verification Process

7.4.2 Cloud Server module

Sl No. of Test Cases	Test case 1
Name of Test	Logging in to the server
Feature being Tested	Login Credentials
Description	Checking if the login to the cloud server is successful or not.
Sample Input	Name:Cloud Password : ***** Cloud Server: Aliyun OSS

Expected Output	Should be able to login properly
Actual Output	Successfully logged in
Remarks	Test Case pass

Table 7.4.2.1Login Verification

Sl No. of Test Cases	Test case 2
Name of Test	Obtaining the Threshold details
Feature being Tested	Capacity of the server
Description	Should be able to receive threshold details
Sample Input	Click on Threshold->Get threshold details Select Cloud server: Aliyun OSS
Expected Output	Should be able to receive Threshold Details

Actual Output	Successfully obtained Threshold Details
Remarks	Test Case pass

Table 7.4.2.2Threshold Details

Sl No. of Test Cases	Test case 3
Name of Test	Updating cost and memory
Feature being Tested	Cost and memory capacity of cloud server
Description	update cost and memory
Sample Input	Cloud Name: Aliyun OSS Virtual Memory: 10000
Expected Output	Should be able to Redirect to another page and update cost and memory
Actual Output	Successfully updated the cost and memory

Remarks	Test Case pass

Table 7.4.2.3 Cost and Memory capacity

Sl No. of Test Cases	Test case 4
Name of Test	Migrating the cloud
Feature being Tested	Capacity of the server
Description	Checking the migration details to migrate the cloud
Sample Input	Click on Migrate cloud
Expected Output	Migration details should be available
Actual Output	Successfully obtained Migration details
Remarks	Test Case pass

Table 7.4.2.4 Migration and Capacity of the Server

Sl No. of Test Cases	Test case 5
Name of Test	Obtaining the Data owner Details
Feature being Tested	Obtaining details of owners
Description	Should be able to receive Data owner details
Sample Input	Click on View Data owners
Expected Output	Should be able to receive Data owner Details
Actual Output	Successfully obtained Data owner Details
Remarks	Test Case pass

Table 7.4.2.5 Data Owner details

Sl No. of Test Cases	Test case 6
Name of Test	Testing whether uploaded file present on cloud
Feature being Tested	Existence of file on cloud along with details

Description	Testing to check if the files being uploaded on cloud are available along with details
Sample Input	Click on view all files
Expected Output	Files on cloud with details should be available
Actual Output	Files with details available
Remarks	Test Case pass

Table 7.4.2.6 Existence of uploaded file

Sl No. of Test Cases	Test case 7
Name of Test	To check Migration utilization
Feature being Tested	Work load details
Description	Checking if Work Load details are available
Sample Input	User Name: sharan

Expected Output	Work Load details should be available
Actual Output	Work load details available
Remarks	Test Case pass

Table 7.4.2.7 Work Load details

Sl No. of Test Cases	Test case 8
Name of Test	Obtaining the Attacker details
Feature being Tested	Attacker details
Description	Should be able to receive attacker details
Sample Input	Click on view attacker
Expected Output	Should be able to receive Attacker Details

Actual Output	Details Unavailable
Remarks	Test Case Fail

Table 7.4.2.8 Attacker Details

Sl No. of Test Cases	Test case 9
Name of Test	Unrevoke user
Feature being Tested	Unrevoking vendor
Description	Should be able to unrevoked user
Sample Input	Revoked user name:neo
Expected Output	Should unrevoked user
Actual Output	Successfully unrevoked user

Remarks	Test Case pass

Table 7.4.2.9 Unrevoking User

Sl No. of Test Cases	Test case 10
Name of Test	Log out cloud server
Feature being Tested	Logging out of cloud server
Description	Should log out of cloud server
Sample Input	Click on log out
Expected Output	Should log out successfully
Actual Output	Successfully logged out
Remarks	Test Case pass

Table 7.4.2.10 User Logout

7.4.3 Proxy Server module

Sl No. of Test Cases	Test case 1
Name of Test	Testing the Login Credentials
Feature being Tested	Login Credentials
Description	Testing to check if the login credentials are working and the login is successful
Sample Input	User ID: Proxy Password: Proxy
Expected Output	Should be able to Login
Actual Output	Login Successful

Remarks	Test Case pass

Table 7.4.3.1 Login Verification

Sl No. of Test Cases	Test case 2
Name of Test	Testing if the migration details are available
Feature being Tested	Migration
Description	Testing to check if the migration details are available and migration takes place
Sample Input	Click on migration details to obtain details regarding the migrated file
Expected Output	Migration details should be available

Actual Output	Migration details available
Remarks	Test Case pass

Table 7.4.3.2 Migration Details

Sl No. of Test Cases	Test case 3
Name of Test	Testing for VM resources
Feature being Tested	Availability of VM resources and the price associated with it
Description	Testing to check available VM resources and price details
Sample Input	Enter cloud server->Aliyun OSS Click on Get

Expected Output	VM resources and price details should be available
Actual Output	VM resources and price details are available
Remarks	Test Case pass

Table 7.4.3.3 VM Resources

Sl No. of Test Cases	Test Case 4
Name of Test	Testing for work load
Feature being Tested	Work Load details
Description	Testing to check whether the work load details are available

Sample Input	Click on view work load-> Username: Sharan Click on Get
Expected Output	Work load details should be available
Actual Output	Work load details available
Remarks	Test Case pass

Table 7.4.3.4 Work Load Details

Sl No. of Test Cases	Test Case 5
Name of Test	Testing for Upload Transaction
Feature being Tested	Upload Transaction

Description	Testing to check whether the uploaded files with details are available
Sample Input	Click on view transaction-> Select transaction: Upload Click on Get
Expected Output	Transaction details for upload should be available
Actual Output	Upload transaction details available
Remarks	Test Case pass

Table 7.4.3.5 Upload Transaction

Sl No. of Test Cases	Test Case 6
Name of Test	Testing for Download Transaction

Feature being Tested	Download Transaction
Description	Testing to check whether the downloaded files with details are available
Sample Input	Click on view transaction-> Select transaction: Download Click on Get
Expected Output	Transaction details for download should be available
Actual Output	Download transaction details available
Remarks	Test Case pass

Table 7.4.3.6 Download Transaction

Sl No. of Test Cases	Test Case 7

Name of Test	Testing for Migrated Transaction
Feature being Tested	Migrated Transaction
Description	Testing to check whether the migrated files with details are available
Sample Input	Click on view transaction-> Select transaction: Upload Click on Get
Expected Output	Transaction details for upload should be available
Actual Output	Upload transaction details available
Remarks	Test Case pass

Table 7.4.3.7 Migration Transaction

Sl No. of Test Cases	Test Case 8
Name of Test	Testing for Proxy Files
Feature being Tested	Proxy File details
Description	Testing to check whether the proxy files with details are available
Sample Input	Click on view proxy files-> List of proxy files with details are shown
Expected Output	List of proxy files should be available
Actual Output	Proxy files available
Remarks	Test Case pass

Table 7.4.3.8 Proxy File details

7.4.4 End User module

Sl No. of Test Cases	Test case 1
Name of Test	End user login
Feature being Tested	Log in credentials of end user
Description	Check the authenticity of end user
Sample Input	Enter user name and password Name: neo Password: *** Select cloud Server: Windows Azure
Expected Output	Should be able to login
Actual Output	Successfully logged in.
Remarks	Test Case pass

Table 7.4.4.1 Login Verification

Sl No. of Test Cases	Test case 2
Name of Test	Testing for availability of file
Feature being Tested	Availability

Description	Should be able to receive File details
Sample Input	Enter the filename to be searched: demo.txt
Expected Output	Should be able to view the file
Actual Output	Successfully viewed the file demo.txt
Remarks	Test Case pass

Table 7.4.4.2 File Availability

Sl No. of Test Cases	Test case 3
Name of Test	Checking for a particular file and send request to data owner
Feature being Tested	Availability of a file
Description	Should be able to send Data owner the details of request
Sample Input	Click on Filename and enter Demo.txt

Expected Output	Should be able to send request to the data owner
Actual Output	Successfully sent the request to data owner
Remarks	Test Case pass

Table 7.4.4.3 File Availability and Request

Sl No. of Test Cases	Test case 4
Name of Test	Downloading files from Server
Feature being Tested	Ability to download
Description	Should be able download the mentioned file
Sample Input	Enter the details to download the file Filename: demo.txt Enter Owner name: Sharan MAC: 8 80000 7032,69828
Expected Output	File should be downloaded successfully.

Actual Output	Successfully downloaded the file
Remarks	Test Case pass

Table 7.4.4.3 Downloading Files

RESULTS

8.1 Data Owner module

Testing to check if the login credentials are working and the login is successful

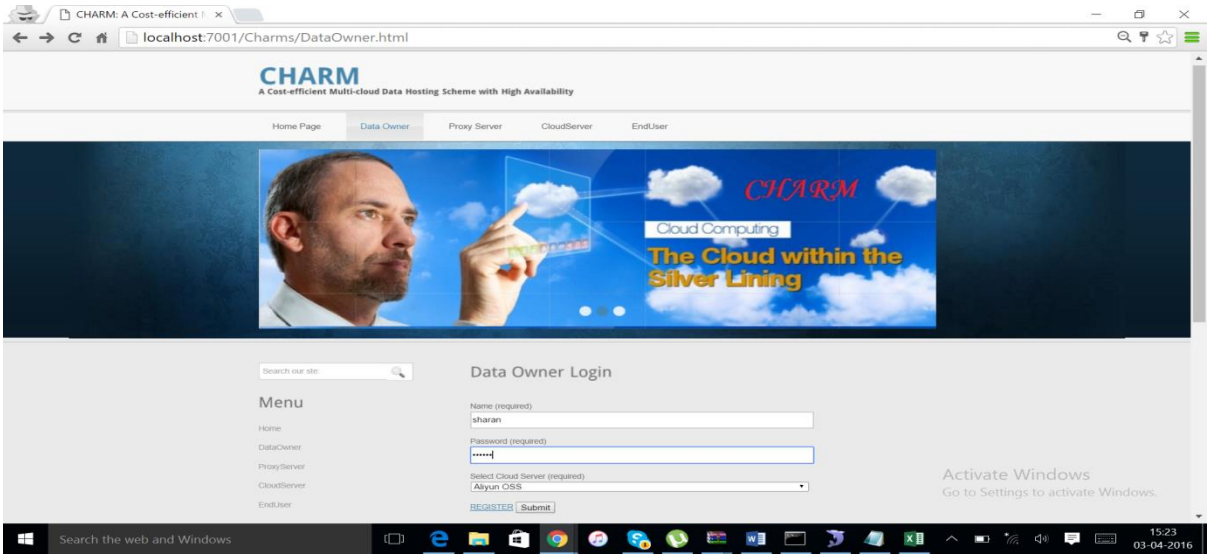

Fig.8.1.1 Login Verification

The process in which a user enters information to be registered as the data owner.

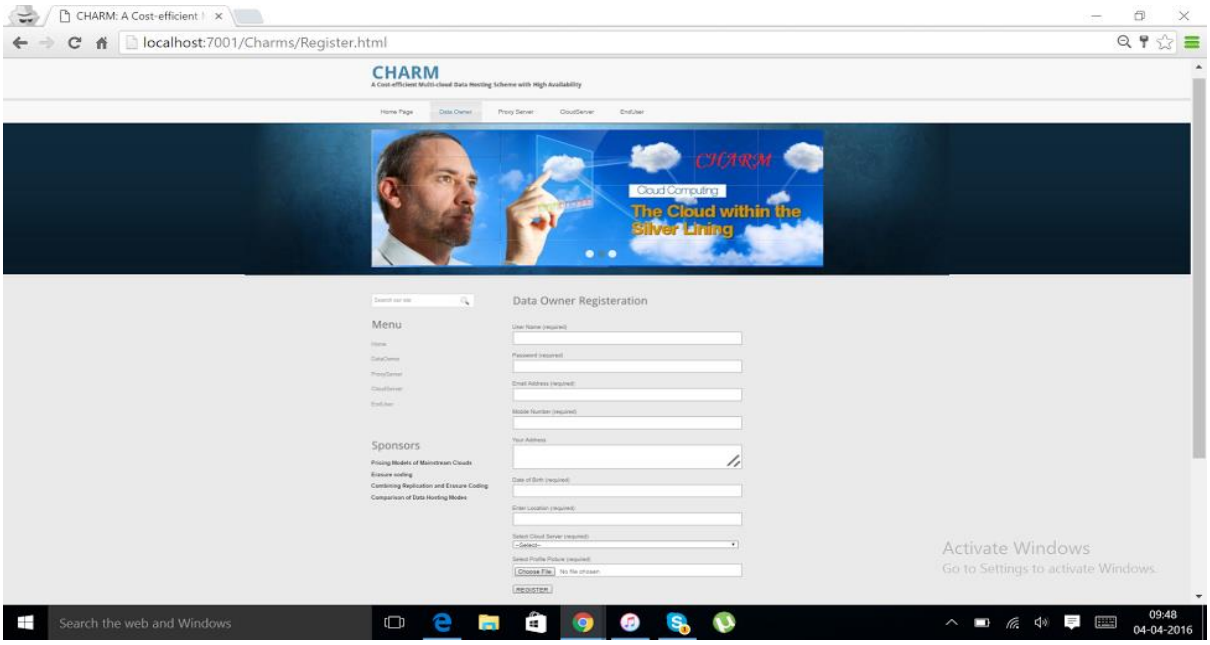

Fig 8.1.2 Owner registration

The file is encrypted as shown in the figure below

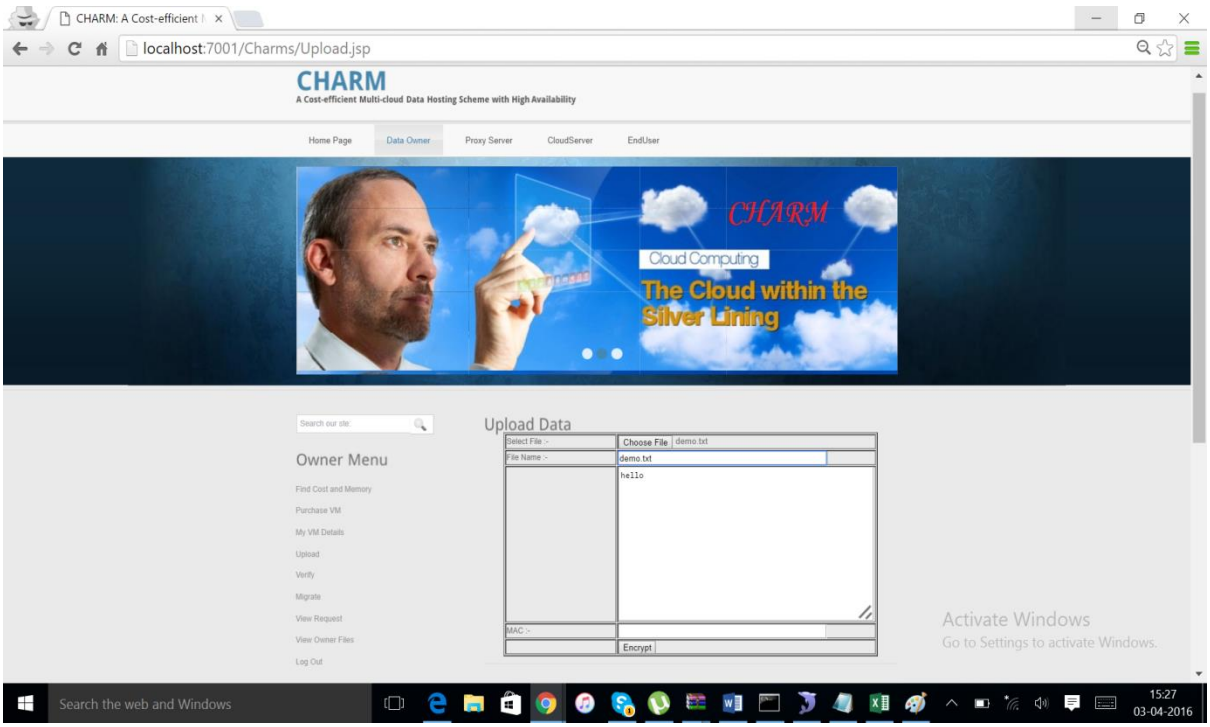

Fig 8.1.3 File Encryption

The encrypted file is then uploaded

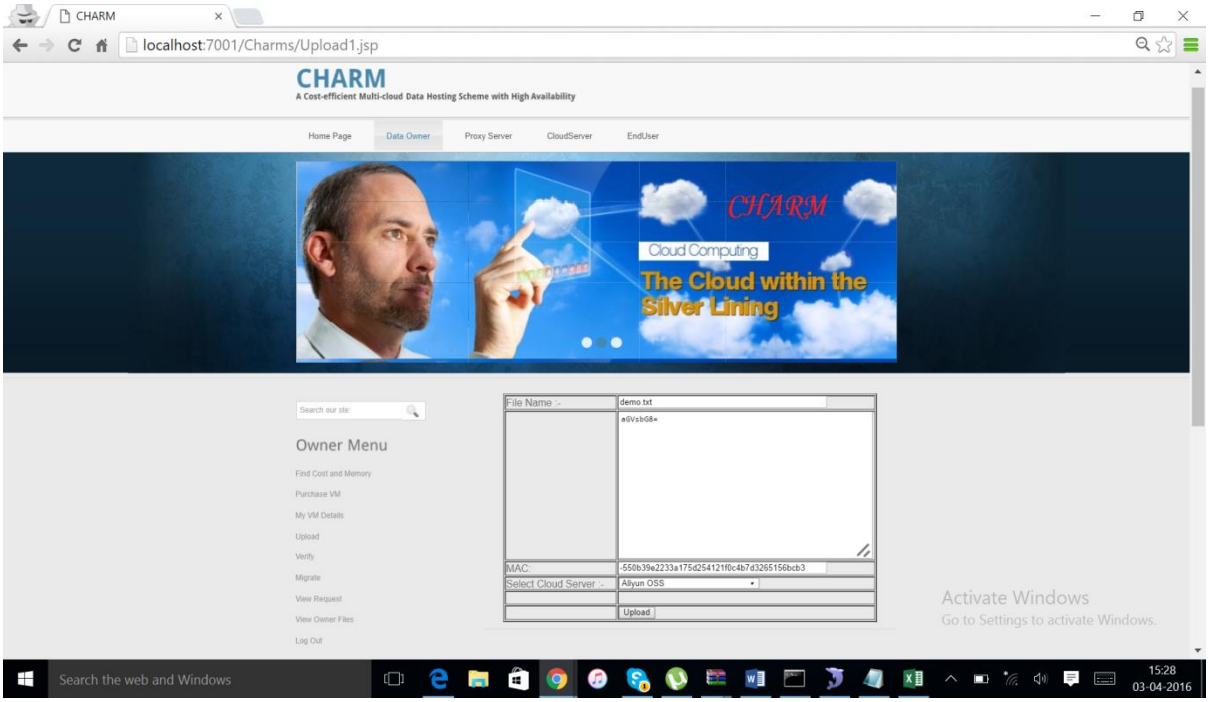

Fig 8.1.4 File Upload

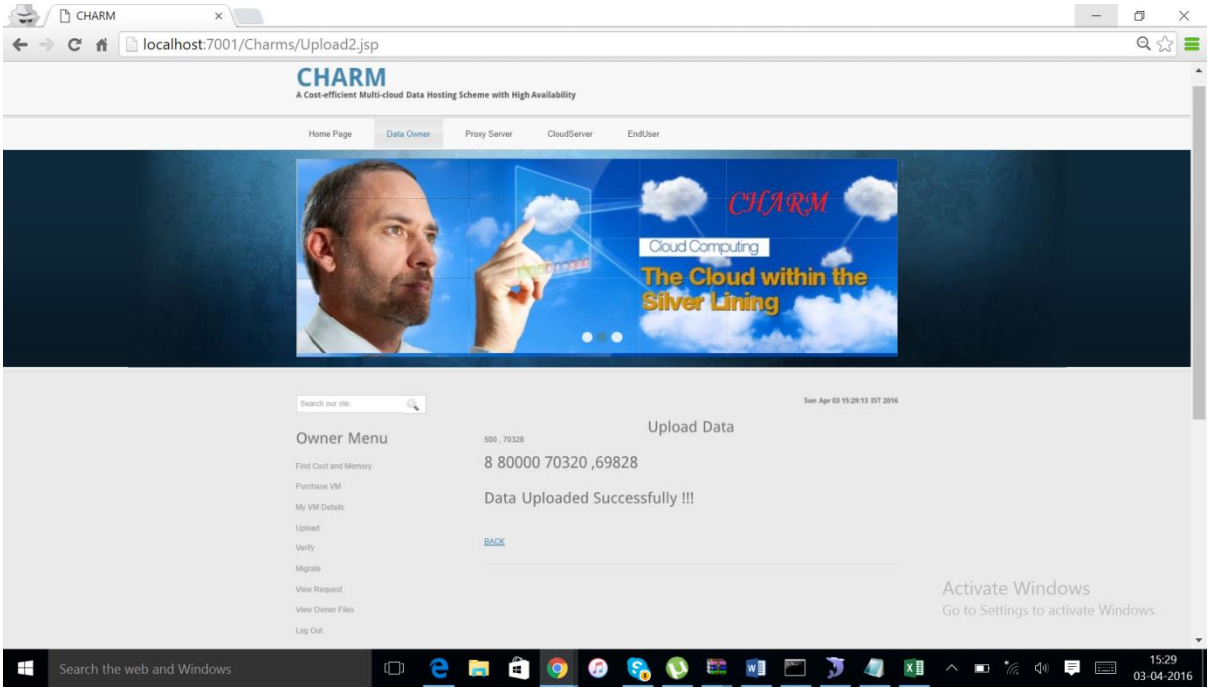

Fig 8.1.5 Upload Successful

Moving of a file from one cloud to another cloud to main the load

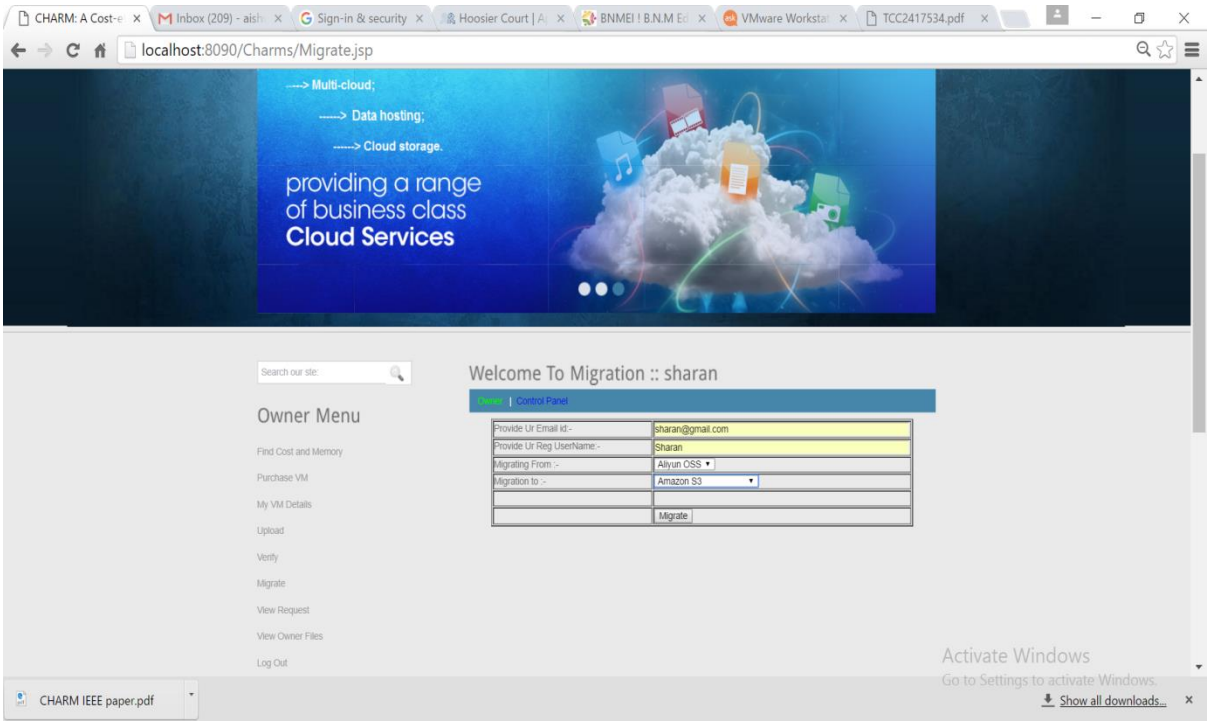

Fig 8.1.6 Migration Process

Checking to see if the uploaded file is protected

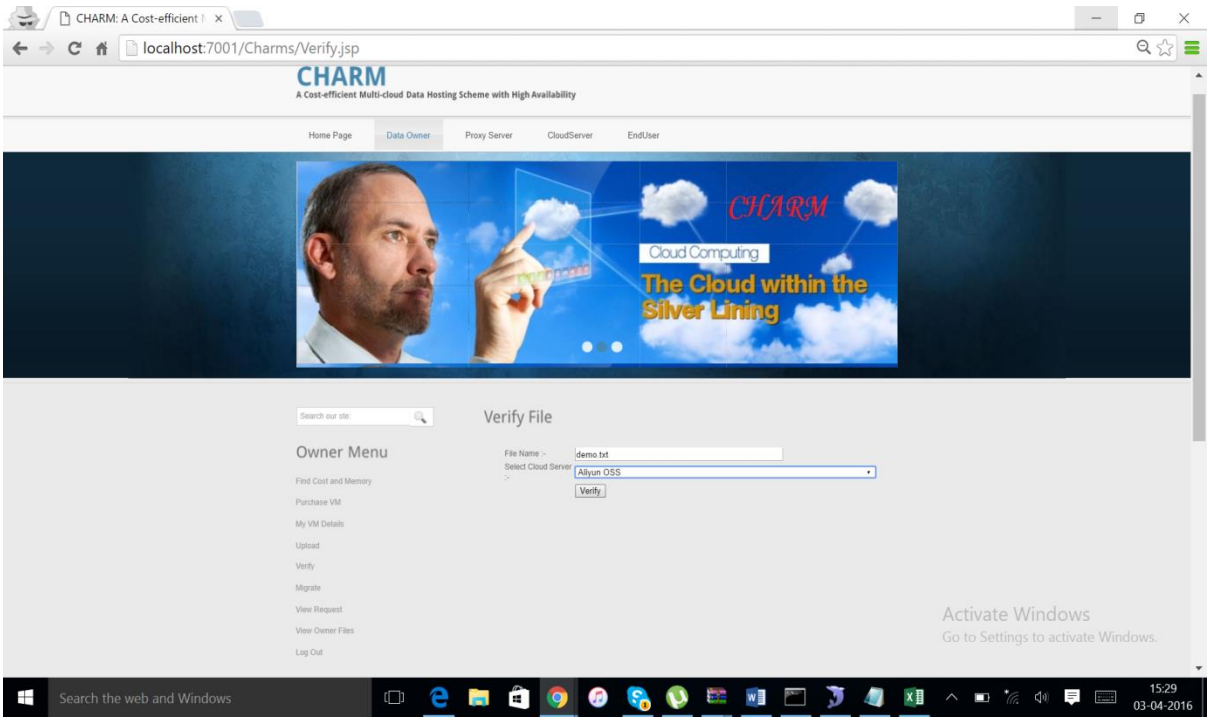

Fig 8.1.7 Verification Process

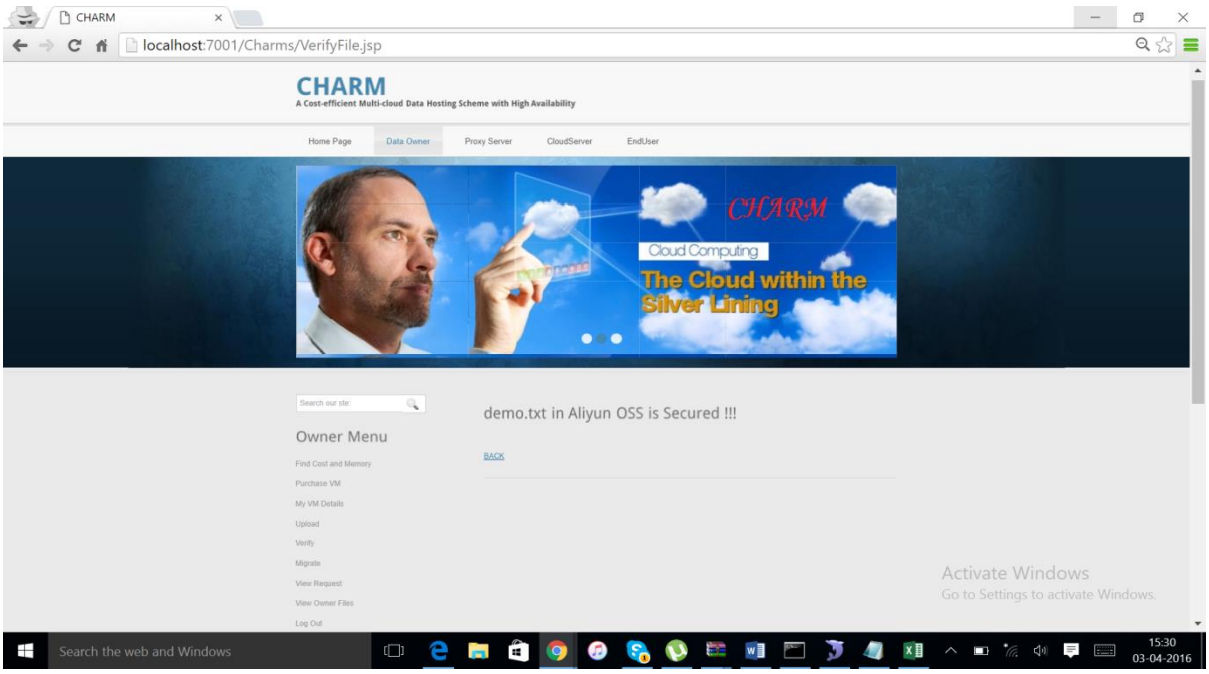

Fig 8.1.8 Verification Successful

8.2 Cloud Server

The login page to the cloud server is as shown below.

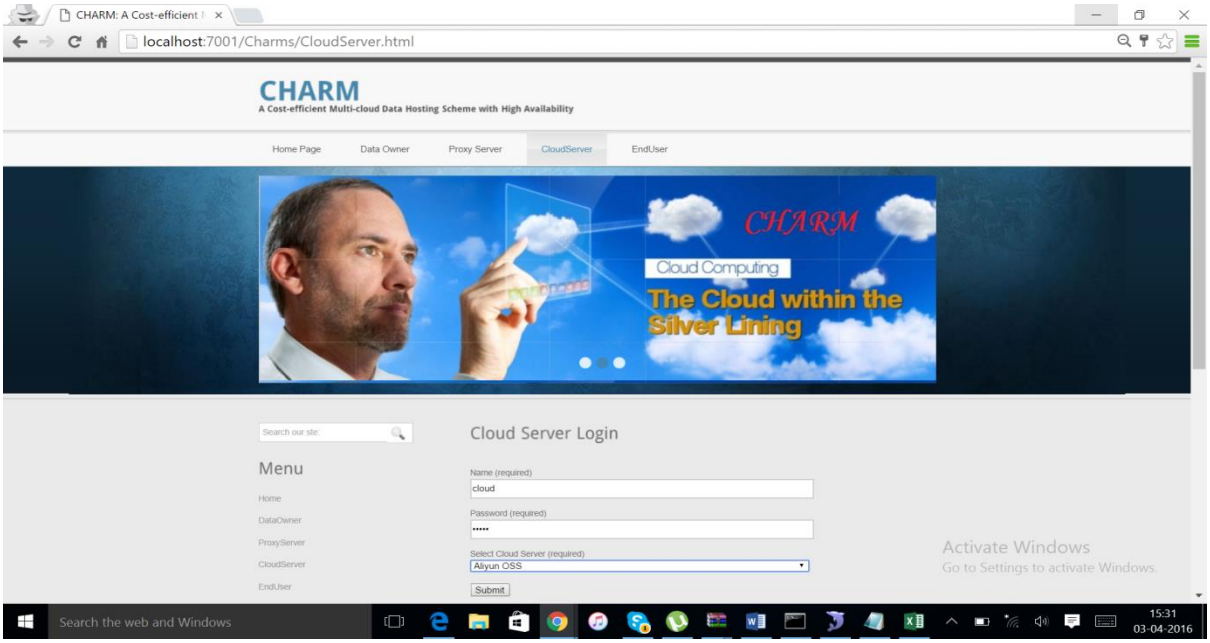

Fig 8.2.1 Cloud Server login verification

Updating cost and memory

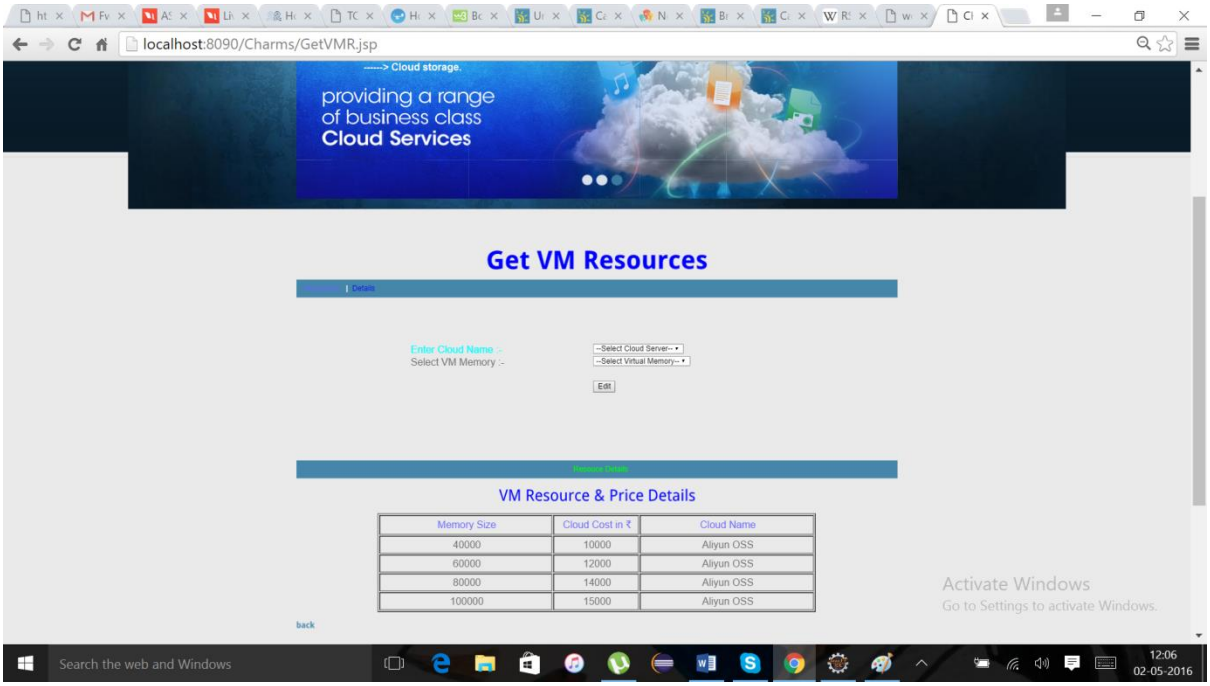

Fig 8.2.2 cost and memory capacity

Obtaining the Threshold details to verify the capacity of the server

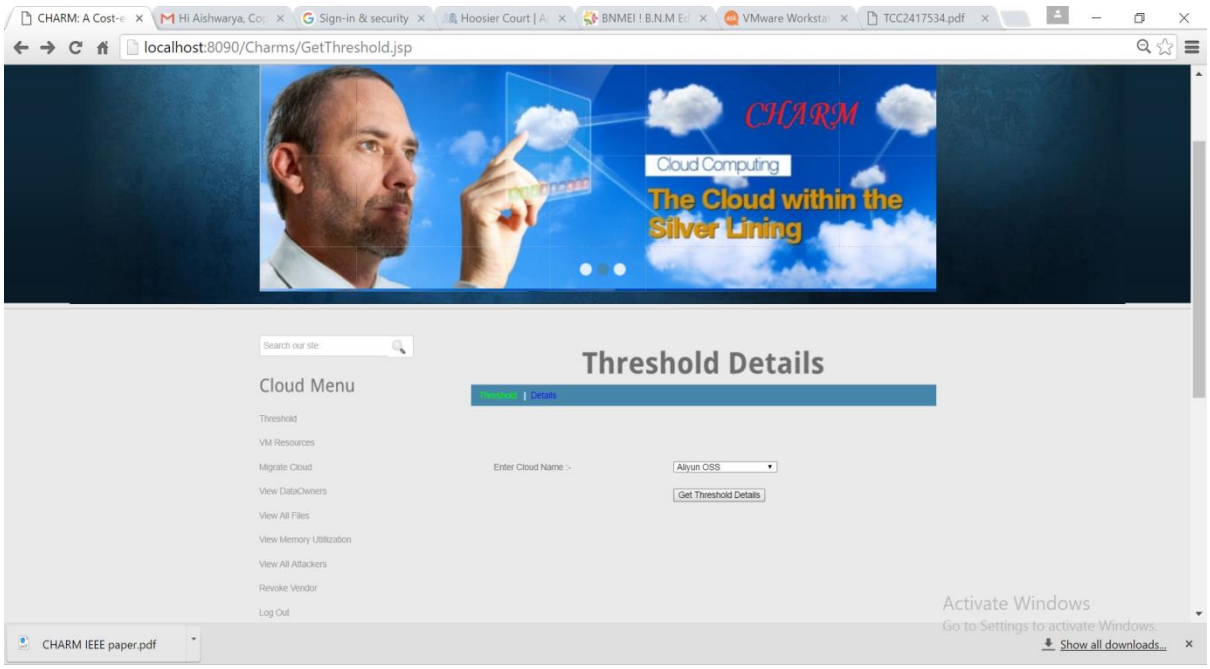

Fig 8.2.3 Threshold Details

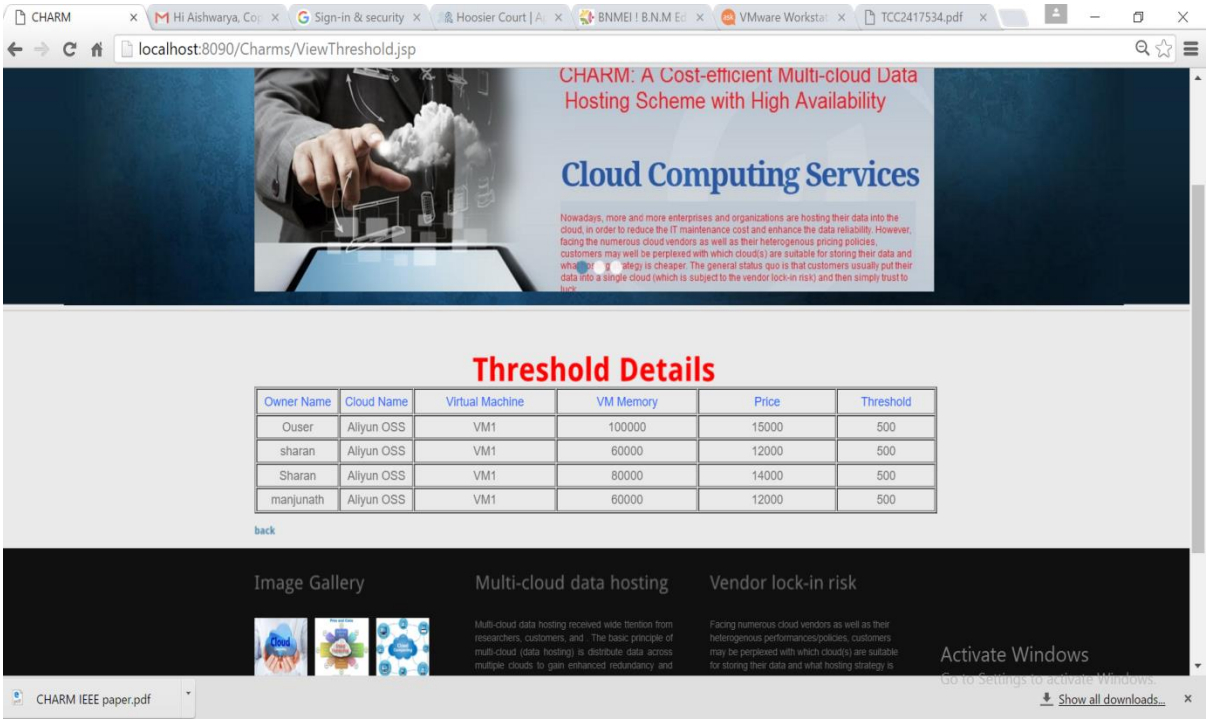

Fig 8.2.4 Threshold details obtained successfully

Checking the migration details to migrate the cloud

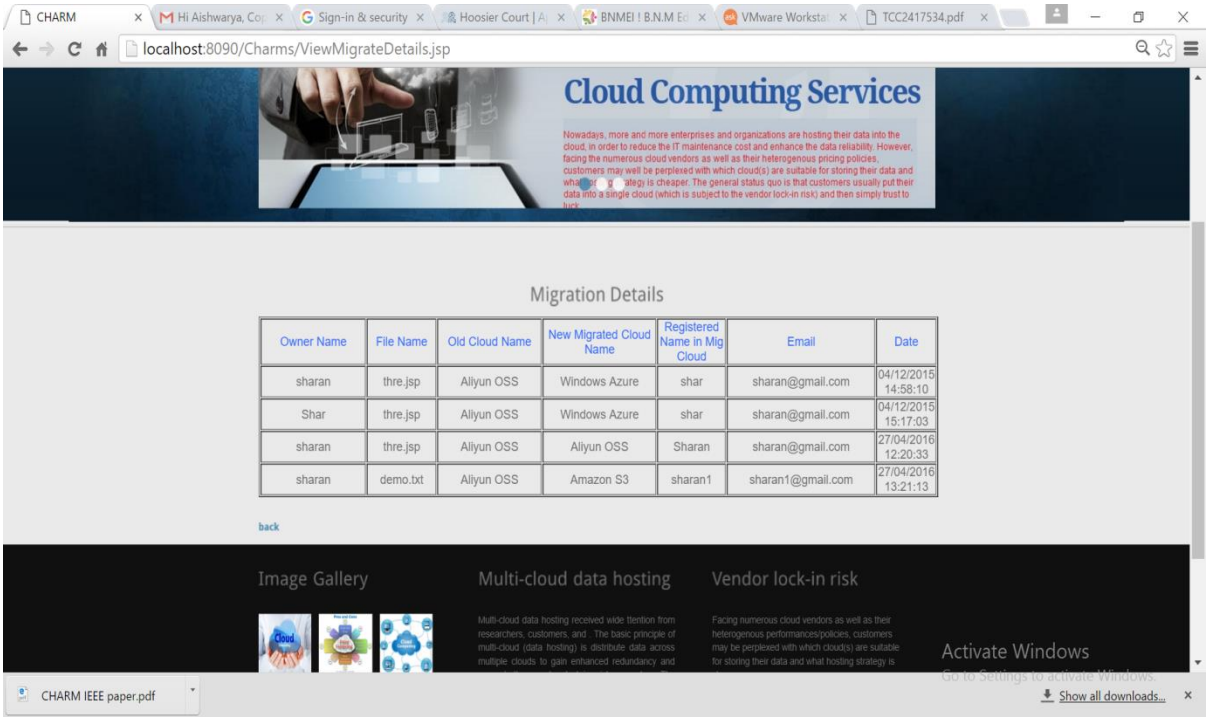

Fig 8.2.5 Migration and Capacity of the Server

Obtaining the Data Owner Details

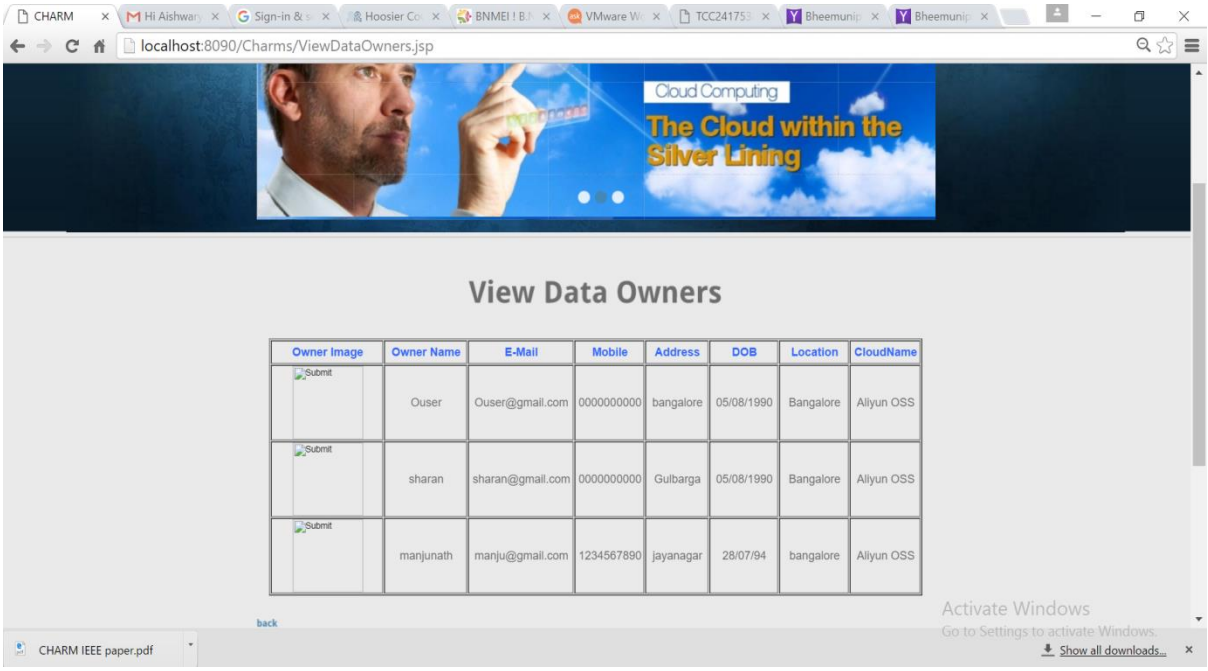

Fig 8.2.6 Data Owner details

Viewing all the files present on the cloud server along with their details

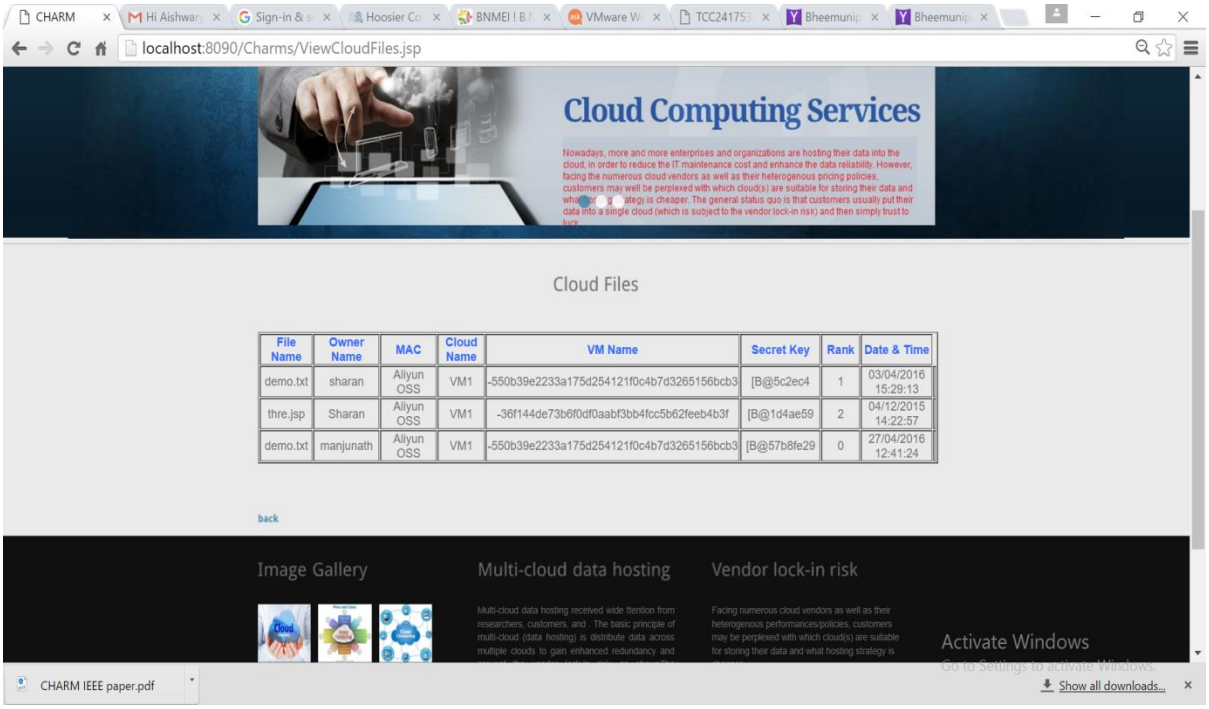

Fig 8.2.7 Existence of uploaded file

Checking if Work Load details are available

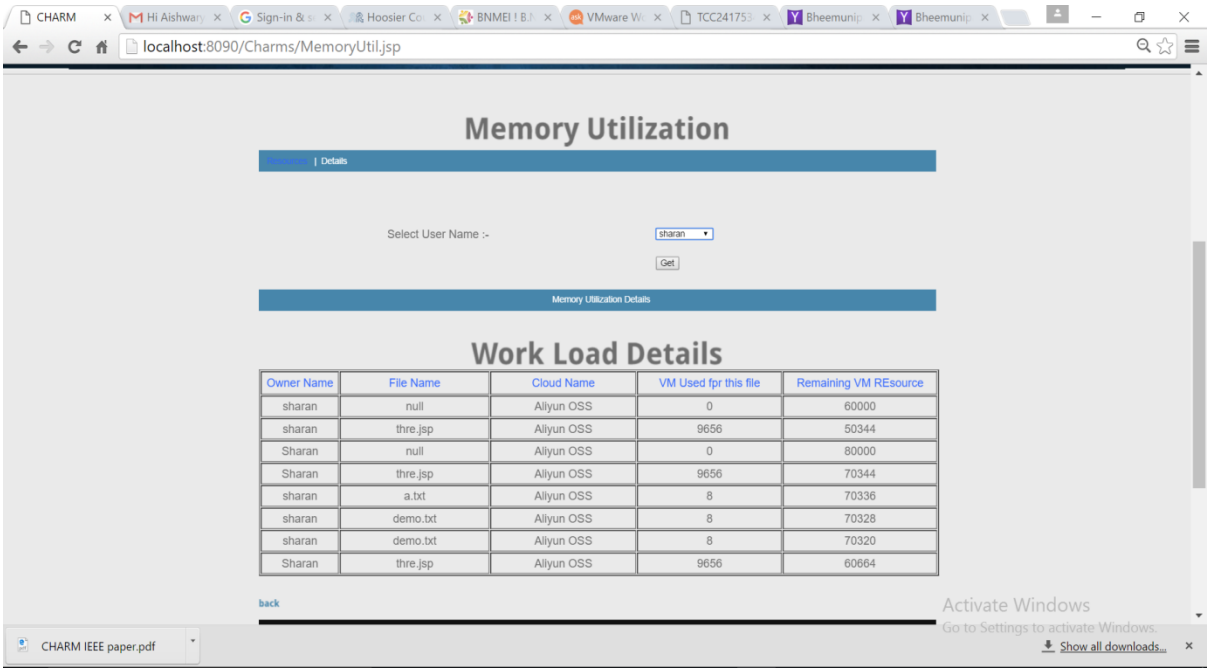

Fig 8.2.8 Work load details

Unrevoking vendor

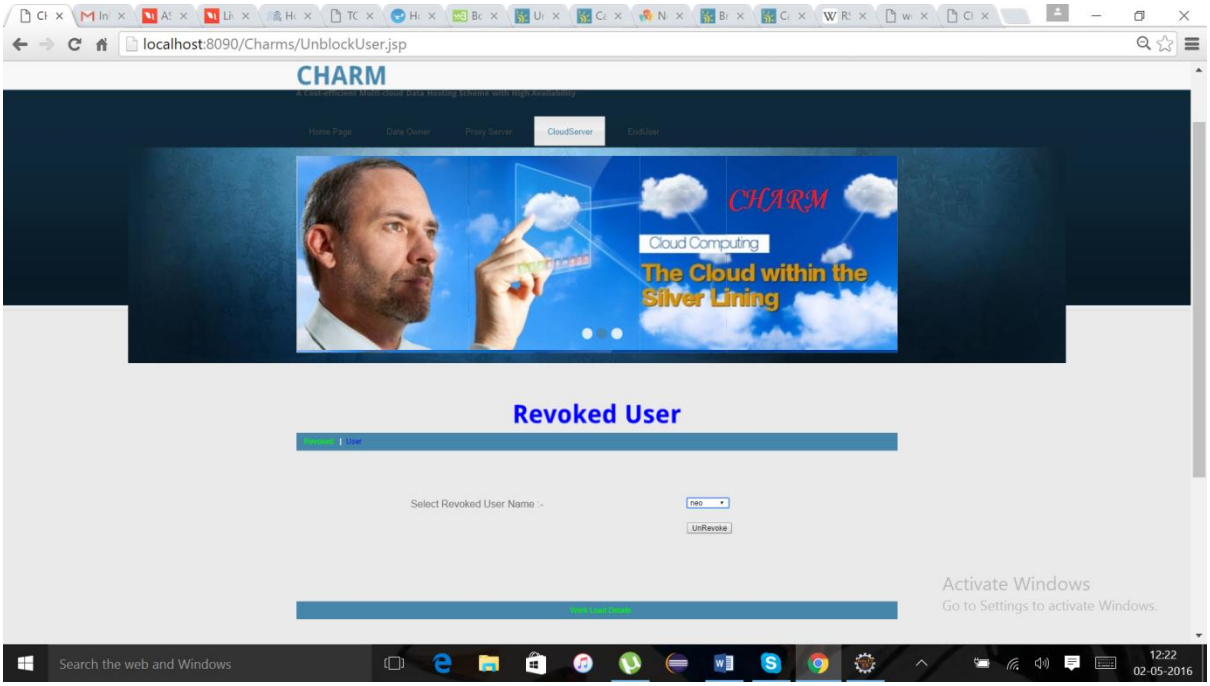

Fig 8.2.9 Unrevoked User

8.3 Proxy Server

Testing the Proxy Server Login Credentials

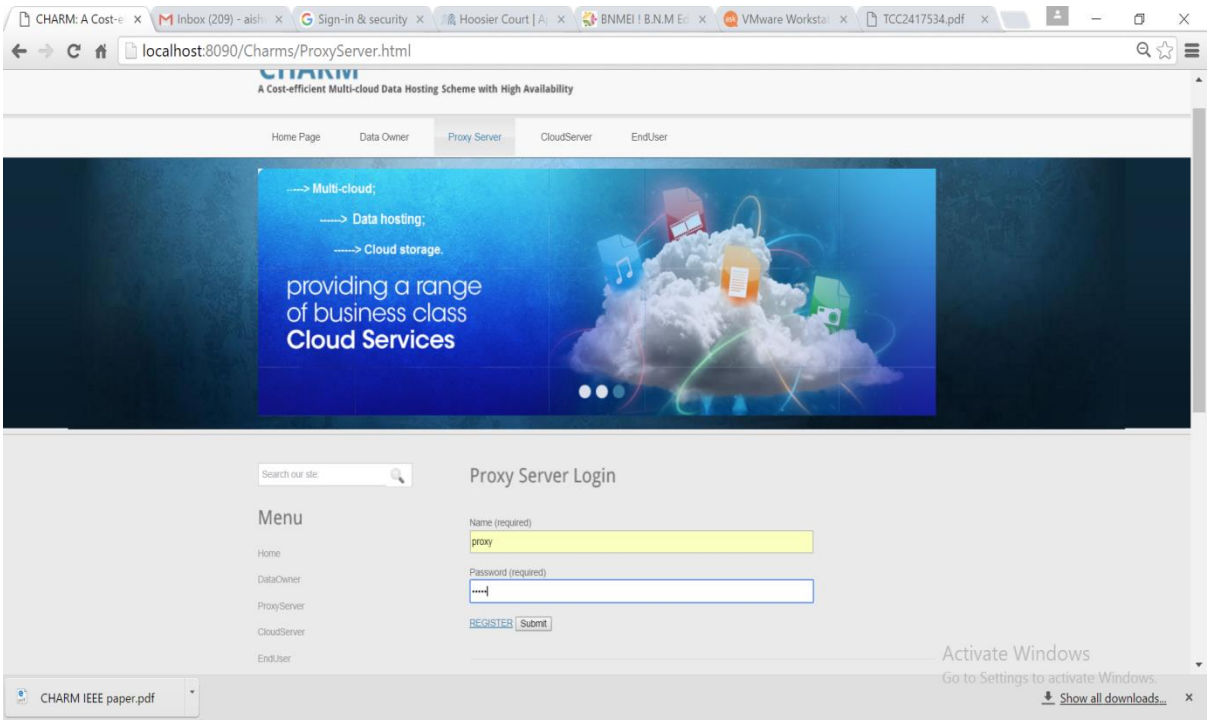

Fig 8.3.1 Proxy Server Login

Testing if the migration details are available

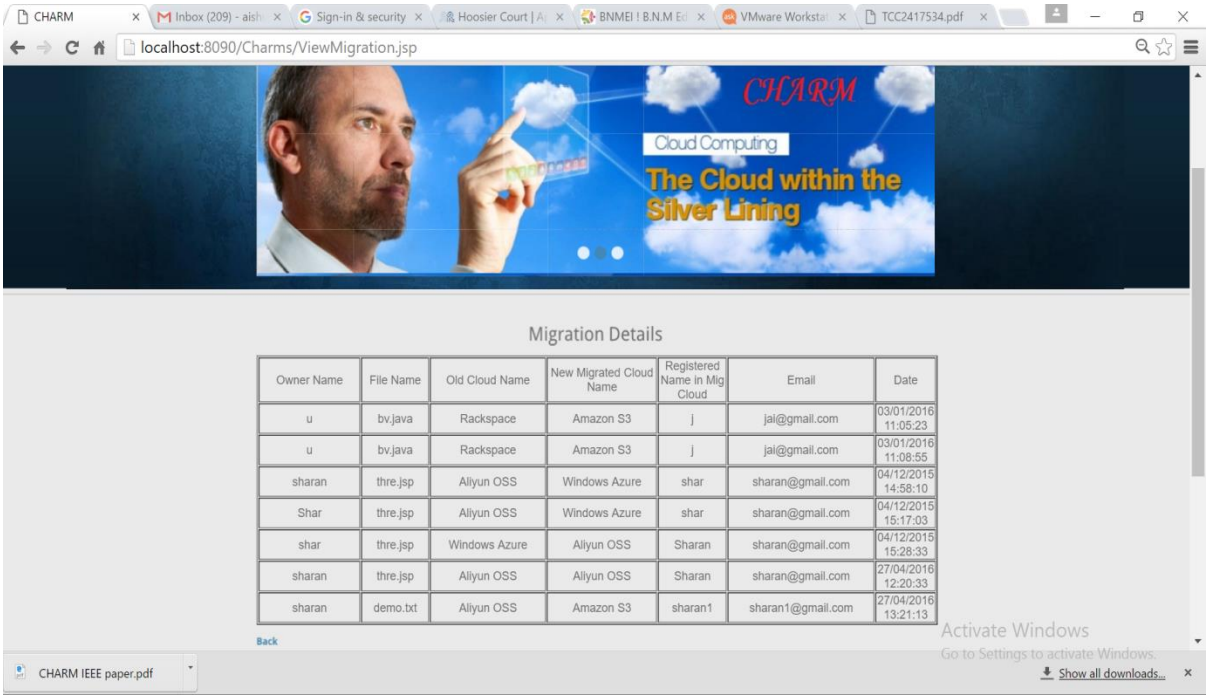

Fig 8.3.2 Migration Details

Checking the availability of VM resources and associated price details

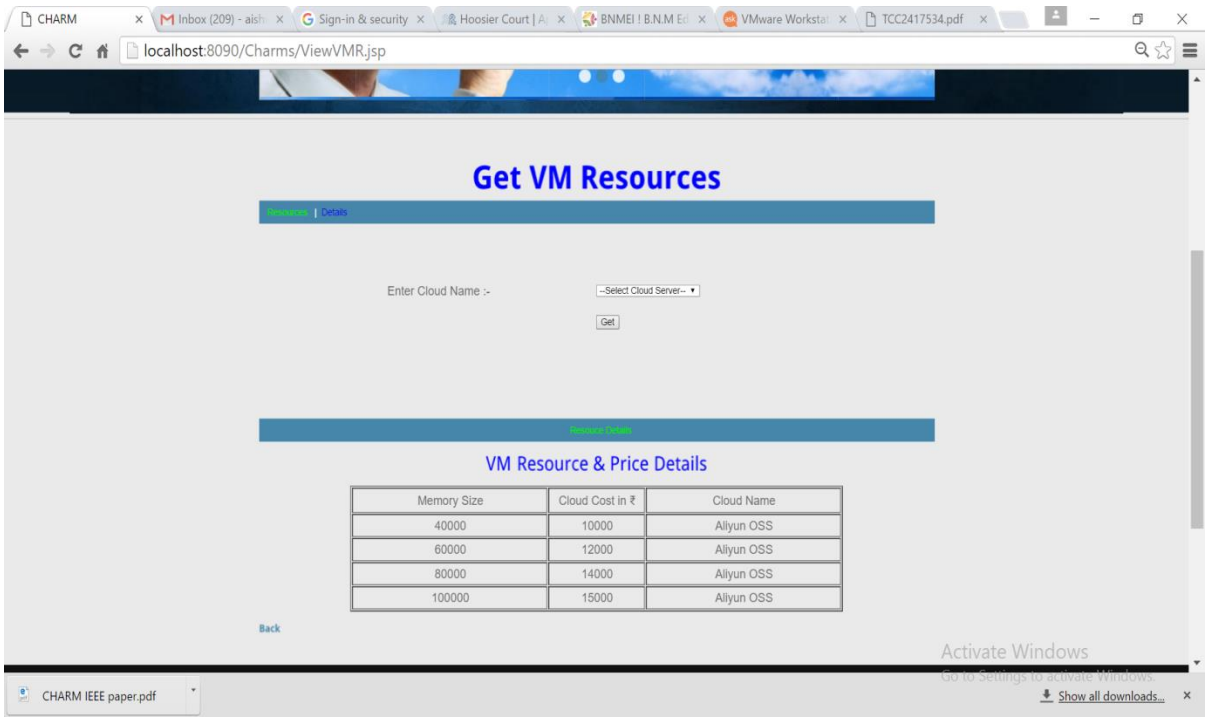

Fig 8.3.3 VM Resources

Checking whether the work load details are available

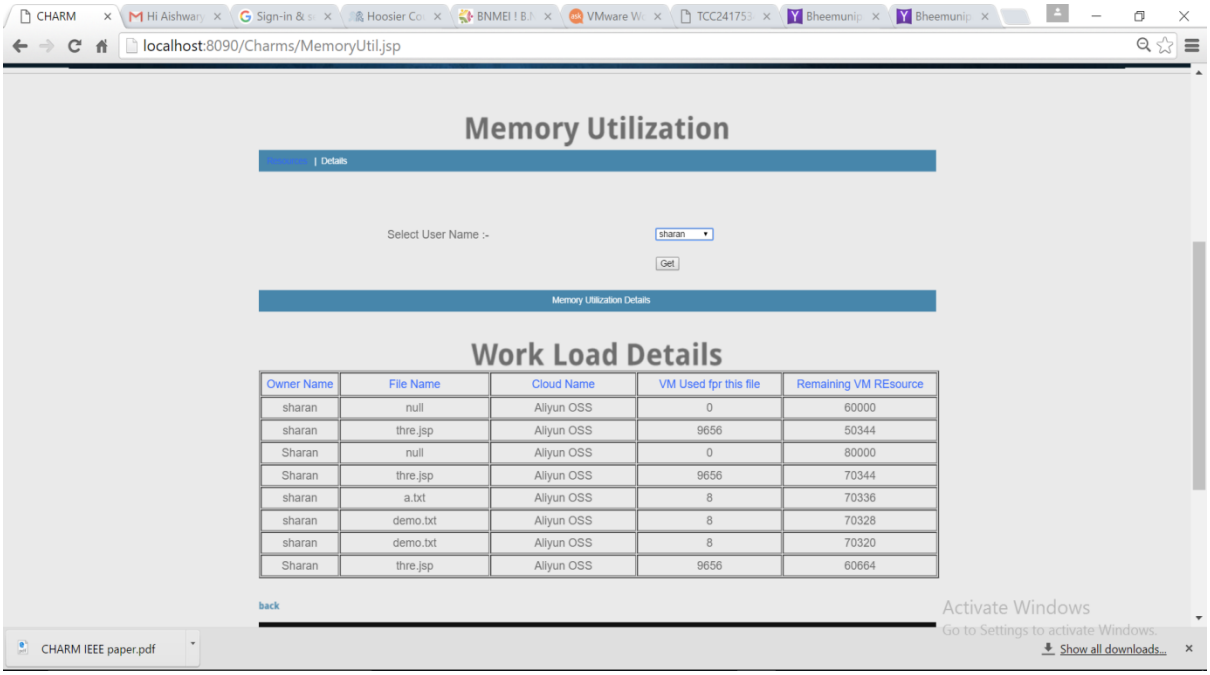

Fig 8.3.4 Workload Details

Checking whether the uploaded files with details are available

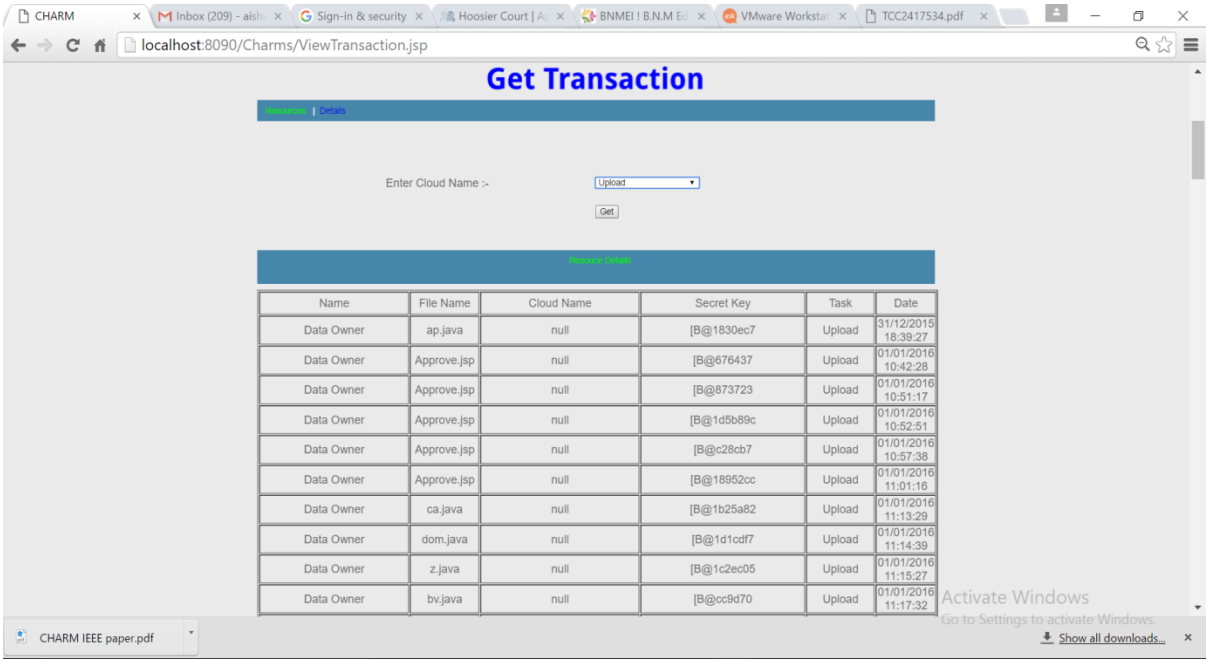

Fig 8.3.5 Upload Transaction

Checking whether the downloaded files with details are available

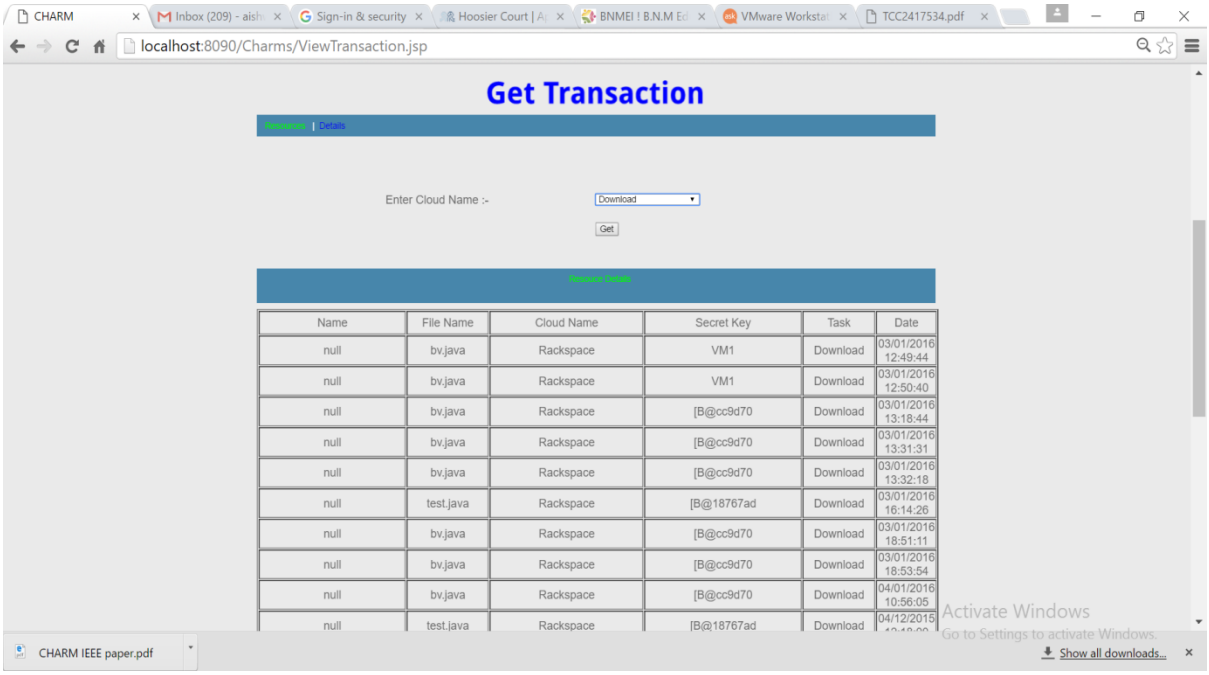

Fig 8.3.6 Download Transaction

Checking whether the migrated files with details are available

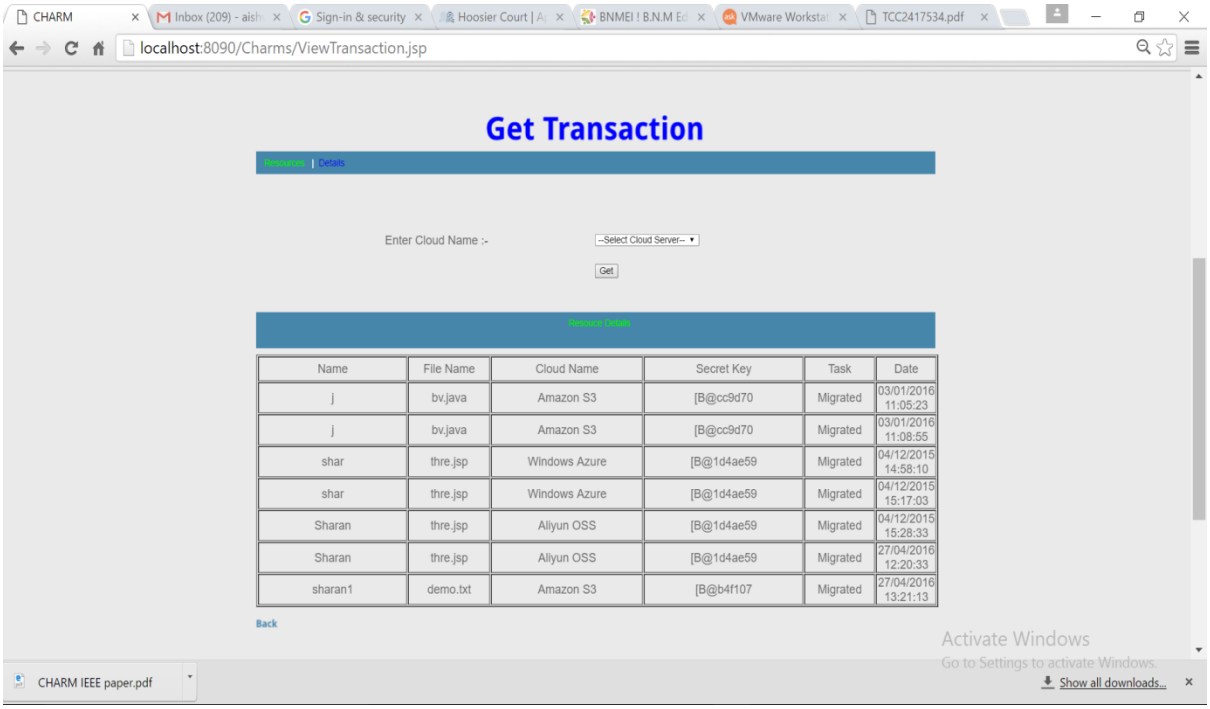

Fig 8.3.7 Migration Transaction

Checking whether the proxy files with details are available

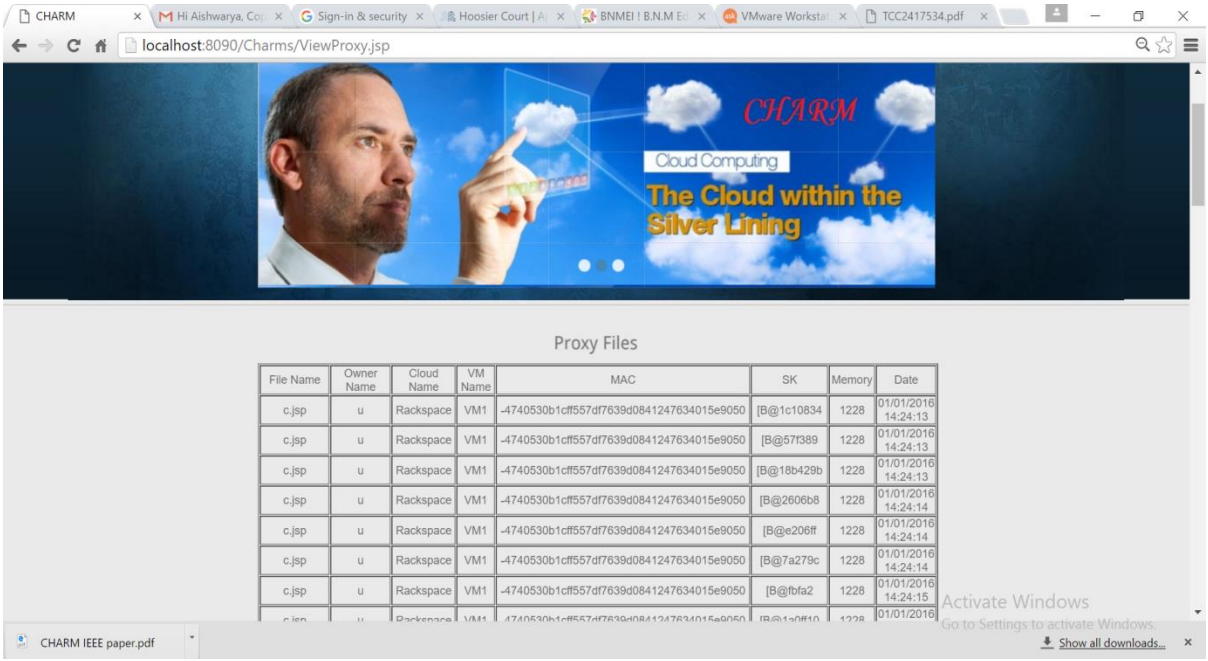

Fig 8.3.8 Proxy Files Details

8.4 End User

Testing the log in credentials to check the authenticity of end user

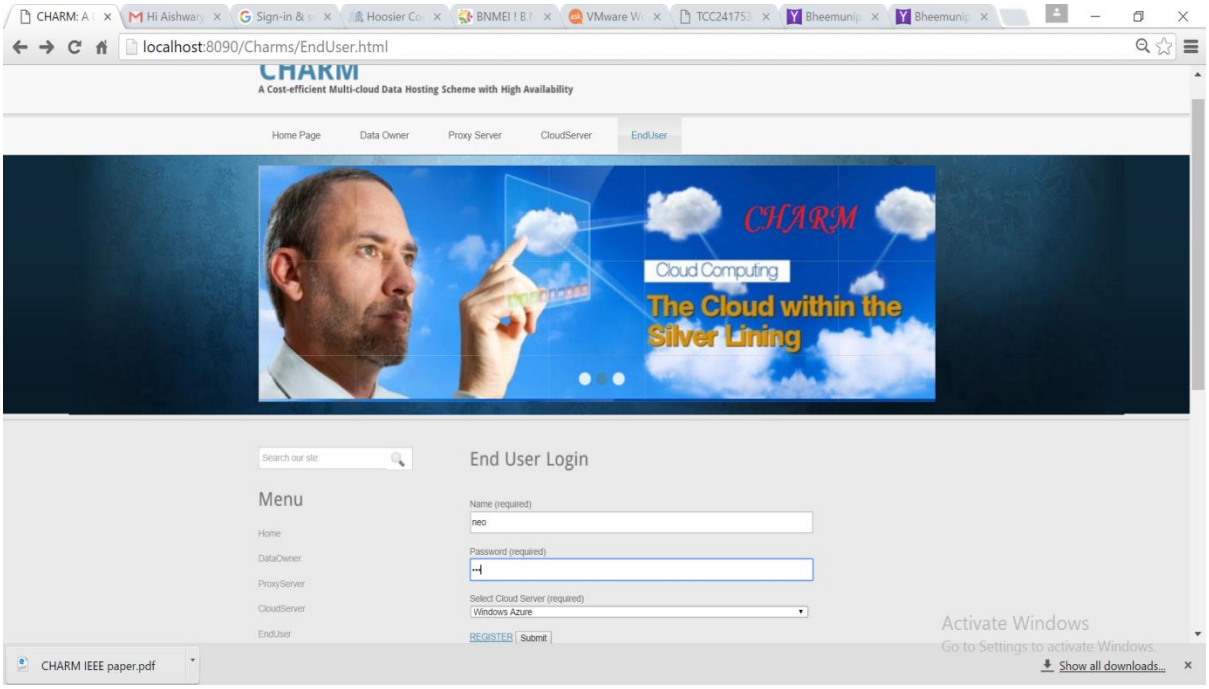

Fig 8.4.1 End User Login

Testing for availability of file

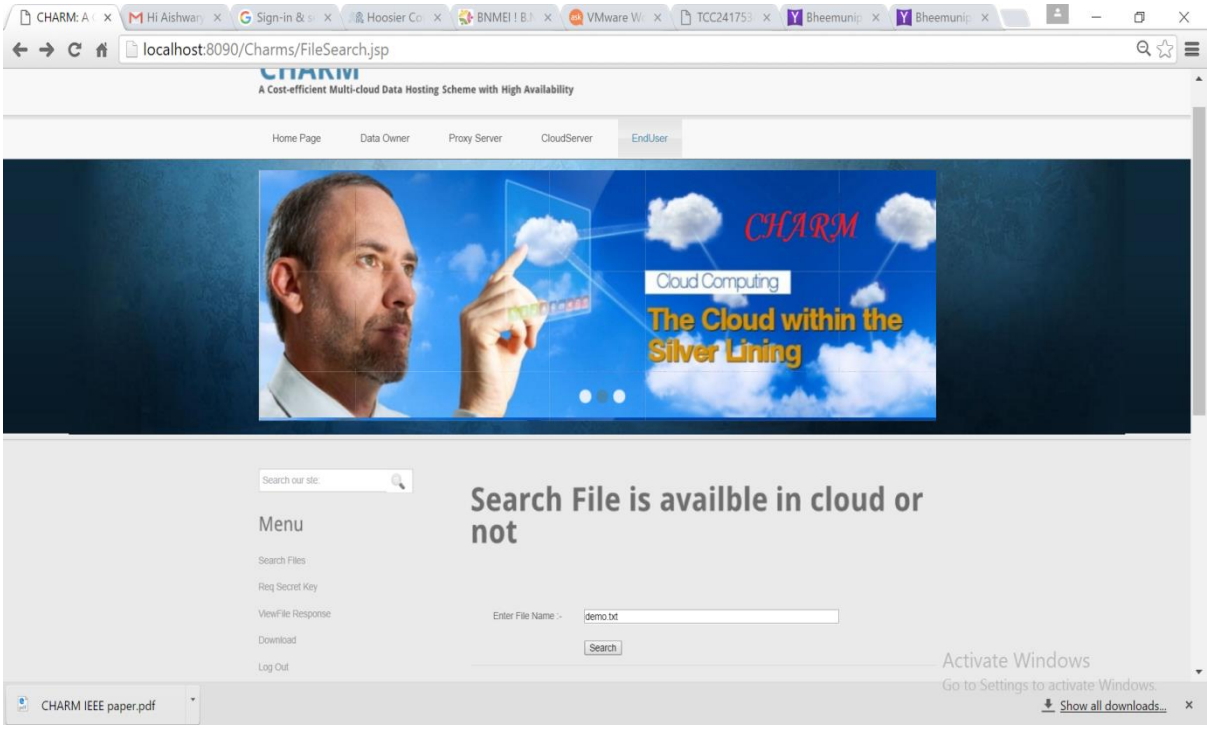

Fig 8.4.2 File availability

Checking for a particular file and send request to data owner

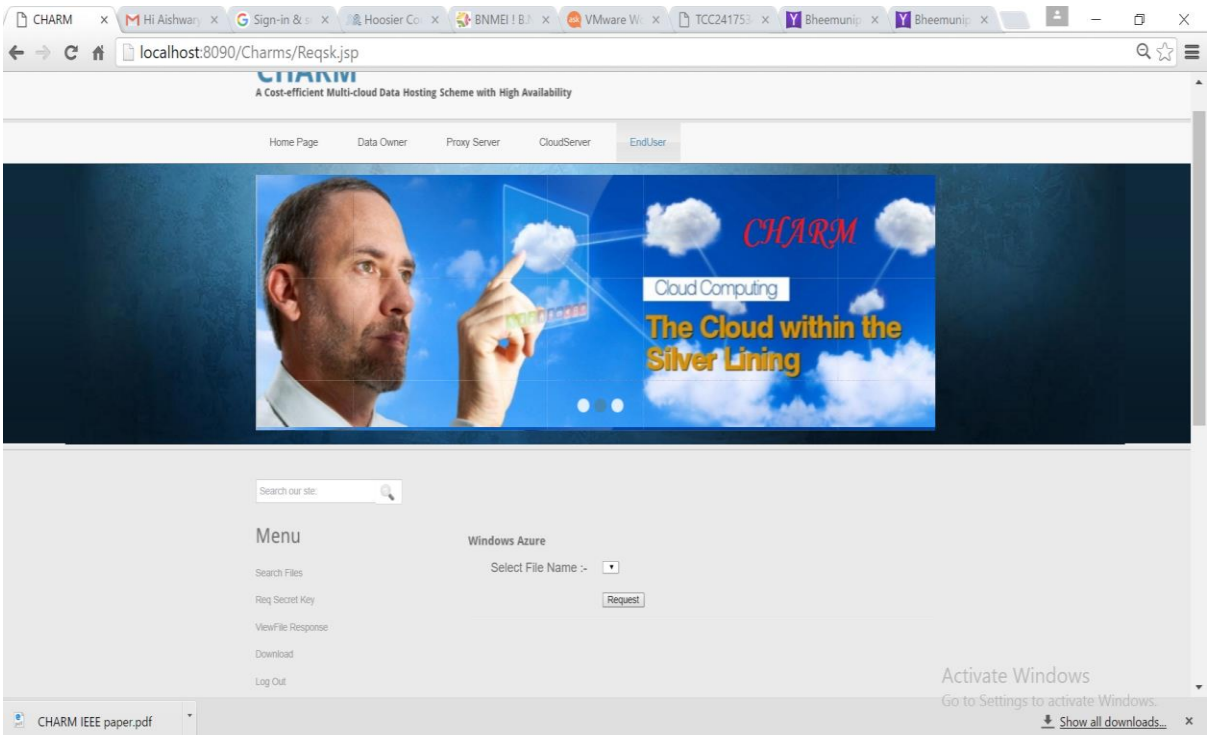

Fig 8.4.3 File Request

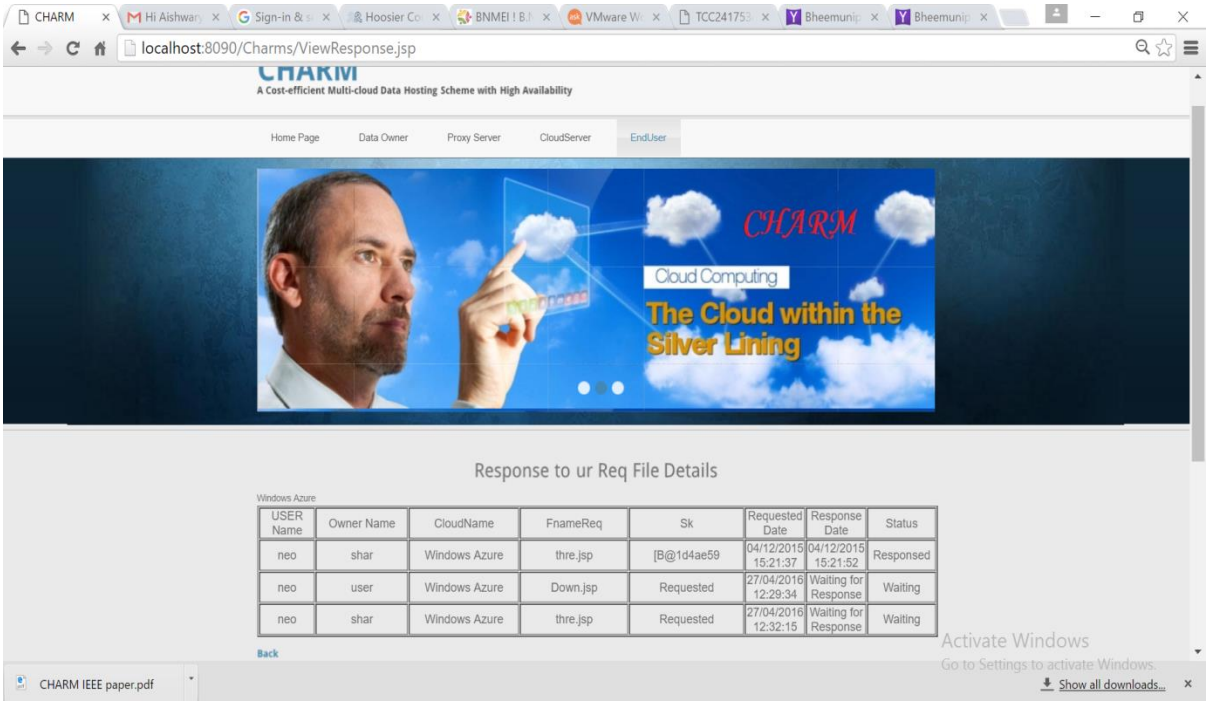

Fig 8.4.4 File Response

Downloading files from Server

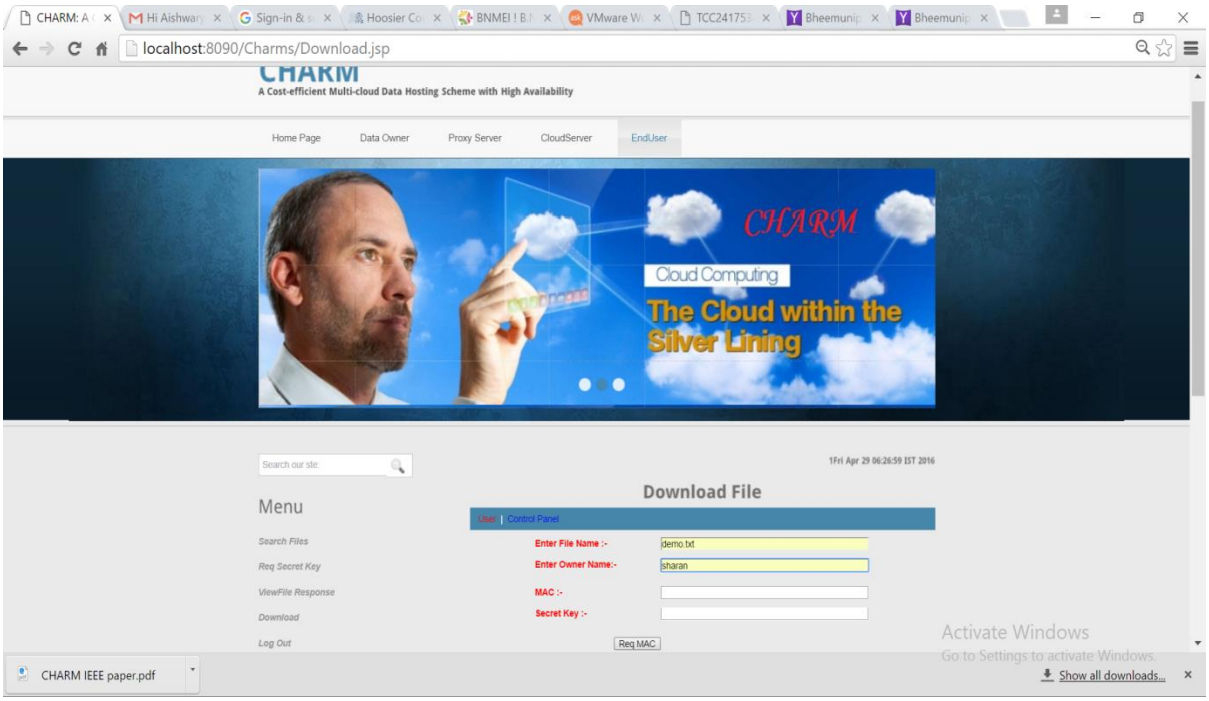

Fig 8.4.5 File Download

CONCLUSION AND FUTURE ENHANCEMENT

Cloud computing is changing the way IT departments buy IT. Businesses have a range of paths to the cloud, including infrastructure, platforms and applications that are available from cloud providers as online services. Many people may be confused by the range of offerings and the terminology used to describe them and will be unsure of the risk and benefits.

Cloud services are experiencing rapid development and the services based on multi-cloud also become prevailing. One of the most concerns, when moving services into clouds, is capital expenditure. So, in this paper, we design a novel storage scheme CHARM, which guides customers to distribute data among clouds cost-effectively. CHARM makes fine-grained decisions about which storage mode to use and which clouds to place data in. The evaluation proves the efficiency of CHARM.

Cloud fears largely stem from the perceived loss of control of sensitive data. Current control measures do not adequately address cloud computing's third-party data storage and processing needs. In approaches I present, the writers propose to extend control measures from the enterprise into the cloud through the use of Trusted Computing and applied cryptographic techniques. These measures should alleviate much of today's fear of cloud computing, and, I believe, have the potential to provide demonstrable business intelligence advantages to cloud participation.

The approaches also relates to likely problems and abuses arising from a greater reliance on cloud computing, and how to maintain security in the face of such attacks. Namely, the new threats require new constructions to maintain and improve security. Among these are tools to control and understand privacy leaks, perform authentication, and guarantee availability in the face of cloud denial-of-service attacks.

Since almost everyone in this era is acquainted with messaging apps like WhatsApp, it would Be a fantastic mode to inform customers or data owners about the details of a particular cloud Server or even the amount of storage capacity of a cloud and its cost. The information can be sent via text messages that are notified on the customer or owners smart phones. Also, various mobile applications can be built to indicate the amount of space left on a particular server, as we add more content into it; just like the battery life indicator on any of the electronic devices. The memory indicator could have 3 different colors: Green- to indicate

full memory, Yellow- to indicate around half of the available space has been used and Red to indicate that very little memory is left on each cloud server. Better encryption strategies could be developed to ensure better privacy and no intrusion from third party sources.

BIBLIOGRAPHY

[1] "Aliyun OSS (Open Storage Service)," http://www:aliyun:com/product/oss.

[2] "Gartner:op 10 cloud storage providers," http://www:networkworld:com/news/2013/010313-gartner-cloud-storage-265459:html?page=1.

[3] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai,and Z.-L. Zhang, "Towards Network-level Efficiency for Cloud StorageServices," in IMC. ACM, 2014.

[4] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Y. Zhao, C. Jin, Z.-L. Zhang,and Y. Dai, "Efficient Batched Synchronization in Dropbox-like CloudStorage Services," in Middleware. ACM/IFIP/USENIX, 2013.

[5] C. M. M. Erin Allen, "Library of Congress and DuraCloud Launch PilotProgram Using Cloud Technologies to Test Perpetual Access to DigitalContent," The Library of Congress, ews Releases, http://www:loc:gov/today/pr/2009/09-140:html.

[6] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: ComparingPublic Cloud Providers," in IMC. ACM, 2010.

[7] "Windows Azure pricing updates," http://azure:microsoft:com/en-us/updates/azure-pricing-updates/.

[8]googleCloudPlatformpricingupdates," //googlecloudplatform:blogspot:com/2014/03/google-cloud-platformlive-blending-iaas-and-aas-moores-law-for-the-cloud:html.

[9] "Its Official, The Nirvanix Cloud Storage Service Is ShuttingDown," http://techcrunch:com/2013/09/27/its-official-the-nirvanixcloud-storage-service-is-shutting-down/.

[10] "Shutting down Ubuntu One file services," http://blog:canonical:com/2014/04/02/shutting-down-ubuntu-one-file-services/

[11]"NirvanixProvidesCautionaryTaleForCloudrage,"http://www:forbes:com/sites/tomcoughlin/2013/09/30/nirvanixprovides-cautionary-tail-for-cloud-storage/.

[12]"GoogleOutagesDamageCloudCredibility," https://www:networkworld:com/news/2009/092409-google-outages-damagecloud:html.

[13] "Rackspace to issue as much as $3.5M in customer credits afteroutage," http://www:networkworld:com/news/2009/070609-rackspaceoutage:html.

[14] "Summary of the Amazon EC2 and Amazon RDS Service Disruptionin the US East Region," http://aws:amazon:com/cn/message/65648/.

[15] A. Bessani, M. Correia, B. Quaresma, F. Andr´e, and P. Sousa, "DepSky:Dependable and Secure Storage in a Cloud-of-Clouds," in EuroSys.[16] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A Casefor Cloud Storage Diversity," in SoCC. ACM, 2010.

[17] "DuraCloud," http://www:duracloud:org/.

[18] "Cloud Foundry," http://www:cloudfoundry:org/.

[19] "Apache Libcloud," http://libcloud:apache:org/.

[20] "AmazingStore," http://cn:amazingstore:org/