A

PROJECT REPORT

ON

ENSURING DISTRIBUTED ACCOUNTABILITY FOR DATA SHARING IN THE CLOUD

A project report submitted to The Jawaharlal Nehru Technological University
in partial fulfillment for the award of the

Bachelor of Technology
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by
B. RAM PRASAD (11RJ1A0514)
D. SAINATH (11RJ1A0527)
A. ABHINEETH REDDY (11RJ1A0502)

UNDER THE GUIDANCE OF

Mr. N. SATEESH

(ASSISTANT PROFESSOR)

(CSE DEPARTMENT)

MALLA REDDY INSTITUTE OF TECHNOLOGY
(Affiliated to JNTU, Hyderabad)
Maisammaguda, Secunderabad- 500 100

MALLA REDDY INSTITUTE OF TECHNOLOGY

Maisammaguda, Dhulapally Post (Via Hakimpet), Secunderabad – 500014

(Approved by AICTE, Affiliated to JNTU, Hyderabad)

CERTIFICATE

This is to certify that this project work entitled "**Ensuring distributed accountability for data sharing in the cloud**" is a bonafide work carried out by **B. Ram Prasad** bearing Hall Ticket Number: **11RJ1A0514**, **D. Sainath** bearing Hall Ticket Number: **11RJ1A0527, A. Abhineeth Reddy** bearing Hall Ticket Number: **11RJ1A0502** of **COMPUTER SCIENCE AND ENGINEERING DEPARTMENT** at **MALLA REDDY INSTITUTE OF TECHNOLOGY** and submitted to **JNT UNIVERSITY, HYDERABAD** in the partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY**.

Guide	Head of the Department
Mr. N. Sateesh M.Tech	Mr. N. Sateesh M.Tech
Assistant Professor,	Assistant Professor,
CSE Department	CSE Department

External Examiner

ACKNOWLEDGEMENT

We hereby take the opportunity to thank our beloved **Principal Dr. M. Murali Krishna** for his gratitude and kindness by giving us all the facilities required for the completion of the project.

We would like to extend our sincere thanks to **Mr. N. Sateesh, Head of the Department, Computer Science and Engineering**, for his encouragement and support at all the stages of the project.

We would like to take this opportunity to express our heartfelt gratitude and our sincere thanks to **Mr. T. Srikanth, Asst. Professor, CSE Dept**., our project coordinator at **Malla Reddy Institute Of Technology** who helped us a lot for the successful completion of our project "**Ensuring distributed accountability for data sharing in the cloud**".

We would also like to extend our enormous gratitude and sincere thanks to our project internal guide **Mr. N. Sateesh, Asst. Professor, CSE Dept**., for his efficient, able advice and helping hand for developing the project.

We also thank the entire teaching faculty who were instrumental in making this project a successful one.

BY
B. Ram Prasad (11RJ1A0514)
D. Sainath (11RJ1A0527)
A. Abhineeth Reddy (11RJ1A0502)

DECLARATION

We hereby declare that the project entitled "**Ensuring distributed accountability for data sharing in the cloud**" submitted to **Malla Reddy Institute Of Technology, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH)** for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** is a result of work done by us. It is further declared that the project report or any part of it has not been previously submitted to any university or institute for the award of degree or diploma.

B.Ram Prasad (11RJ1A0514)
D.Sai Nath (11RJ1A0527)
A. Abhineeth Reddy (11RJ1A0502)

ABSTRACT

Description:

There is a huge problem where all users' fears of losing control of their own data (particularly, financial and health data) can become a significant barrier to the wide adoption of cloud services. To address this problem, this app provides a Mobile services interface that can be used to store and retrieve any type of files, at any time, from anywhere. Here user's files will be stored in a memory location called Bucket. From this bucket user can upload as single or multiple files and user can download as single and multiple files from the bucket. A fixed amount space will be given to all users. Once the space is completely used by the user, he/she can't use this free app anymore. This cloud folder will make sure that all your data which is uploaded is safe in the cloud and accessible from anywhere.

The main theme of our application is store the data in the cloud folder. This application can be used for all the android mobile users who may want to store image files, audio/video files, text files in the cloud and also provides to download the data (images files, audio/video files, text files) in the SDCard.

INDEX

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Cloud computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. This technology allows for much more efficient computing by centralizing storage, memory, processing and bandwidth. Cloud computing is a comprehensive solution that delivers IT as a service.

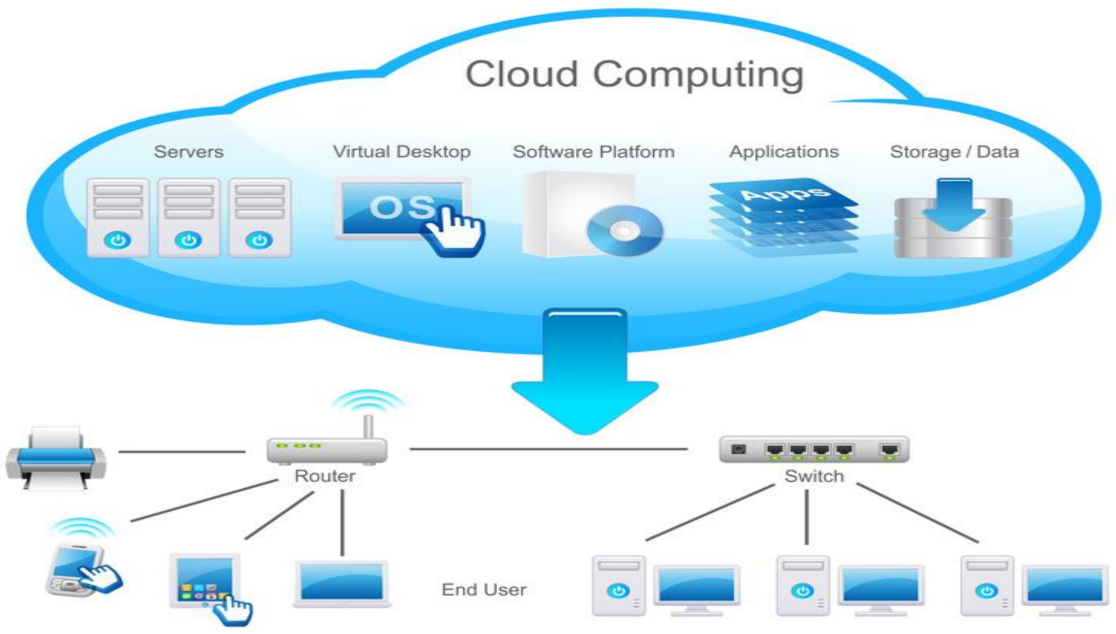


Fig 1.1 Cloud Computing

The flexibility of cloud computing is a function of the allocation of resources on demand. Before cloud computing, websites and server-based applications were executed on a specific system. Cloud computing is broken down into three segments application, storage and connectivity.

CLOUD COMPUTING MODELS

Service model

There are three fundamental Service models in Cloud computing. Three service models are explained below.

1.Software as a Service

Cloud Applications or Software as a Service (SaaS) refers to software delivered over a browser. SaaS eliminates the need to install and run applications on the customer's own computers/servers and simplifies maintenance, upgrades and support. Examples of SaaS are Facebook, SalesForce, etc. The Cloud Computing service model is given in figure 1.2.

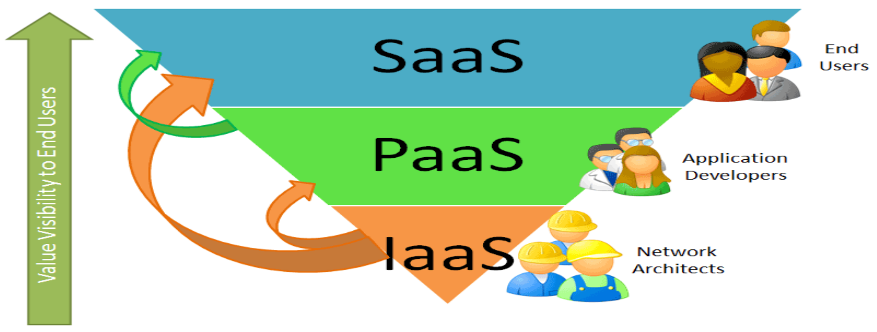


Figure 1.2 Cloud Computing Service model

2. Platform as a Service

Cloud platform services or Platform as a Service (PaaS) refers to an environment for software development, storage and hosting delivered as-a-service over the Internet. Examples of PaaS are Google App Engine, Force.com, Microsoft Azure, WOLF, etc.

3. Infrastructure as a Service

Cloud infrastructure services or Infrastructure as a Service (IaaS) delivers a computing infrastructure, typically a virtualization environment, as-a-service. Examples of IaaS are virtual servers leased by Amazon, Rackspace, etc.

Deployment model

Cloud Computing Deployment model is given in figure 1.3.

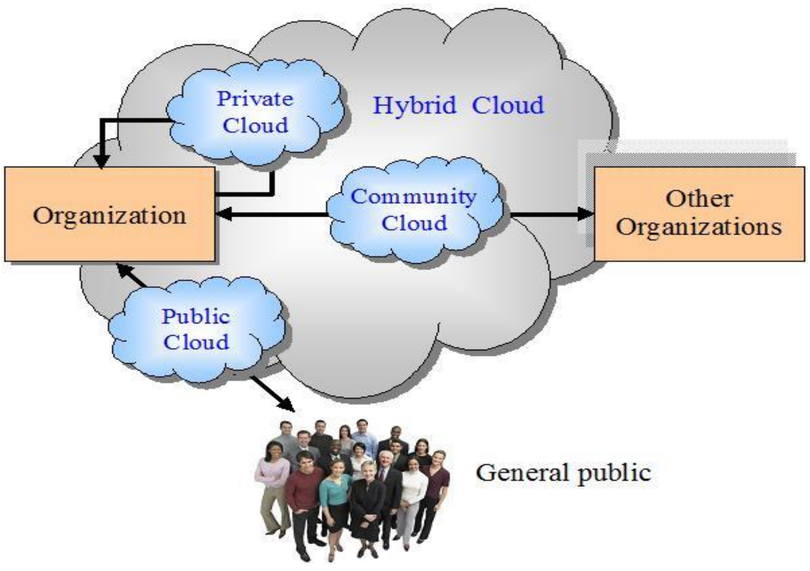


Figure 1.3 Cloud Computing Deployment model

Each company chooses a deployment model for a cloud computing solution based on their specific business, operational, and technical requirements. Four primary cloud deployment models are private cloud, community cloud, public cloud, and hybrid cloud.

- **Public Cloud**

Public cloud refers to Cloud Computing in the traditional mainstream sense, whereby resources are dynamically provisioned on a fine-grained, self-service basis over the Internet. These resources are provisioned via web applications/web services, from an off-site third-party provider who shares resources and bills the customer on a fine-grained utility computing basis.

- **Community Cloud**

A community cloud is established among several organizations that have similar requirements and seek to share their computing infrastructure in order to realize some of the benefits of the Public Cloud. With the costs spread over fewer users than a Public Cloud (but more than a single tenant) this option is more expensive but may offer a higher level of privacy, security and/or policy compliance.

- **Private Cloud**

A term that is similar to, and derived from, the concept of Virtual Private Network (VPN), is applied to Cloud Computing. The Private Cloud delivers the benefits of Cloud Computing with the option to optimize on data security, corporate governance and reliability.

- **Hybrid cloud**

The cloud infrastructure is shared by several organizations with common concerns (eg, mission, security requirements, policy, and compliance considerations). For example, the Google GovCloud provides the Los Angeles City Council with a segregated data environment to store its applications and data that are accessible only to the city's agencies.

1.1 Project Description

In the present generation mobile is a common and sophisticated use in the human daily life. It has been merged into the life's and made a space for its time, so everything has brought into the portable life style, whatever we need where ever you need, if it is a file, image, document or a video or any ticket booking at any place around you, everything happen around our figure tips only because of this great invention within a small size device, its nothing but a mobile device. We can send any kind of a file, any kind of a text to anywhere within a fraction of seconds only by moving your fingers. This type of smart phones made our life's easier than before. For example if we need to send a mail, we reach to the desktop and get the website and login into our account and send it to him, but now a days we had a App to create a mail and a network to send.

As the mobile users are more the mobile devices also great in brands, there are different kinds of brands with different kinds of Operating Systems but among those only two OS are in top they are

1. Apple iOS
2. Android

Apple iOS:

iOS (previously **iPhone OS**) is a mobile operating system developed by Apple Inc. and distributed exclusively for Apple hardware. It is the operating system that powers many of the company's iDevices. It’s been one of the most wanted and the best operating system, which has the flexibility in operating and the fluid Graphical User Interface which creates good interaction between system and the user. In security IOS has its own symbol with a powerful encryption algorithms and the newly developed security figure scanner which will pass all the secure websites like online transaction and online email accounts.

In this IOS system has the best and powerful applications like free pass which will operate like ATM card presently this was been implicated at other countries like USA. In IOS System there is an app which is ‘App Store’ it only has the standard and the most useful apps it won’t allow for the useless apps like other Operating systems.

Android OS:

Android is a mobile Operating System which was developed by Google. And distributed for different mobile phone, which it doesn’t stick to a single hardware. This is an Open Source System which is developed in Linux system which is flexibility to use on any sort of hardware. In security it uses the Google encryptions and this was very much secure and there is an option of encrypting the sd to secure all of the users data. The Cloud storage which it is the main assert to this operating system which backups all the data into cloud like contacts, photos and videos it makes lot more easy to the user to grasp any data whenever he wants, and he won’t lose any information from his mobile even he format his mobile for some reason.

The Google mobiles are very much popular for its special and great quantity of applications, in this user gets thousands and lakhs of applications to use on his system. To acquire those applications Google has the application which is ‘Google Play’ which will show applications and install if you need into your mobile.

Advantages of Android on IOS:

Android has more advantages comparing it with IOS as android is an Open Source system it is free to use on any hardware system as the IOS system has no advantage towards it because it’s only stick to Apple hardware systems, As the powerful operating system need a powerful hardware Apple build their own systems.

In Developing side of view Android is the best operating system which gives the Administrative rights to develop the application and run on that mobile as user wants but in IOS the Administrative rights won’t be given to the user as in the developing flip also it won’t be allowed to grasp all its administrative rights to the user.

Cloud Sharing:

Mostly all the organizations are connected with Desktops or Laptops but it can’t be carry all the way around the places but the Mobile phones are more used by everyone and it’s easy to access so it’s an innovative thought of collaborating both the Mobile and the organization by creating a social app and sharing the information to one another so that the information transfer very easily and securely by giving some access restrictions like public and private.

This type of application gives more comfort and more useful to the organizations to make contact to all and transferring the information from one another so that it saves most of the time and this application has the capability of sharing the photos, videos and other type of data which are like text, program files, music files or any type of an extension files depending upon the organization which they are working on and wanted to send.

Security:

In any organization security is the main aspect as there should be no leakage outside of the organization, this application has a higher security configuration. For the organization there will be registration mobile number and the password which given by the organization.

As it deals with the confidential data it separates the data depending upon how the user wants to share to others. Data are differentiating between two types which are:

Private Data:

In this the data are been in cloud but the data not been shown to anyone but the only person who has uploaded and this type of data cannot be access even in the read option also so that data can be secure and confidential data.

Public Data:

In this the data which is uploaded into the cloud are been shown to everyone and they can be access by anyone but only if he is in that organization and they are can be downloaded or deleted by anyone in that organization.

In future these types of applications are more useful and greatly working for the organization to be in connective and more communicative to others and time saving process.

CHAPTER 2

LITERATURE SURVEY

Access control policies:

There are broadly three types of access control: user-based access control (UBAC), role-based access control (RBAC), and attribute-based access control (ABAC).

1. User-based access control :

There are a number of existing systems based on the concept of roles and permissions in the market and we will take a closer look at those systems.

A. DAC

DAC is a kind of access control model which can allow subjects to grants certain restriction to access. It’s based on access matrix model. DAC allows subjects grant or revoke access privilege to the objects which belong to them. This makes access control discretionary, which is the main disadvantage of DAC. Management of access control is very discrete. Relationship among clients in system cannot be displayed clearly, which also makes management very difficult. Discretionary Access Control (DAC) allows authorized users to change the access control attributes of objects, thereby specifying whether other users have access to the object. A simple form of Discretionary Access Control (DAC) might be file passwords, where access to a file requires the knowledge of a password created by the file owner. In Linux, the file permission is the general form of Discretionary Access Control (DAC). Discretionary Access Control (DAC) is the setting of permissions on files, folders, and shared resources. The owner of the object (normally the user who created the object) in most operating system environments applies discretionary access controls. This ownership may be transferred or controlled by root/administrator accounts. Discretionary Access Control (DAC) is controlled by the owner or root/administrator of the Operating System, rather than being hard coded into the system. The Discretionary Access Control (DAC) mechanisms have a basic weakness, and that is they fail to recognize a fundamental difference between human users and computer programs.

B. MAC

MAC is also called Latices-based Access Control, which is designed for stricter and more secure access control model than DAC. In MAC model, system assigns a special security attribute to subject and object. Generally, a subject can't change the security attributes of another subject. It is the system that decides whether the subject has right to access object by comparing the security attributes of subject and object. The traditional discretionary and mandatory access controls (DAC and MAC**,** respectively) are inappropriate for the information security needs of many organizations. RBAC has been proposed **as** an alternative, and supplement, to traditional DAC and MAC. MAC, however, is not without serious limitations. The assignment and enforcement of security levels by the system under the MAC model places restrictions on user actions that, while adhering to security policies, prevents dynamic alteration of the underlying policies, and requires large parts of the operating system and associated utilities to be "trusted" and placed outside of the access control framework.

2. Role-based access control :

Today cloud services have been mostly used by different industries to improve the productivity and automate their customer support. So the RBAC is used to provide flexible security and access control to organizational information and resources. This can be implemented by providing authentication, authority and audit control. The basic idea of RBAC is grants access privilege to a certain role. User will play some roles in a system in order to get privileges. RBAC divides roles by its responsibility and duty which are relatively stable in a system. The systems only grant roles to users. The roles become bridge between subjects and objects in access control mechanism. The major concepts in this system are the use of a hierarchy used for the assignment of roles and permissions along with making the framework independent of any specific database so that it can be used on any of the databases. The additional support to active and non-active users is also proposed in the system and a mechanism to assigning permissions to a group of users depending on their roles.

A. Authentication

This confirms the user's identity. It checks the user identity by the application. This is a two-step method in this first who are you? Is checked by using the related information of the user and then the authentication will be provided i.e. who are you; this can be implemented by providing the username and password. It can be treated as a low level security.

B. Authorization:-

Authorization means what a user can do. In this step the actual security to the information and resources of the organization is provided by assigning the roles to the user but sometimes it is very good that providing an access to a user dynamically that means at the time of authentication the roles are not assigned to the user. The roles are provided when the request is send by the user to the admin. Then the admin checks all the permission and policies to the requested method if the user has access on to that particular requested method then and then only the request is granted else the message will be shown i.e. the access is limited. This step is heaviest than the authentication because all the access control and permissions are implemented by coding.

C. Audit Control:-

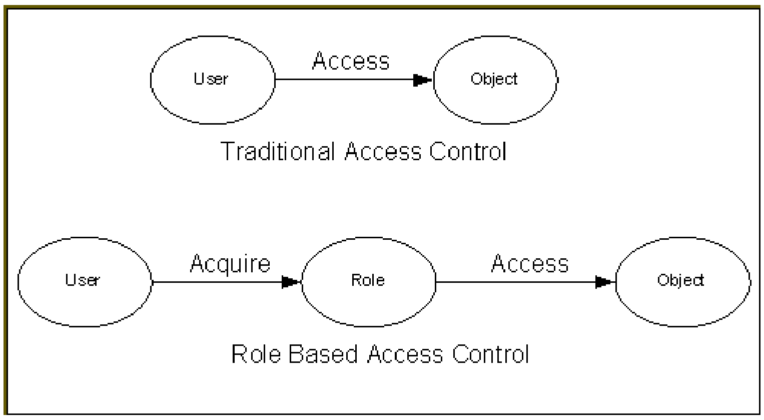
Keep tracking of the sensitive transaction is known as an audit control. Audit should enable you to review who did what in your application, when and who granted which permissions to which user.

D. Persistence Schema:

The schema should have to be persisted after each transaction and it should not have to be change in between transactions.

E AD/Non-AD Users:

AD user means active directory user. Those users having or working in the same organization. These users are having a access to the internal information of the organization as per their role assigned in an organization. But also here in RBAC we are going to provide the access to the non-AD user on the information of the organization as per their requirement. Again before providing the access to the non-AD user all the permissions and condition are going to be checked. Non-AD user all the permissions and condition are going to be checked.



Formal Description of RBAC

To clarify the notions presented in the previous section, we give a simple formal description, in terms of sets and relations, of role based access control. No particular implementation mechanism is implied.

For each subject, the active role is the one that the subject is currently using:

AR(s: subject) = {the active role for subject s}.

Each subject may be authorized to perform one or more roles:

RA(s: subject) = {authorized roles for subject s}.

Each role may be authorized to perform one or more transactions:

TA(r: role) = {transactions authorized for role r}.

Subjects may execute transactions. The predicate exec(s,t) is true if subject s can execute transaction t at the current time, otherwise it is false:

exec(s: subject, t: tran) = true iff subject s can execute transaction t.

Three basic rules are required:

1. Role assignment: A subject can execute a transaction only if the subject has selected or been assigned a role:

s : subject, t : tran,(exec(s, t) AR(s)).

The identification and authentication process (e.g. login) is not considered a transaction. All other user activities on the system are conducted through transactions. Thus all active users are required to have some active role.

3. Role authorization:

A subject's active role must be authorized for the subject:

s : subject,(AR(s) RA(s)).

With (1) above, this rule ensures that users can take on only roles for which they are authorized.

3. Transaction authorization: A subject can execute a transaction only if the transaction is authorized for the subject's active role:

s : subject, t : tran, (exec(s, t) t TA(AR(s))).

With (1) and (2), this rule ensures that users can execute only transactions for which they are authorized. Note that, because the conditional is ``only if", this rule allows the possibility that additional restrictions may be placed on transaction execution. That is, the rule does not guarantee a transaction to be executable just because it is in TA(AR(s)), the set of transactions potentially executable by the subject's active role.

In the preceding discussion, a transaction has been defined as a transformation procedure, plus a set of data items accessed by the transformation procedure. Access control in the rules above does not require any checks on the user's right to access a data object, or on the transformation procedure's right to access a data item, since the data accesses are built into the transaction. Security issues are addressed by binding operations and data into a transaction at design time, such as when privacy issues are addressed in an insurance query transaction.

It is also possible to redefine the meaning of ``transaction" in the above rules to refer only to the transformation procedure, without including a binding to objects. This would require a fourth rule to enforce control over the modes in which users can access objects through transaction programs. For example, a fourth rule such as

4. s : subject, t : tran, o : object, (exec(s, t) access(AR(s), t, o, x)).

could be defined using a transaction (redefined to transformation procedure) to object access function access(r, i, o, x) which indicates if it is permissible for a subject in role r to access object o in mode x using transaction t, where x is taken from some set of modes such as read, write, append. Note that the Clark-Wilson access control triple could be implemented by letting the modes x be the access modes required by transaction t, and having a one-to-one relationship between subjects and roles. RBAC, as presented in this paper, thus includes Clark and Wilson access control as a special case.

Access control decisions are often based on the roles individual users take on as part of an organization. A role specifies a set of transactions that a user or set of users can perform within the context of an organization. RBAC provide a means of naming and describing relationships between individuals and rights, providing a method of meeting the secure processing needs of many commercial and civilian government organizations.

3. Attribute based access control :

In RBAC, users are classified based on their individual roles. Data can be accessed by users who have matching roles. The roles are defined by the system. For example, only faculty members and senior secretaries might have access to data but not the junior secretaries. ABAC is more extended in scope, in which users are given attributes, and the data has attached access policy. Only users with valid set of attributes, satisfying the access policy, an access the data. For instance, in the above example certain records might be accessible by faculty members with more than 10 years of research experience or by senior secretaries with more than 8 years' experience.

Access policies can be in any of the following formats:

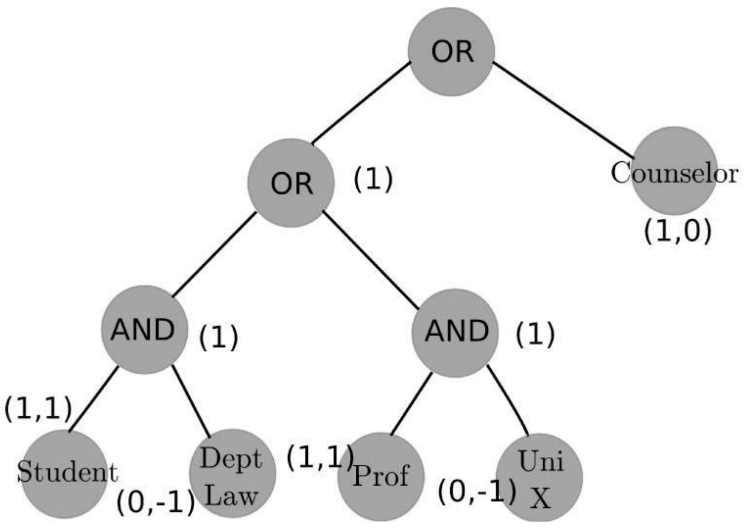
1. Boolean functions of attributes
2. Linear secret sharing scheme (LSSS) matrix (or)
3. Monotone span programs.

Any access structure can be converted into a Boolean function. An example of a Boolean function is $((b1 \wedge b2 \wedge b3) \vee (b4 \wedge b5)) \vee (b6 \vee b7))$

Where b1; b2; . . . ; b7 are attributes.

We now revisit the problem we stated in the introduction. We will use a relaxed setting. Suppose Alice is a law student and wants to send a series of reports about malpractices by authorities of University X to all the professors of University X, Research chairs of universities X; Y; Z and students belonging to Law department in university X. She wants to remain anonymous, while publishing all evidence. All information is stored in the cloud. It is important that users should not be able to know her identity, but must trust that the information is from a valid source. For this reason she also sends a claim message which states that she "Is a law student" or "Is a student counselor" or "Professor at university X." The tree corresponding to the claim policy is shown in Fig. The leaves of the tree consist of attributes and the intermediary nodes consist of Boolean operators. In this

example the attributes are “Student,” “Prof,” “Dept Law,” “Uni X,” “Counselor.” The above claim policy can be written as a Boolean function of attributes as
((Student AND Dept Law) OR (Prof AND Uni X)) OR (Counselor).



Boolean functions can also be represented by access tree, with attributes at the leaves and AND(∧) and OR(∨) as the intermediate nodes and root. Boolean functions can be converted to LSSS matrix as below: Let v[x] is parent vector. If node x =AND, then the left child is (v[x]|1), and the right child is (0, . . . , 1). If x = OR, then both children also have unchanged vector v[x]. Finally, pad with 0s in front, such that all vectors are of equal length. We do not present it here due to lack of space.

Using this algorithm, the span program for this policy is

$$M = \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 1 & 1 \\ 0 & -1 \\ 1 & 0 \end{pmatrix}$$

An assignment v = (v1, v2, v3, v4, v5) satisfies this span program if vM = (1, 0). The cloud should verify that Alice indeed satisfies this claim. Since she is a law student, v = (1, 1, 0, 0, 0) and is a valid assignment. As a valid user she can then store all the encrypted records under the set of access policy that she has decided. The access policy in case of Alice is
((Prof AND Uni. X) OR (Research Chair AND ((Uni X OR Uni Y) OR Uni Z)) OR ((Student AND Dept Law) AND Uni X).

Later when a valid user, say Bob wants to modify any of these reports he also attaches a set of claims which the cloud verifies. For example, Bob is a research chair and might send a claim “Research chair” or “Department head” which is then verified by the cloud. It then sends the encrypted data to the Bob. Since Bob is a valid user and has matching attributes, he can decrypt and get back the information. If Bob wants to read the contents without modifying them, then there is no need to attach a claim. He will be able to decrypt only if he is a Professor in University X or a Research chair in one of the universities X; Y ;Z or a student belonging to Department of Law in university X. Here it is to be noted that the attributes can belong to several KDCs. For example, the Professors belonging to university X have credentials given by the university X, and the Ph.D. degree from a University P, the student counselor might be a psychologist authorized by the

Canadian Psychological Association and assigned an employee number by a university, the research chairs can be jointly appointed by the universities X, Y , Z and the government. The students can have credentials from the university and also a department. Initially, Alice goes to a trustee, for example, the Canadian health service and presents her a health insurance number or federal agency presents her a social insurance number. Either or both of these trustees can give her token(s) Υ=(u,K_{base}, K_0, ρ). With the token she approaches the KDCs in the university X and department D and obtains the secret keys for decryption and for keys K_x and K_y for signing the assess policy. She can also access the public keys APK[i] of other KDCs.

CHAPTER 3

SYSTEM STUDY

3.1 Feasibility Study

Feasibility study refers to the overall idea of the package which we are designing. Software feasibility has four solid dimensions. Technical- is a project technically feasible? Is it within the state of the art? Can defects be reduced to a level matching the application's needs? Economical- is it financially feasible? Can development be completed at a cost the software organization, its client, or the market can afford? Operational- will the project's time-to-market beat the competition?

This study tells about how this package is useful to the users and its advantages and disadvantages, and also it tells whether this package is cost effective or not. There are three types of feasibility study, they are:

- Economical Feasibility.
- Technical Feasibility.
- Social Feasibility.

3.1.1 Economical Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available. There is nominal expenditure and economical feasibility for certain.

3.1.2 Technical Feasibility

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

3.1.3 Social Feasibility

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

3.2 Existing System

There are many applications like mobile applications, windows applications, and web applications. In mobile applications data is outsourced onto the cloud. Much of the data stored in cloud is highly sensitive like medical records and social networks. Security and privacy are thus very important issues in cloud computing. There is no security to the data which is stored in cloud. The cloud server leaks the sensitive information so that there is no privacy of data.

3.3 Proposed System

This project proposes Access control policies. Access control in cloud is gaining attention because it is important that only authorized users have access to valid service.

A huge amount of information is being stored in the cloud, and much of this is sensitive information. Care should be taken to ensure access control of this sensitive information which can important documents (as in Google Docs or Drop box), or which can often be related to health, or even personal information (as in social networking).

There are three types of access control policies:

- **User-based access control (UBAC)**
- **Role-based access control(RBAC)**
- **Attribute-based access control(ABAC)**

In UBAC, the access control list contains the list of users who are authorized to access data. This is based on the identity of the user or userid. This is not feasible in clouds where there are any users.

In RBAC, users are classified based on their individual roles or based on their job. Data can be accessed by users who have matching roles. The roles are defined by the system.

ABAC is more extended in scope, this is the extension of RBAC, in which users are given attributes, and the data has attached access policy.

When users have valid set of attributes, satisfying the access policy, at that only the user can access the data.

The proposed system employs access based policies to control the data and data security is ensured.

CHAPTER 4

SYSTEM ANALYSIS

The Software Requirement Specifications (SRS) will form the basis for the development and testing of the envisaged application. The SRS document is expected to describe all the functional requirements of the system without going into the implementation details. The SRS document will serve as the input document for design phase of the software project.

4.1 Modules:

- Out Sourcing the Data Module
- Access the Files from Cloud Module
- Private Profile Module
- Public Profile Module

4.2 Module Description:

4.2.1 Out Sourcing the Data Module:

The Upload module is used to upload the files in to the cloud. In this cloud provide some space to every user to upload files. The user can upload different file formats like documents, audio files, images and videos. In this module files stored in the specified locations like all images are in image folders etc. which are in the specified device like sdcards etc., in this user have two profiles one is Private profile and second one is public profile. Every time when you upload a file into cloud it asks which profile you want to upload by default the files stored in the public profile. In private profile you can upload any file formats which is one visible to you only. Where as in public the files which uploaded by you is visible to every registered user in the cloud.

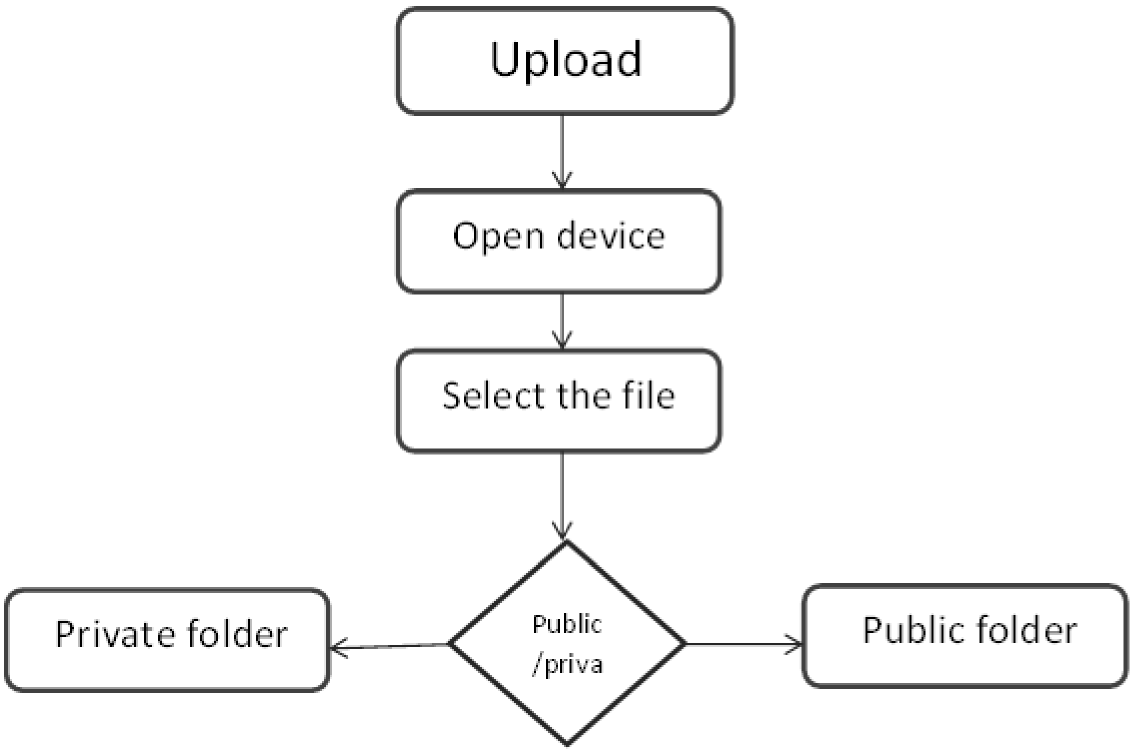


Figure 2.1 Flow Chart for Out Sourcing Data Module

4.2.2 Access the Files from Cloud Module:

Accessing the files from the cloud is another module in this work. In this module registered user can view the files in the cloud. User can view the private files and as well as public files based his wish. User can only view his private files in his private profile. In public profile user can view the all files which are uploaded by the public profile by all the users.

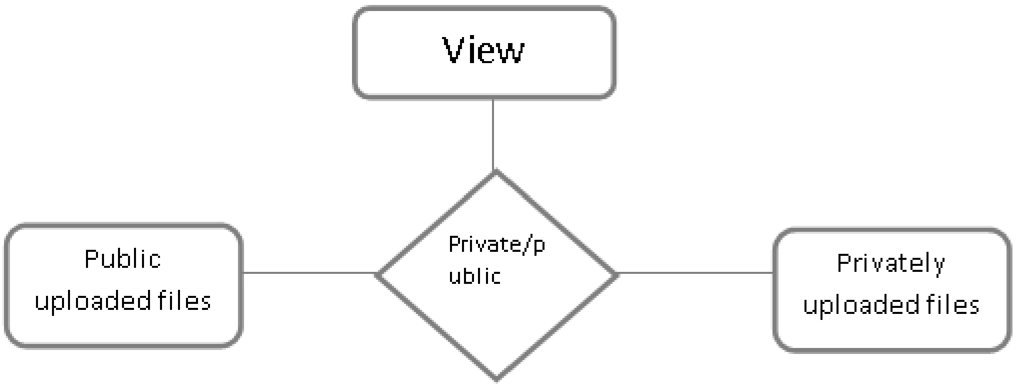


Figure 2.2 Flow Chart for Access the Files from Cloud Module

4.2.3 Private Profile Module:

The private profile provides security for the files which are uploaded by the users. In this profile the files only visible only to the respective profile users. The files in the private profiles stored in the respective locations user can upload all formats available in the market. Private profile provides two options for accessing files like download and delete files by default option is here downloading. The files which are more important only can upload in this profile for their secure. This profile provides delete file option to delete unnecessary files in the cloud and makes less cost in the cloud usage. The downloaded files are stored in the external cards like sd cards.

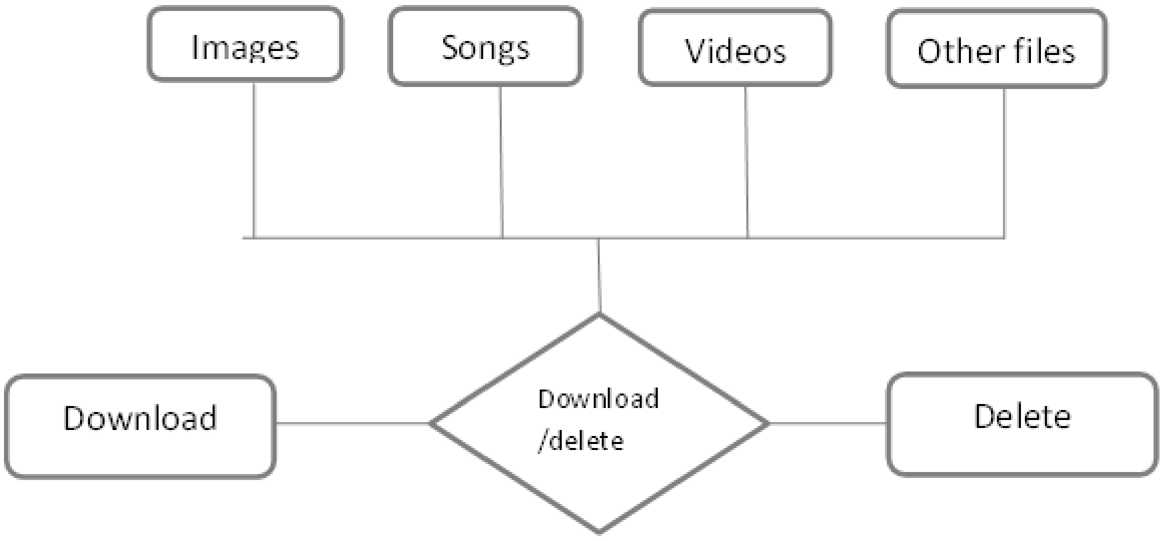


Figure 2.3 Flow Chart for Private Profile Module

4.2.4 Public Profile Module:

The public profile module is used to share the files in the cloud environment. The public profile is used to upload the files as visible to the all registered users in the cloud. This profile supports all formats available in the market User can view the all files in the cloud which are stored as public. In this user can download the files as he wish. The downloaded files are stored in the external sd cards in the mobile.

Public profile is default profile to upload in the cloud. In this data sorted based upon the file formats and stored in the specified file locations only.

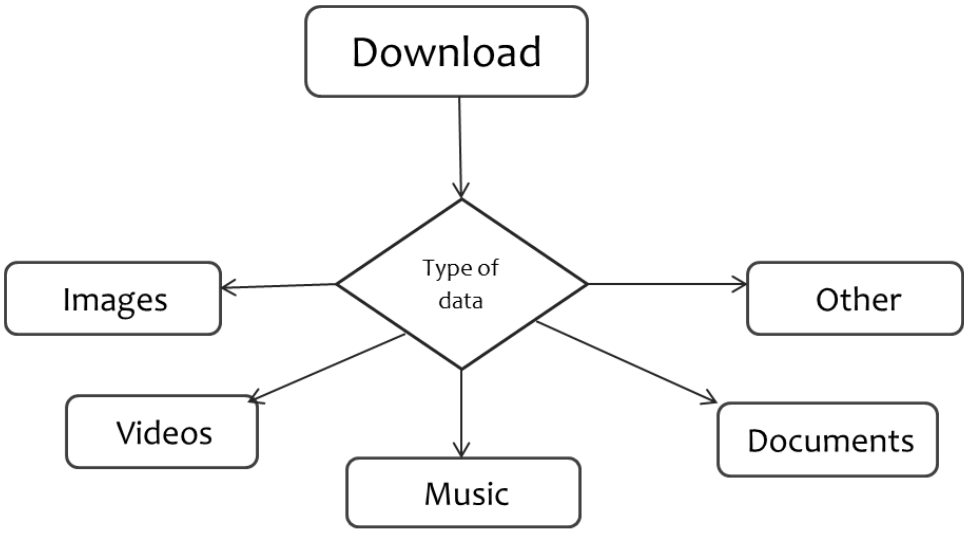


Figure 2.4 Flow Chart for Public Profile Module

3.3 Technology to be Used:

Programing Language:

Java is a powerful but lean object-oriented programming language. It has generated a lot of excitement because it makes it possible to program for Internet by creating Applets. Programs that can be embedded in web page. The context of an applet can be an animation with sound, an interactive game or a ticker tape. With constantly updated stock prices. Applets can be just little decorations to liven up web page, or they can be serious applications like Word processor or Spreadsheet.

But Java is more than a programming language for writing Applets. It is being used more and more for writing standalone applications as well. It is becoming so popular that many people believe it will become standard language for both general purpose and Internet programming.

Java is simple, elegant, and powerful and easy-to-use.

Java is actually a platform consisting of 3 components:

Java Programming Language.

Java Library of Classes and Interfaces.

Java Virtual Machine

To code, edit, debug and test the java programs, one needs to have a java development environment. At the minimum this will consists of a java compiler interpreter and applet viewer where applets can be tested. Sun’s java development kit (JDK) latest version is 2.2 can be freely downloaded from the Internet. Java compiler is available on DOS, Win95, WIN’NT, Solaris and MAC etc.

Data flow diagram is a graphical tool used to describe analyze the movement of data through a system manual or automated including the processes, stores of data, and delays in the system.

Android

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

The Android Development Tools (ADT) plug-in for Eclipse adds powerful extensions to the Eclipse integrated development environment. It allows you to create and debug Android applications easier and faster. If you use Eclipse, the ADT plug-in gives you an incredible boost in developing Android applications:

It gives you access to other Android development tools from inside the Eclipse IDE. For example, ADT lets you access the many capabilities of the DDMS tool: take screenshots, manage port-forwarding, set breakpoints, and view thread and process information directly from Eclipse.

It provides a New Project Wizard, which helps you quickly create and set up all of the basic files you'll need for a new Android application. It automates and simplifies the process of building your Android application.

Amazon Web Services (AWS)

AWS is a collection of remote computing services, also called web services that together make up a cloud computing platform by Amazon.com since 2006. The most central and well-known of these services are Amazon EC2 and Amazon S3. The service is advertised as providing a large computing capacity (potentially many servers) much faster and cheaper than building a physical server farm.

Amazon SimpleDB

Amazon SimpleDB has been accomplished with a clustered relational database that requires a sizable upfront investment, brings more complexity than is typically needed, and often requires a DBA to maintain and administer. In contrast, Amazon SimpleDB is easy to use and provides the core functionality of a database - real-time lookup and simple querying of structured data - without the operational complexity. Amazon SimpleDB requires no schema, automatically indexes your data and provides a simple API for storage and access. This eliminates the administrative burden of data modeling, index maintenance, and performance tuning. Developers gain access to this functionality within Amazon's proven computing environment, are able to scale instantly, and pay only for what they use.

Amazon S3

The Amazon Simple Storage Service (Amazon S3) is a scalable, high-speed, low-cost Web-based service designed for online backup and archiving of data and application programs. According to the Amazon Web services pages, the S3 was intentionally designed with a minimal feature set and was created to make Web-scale computing easier for developers. The service gives subscribers access to the same systems that Amazon uses to run its own Web sites. The S3 allows uploading, storage and downloading of practically any file or object. Amazon.com imposes no limit on the number of items that a subscriber can store. Subscriber data is stored on redundant servers in multiple data centers. The S3 employs a simple Web-based interface and uses encryption for the purpose of user authentication.

Subscribers can choose to keep their data private or make it publicly accessible. Users can also, if they so desire, encrypt data prior to storage. Rights may be specified for individual users. When a subscriber stores data on the S3, Amazon.com tracks usage for billing purposes but does not otherwise access the data unless required to do so by law.

Software Requirement:

Language	: JAVA
Operating System	: Microsoft Windows 7,Windows 8, Android
Cloud Framework	: Amazon Web Server

Hardware Requirements:

Processor: Dual core or high

Hard Disk: 160 GB

RAM: 2 GB

CHAPTER 5

SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development blends the perspective of marketing, design, and manufacturing into a single approach to product development, then design is the act of taking the marketing information and creating the design of the product to be manufactured. System design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

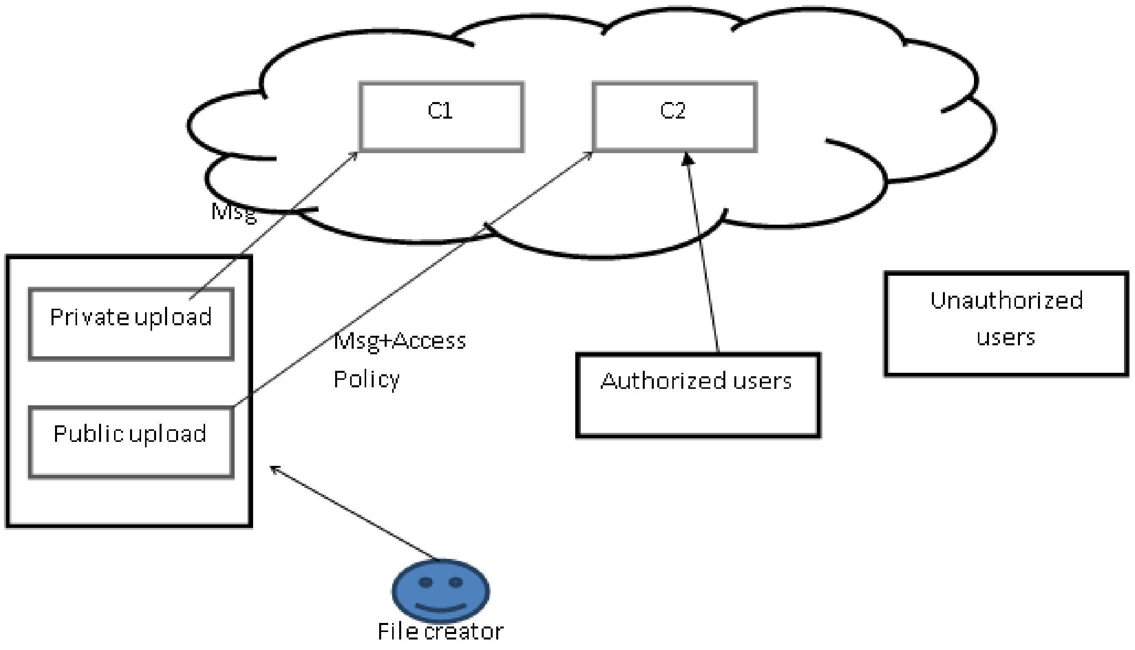


Figure 5.1 System Architeture

5.1 UML Diagrams:

5.1.1 Use Case diagram

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors. An actor is represents a user or another system that will interact with the system modeled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Here actors are the Group Managers and the Group Members. The system is the cloud. The following use case diagram shows the interaction between actors and the system. Some of the interactions are Registration, Group login and signature verification, Maintenance of Group Accounts and Group Modification.

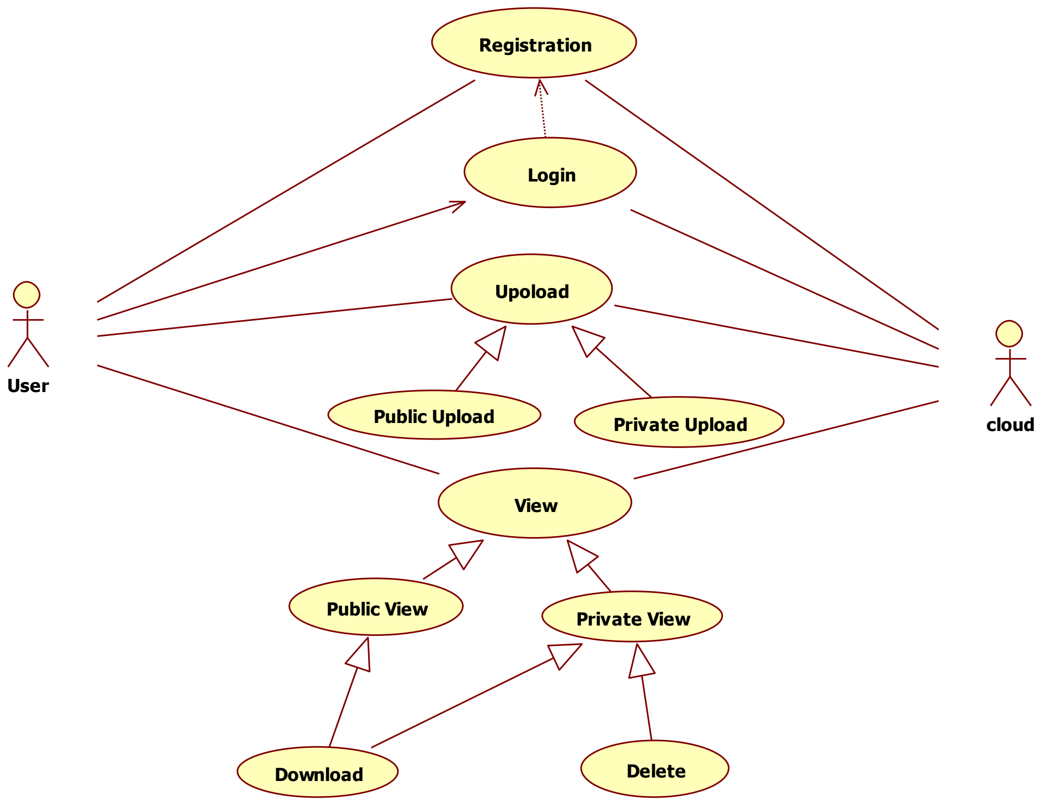


Fig 5.2: Use case Diagram

5.1.2 Class Diagram

Class diagram express the class model. Classes define the attribute values carried by each object and the operations that each object performs or undergoes.

The classes here are Group Member, Group Manager and Cloud. These classes along with their attributes and operations are illustrated below.

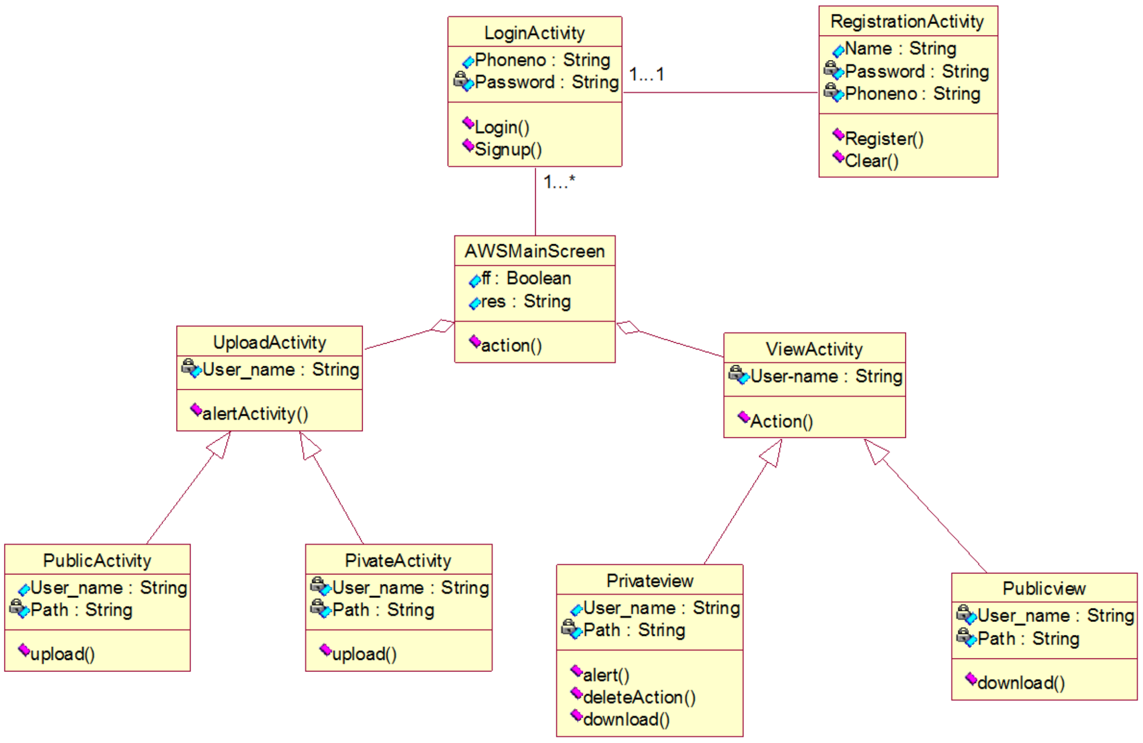


Fig 5.3: Class Diagram

5.1.3 Sequence Diagram

A sequence diagram is an interaction diagram that emphasizes the time ordering of the messages. Graphically, a sequence diagram is a table that shows objects arranged along the X-axis and messages, ordered in increasing time, along the Y-axis. Typically you place the object that initiates the interaction at the left and increasingly more sub-routine objects to the right. Next, you place the messages that these objects send and receive along the Y-axis, in order of increasing time from top to the bottom. This gives the reader a clear visual cue to the flow of control over time. Sequence diagrams have two interesting features:

- There is the object lifeline. An object lifeline is the vertical dashed line that represents the existence of an object over a period of time. Most objects that appear in the interaction diagrams will be in existence for the duration of the interaction, so these objects are all aligned at the top of the diagram, with their lifelines drawn from the top of the diagram to the bottom.
- There is a focus of the control. The focus of control is tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through the subordinate procedure. The top of the rectangle is aligns with the action; the bottom is aligned with its completion.

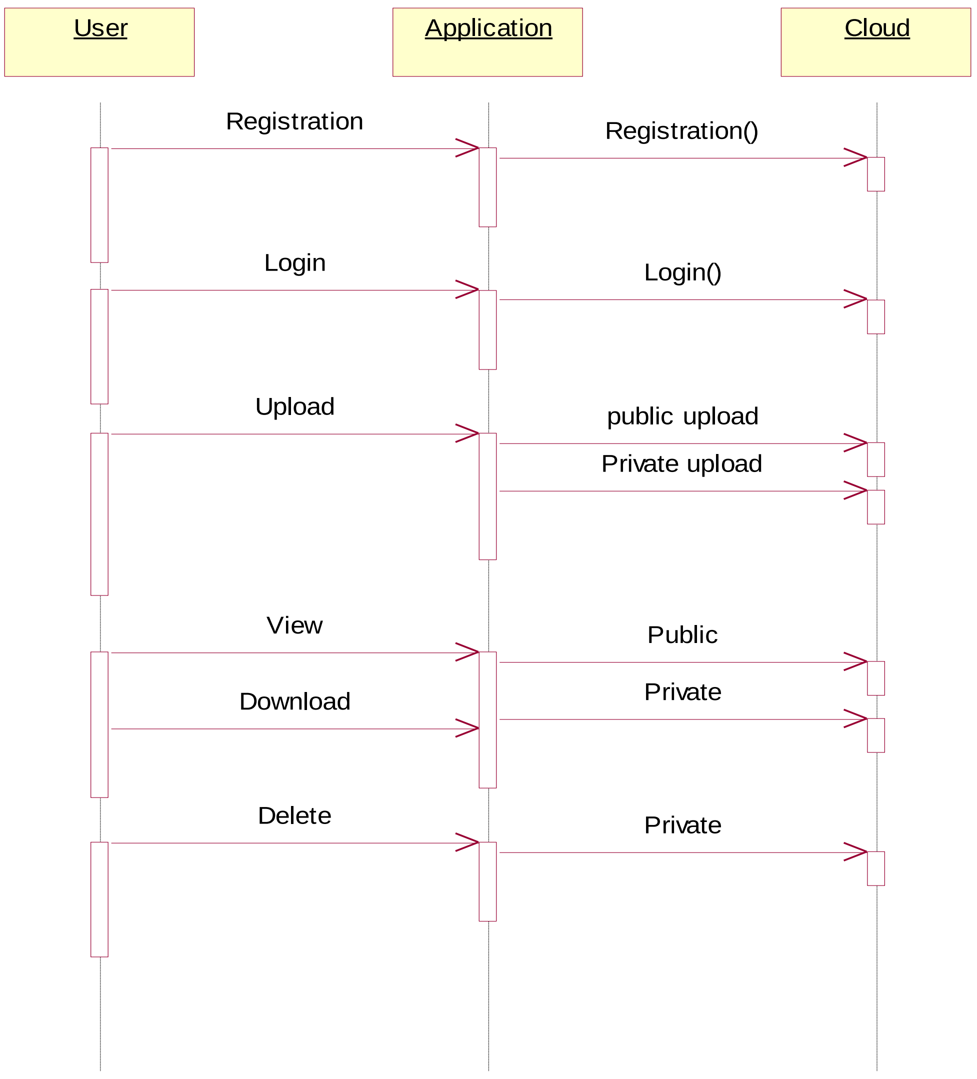


Fig 5.4 Sequence Diagram

5.2 Activity Diagram:

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

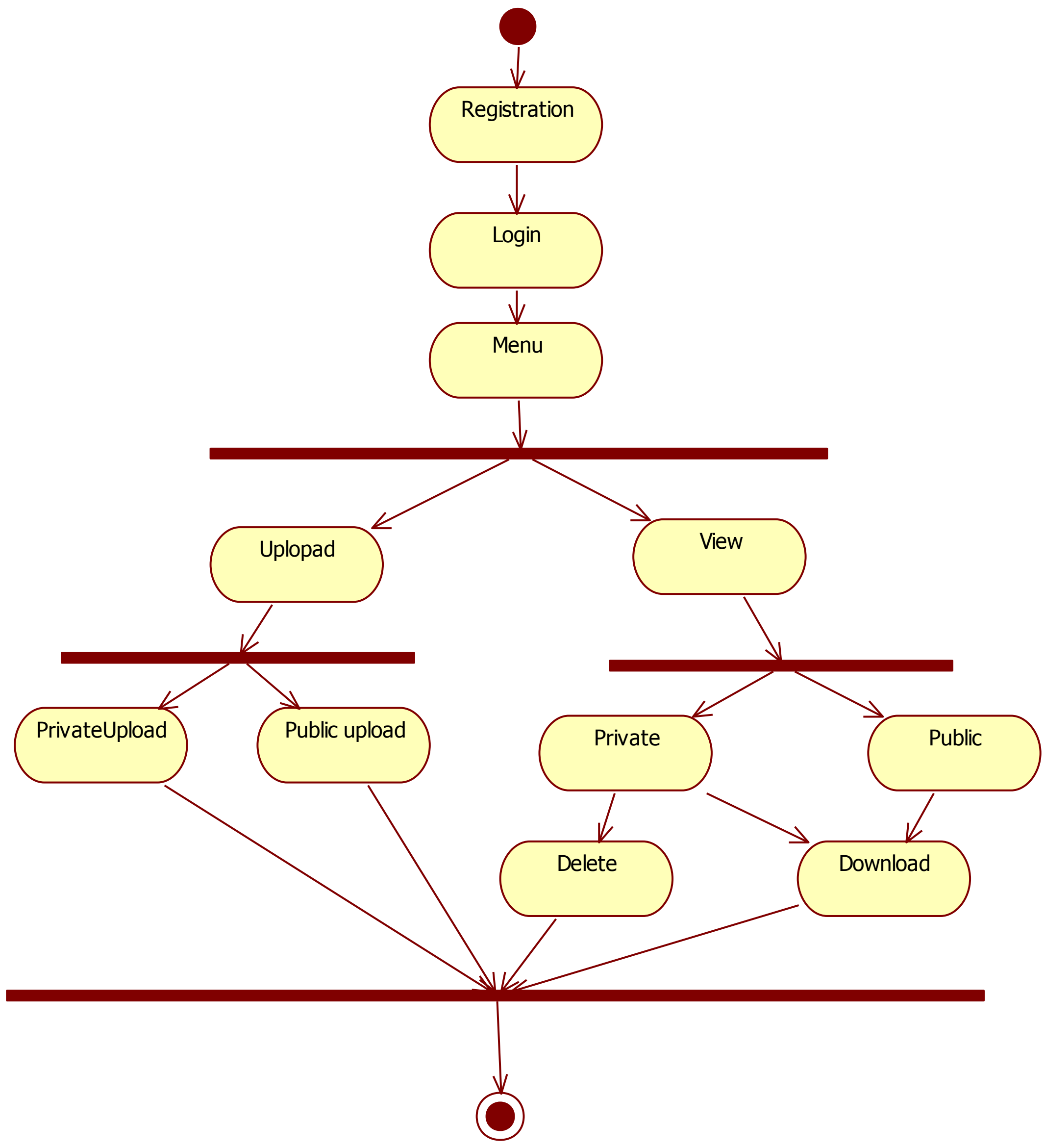


Fig 5.5: Activity Diagram

CHAPTER 6

SYSTEM TESTING

6.1 Introduction:

Testing is the major quality control measure employed during software development. Its basic function is to detect errors in the software. During requirement analysis and design, the output is a document that is usually textual and non-executable. After the coding phase, computer programs are available that can be executed for testing phases. This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during the previous phases. Thus, the goal of testing is to uncover requirement, design or coding errors in the programs.

Consequently, different levels of testing are employed. The starting point of testing is unit testing. In this a module is tested separately and is often performed by the coder himself simultaneously with the coding of the module. The purpose is to execute the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystem, which are then integrated themselves eventually form the entire system. During integration of modules, integration testing is performed. The goal of this testing is to detect design errors, while focusing on testing the interconnection between modules. After the system is put together, system testing is performed. Here the system is tested against tech system requirements to see if all the requirements are met and the system performs as specified by the requirements. Finally, acceptance testing is performed to demonstrate to the client, on the real life data of the client, the separation of the system.

For testing to be successful, proper selection of test cases is essential. There are two different approaches to selecting test cases-functional testing and structural testing. In functional testing the software for the module to be tested is treated as black box, and then test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called "black box testing". The focus is on testing the external behavior of the system. In structural testing the test cases are decided based on the logic of the module to be tested. Structural testing is sometimes called "glass box testing". Structural testing is used for lower levels of testing and functional testing is used for higher levels.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all the testing related activities that must be performed and specifies the schedule, allocates the resources, and specify guidelines for testing. The test plan specifies manner in which the modules will integrate together. Then for different test units, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit, the specified test cases are executed and actual result is compared with the expected output. The final output of the testing phases is to the text report and the error report, or set of such reports (one of each unit is tested). Each test report contains the set of such test cases and the result of executing the code with these test cases the error report describes the errors encountered and action taken to remove those errors.

6.2 Test Cases:

Unit Testing:

Unit testing is smallest unit of software design is a module. Unit testing is performed to check the functionality of these units. It is done before these modules are integrated together to build the overall system. Since the modules are small in size, individual programmers can do unit testing on their respective modules. So unit testing is basically white box oriented. Procedural design descriptions are used and control paths are tested to uncover errors within individual modules. Unit testing can be done for more than one module at a time.

The following are the tests that are performed during the unit testing:

- Module interface test: here it is checked if the information is properly flowing into the program unit and properly coming out of it.
- Local data structures: these are tested to see if the local data within unit(module) is stored properly by them.
- Boundary conditions: It is observed that much software often fails at boundary conditions. That's why boundary conditions are tested to ensure that the program is properly working at its boundary conditions.
- Independent paths: All independent paths are tested to see that they are properly executing their task and terminating at the end of the program.
- Error handling paths: These are tested to check if errors are handled properly by them.

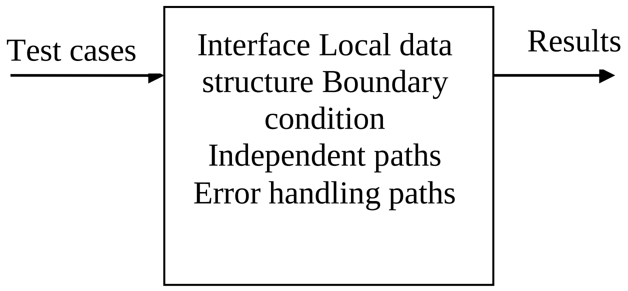

Test cases
Interface Local data structure Boundary condition Independent paths Error handling paths
Results

Fig 6.1 Unit Testing

Unit Testing Procedure:

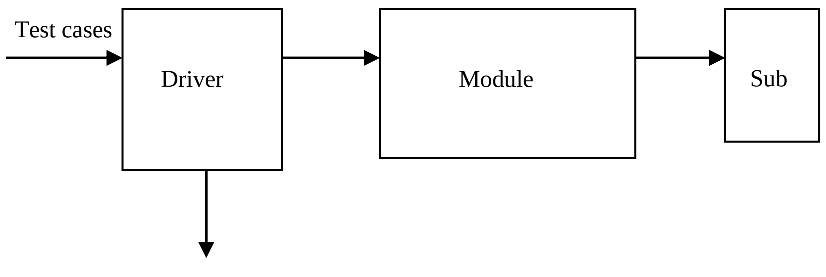


Fig 6.2 Unit Test Procedure

Unit testing begins after the source code is developed, reviewed and verified for the correct syntax. Here design documents help in making test cases. Though each module performs a specific task yet it is not a standalone program. It may need data from some other module or it may need to send some data or control information to some other module. Since in unit testing each module is tested individually, so the need to obtain data from other module or passing data to other module is achieved by the use of stubs and drivers. Stubs and drivers are used to simulate those modules. A driver is basically a program that accepts test case data and passes that data to the module that is being tested. It also prints the relevant results. Similarly stubs are also programs that are used to replace modules that are subordinate to the module to be tested. It does minimal data manipulation, prints verification of entry, and returns. Fig. 9.5 illustrates this unit test procedure.

Drivers and stubs are overhead because they are developed but are not a part of the product. This overhead can be reduced if these are kept very simple.

Once the individual modules are tested then these modules are integrated to form the bigger program structures. So next stage of testing deals with the errors that occur while integrating modules. That's why next testing done is called integration testing, which is discussed next.

Login Form:

Table 6.1 Test Case for Login Form

Test	Input	Expected Output	Actual Output	Result and Description
Valid login	Valid User Name and Password	Login Successfully	Login Successfully	Test Passed
Valid login	Either User name or Password or both give null	Login Unsuccessfully	Login Unsuccessfully	Test Passed
Valid login	Invalid User Name or valid	Login Unsuccessfully	Login Unsuccessfully	Test Passed

	Password			
Valid login	Valid User Name or invalid Password	Login Unsuccessfully	Login Unsuccessfully	Test Passed

Upload Files:

Table 6.2 Test Case for Upload Files

Test	Input	Uploading Type	Expected Output	Actual Output	Result and Description
Valid Upload	Image file	Private	Upload Successfully	Upload Successfully	Test Passed
Valid Upload	Image file	Public	Upload Successfully	Upload Successfully	Test Passed
Valid Upload	Songs file	Private	Upload Successfully	Upload Successfully	Test Passed
Valid Upload	Songs file	Public	Upload Successfully	Upload Successfully	Test Passed
Valid Upload	Videos file	Private	Upload Successfully	Upload Successfully	Test Passed
Valid Upload	Videos file	Public	Upload Successfully	Upload Successfully	Test Passed
Valid Upload	Other files	Private	Upload Successfully	Upload Successfully	Test Passed
Valid Upload	Other files	Public	Upload Successfully	Upload Successfully	Test Passed

Private View:

Table 6.3 Test Case for Private View

Test	Input	Expected Output	Actual Output	Result and Description
Image download	Select Image	Downloaded successfully	Downloaded successfully	Test Passed
Image delete	Select Image	Deleted successfully	Deleted successfully	Test Passed
Song download	Select Song	Downloaded successfully	Downloaded successfully	Test Passed
Song delete	Select Song	Deleted successfully	Deleted successfully	Test Passed
Video download	Select Video	Downloaded successfully	Downloaded successfully	Test Passed
Video delete	Select Video	Deleted successfully	Deleted successfully	Test Passed
Other file download	Select File	Downloaded successfully	Downloaded successfully	Test Passed
Other file delete	Select File	Deleted successfully	Deleted successfully	Test Passed

Public View:

Table 6.4 Test Case for public View

Test	Input	Expected Output	Actual Output	Result and Description
Image download	Select Image	Downloaded successfully	Downloaded successfully	Test Passed
Song download	Select Song	Downloaded successfully	Downloaded successfully	Test Passed
Video download	Select Video	Downloaded successfully	Downloaded successfully	Test Passed

Other file download	Select File	Downloaded successfully	Downloaded successfully	Test Passed

Chapter 7

SYSTEM IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on the user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user.

7.1 Sample Code:

APPENDIX-A

Sample Code for User Home Page:

```
public class AWSMainScreen extends Activity{
      Button
viewmyfiles,createfolder,upload1file,uploadmultifiles,download1file,downlo
admultifiles,help,statics,Contacts;
      InputStream in=null;
      NetworkInfo info=null;
    ProgressDialog dialog;
      String res="";
      boolean ff=true;
       List<String> list=new ArrayList();

       private Handler mHandler;
            private static final String fail = "Load Failed. Please Try
Restarting the Application.";


            public static BasicAWSCredentials credentials = null;

            protected Button s3Button;

            protected TextView welcomeText;

            private boolean credentials_found;
            String username;



      @Override
      protected void onCreate(Bundle savedInstanceState) {

            super.onCreate(savedInstanceState);
            setContentView(R.layout.awsmainscreen);
            username=getIntent().getStringExtra("cont");

              dialog = new ProgressDialog(this);
              dialog.setCancelable(true);
              dialog.setMessage("Getting Files Plz Wait...");
              dialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
              startGetCredentials();

            upload1file=(Button) findViewById(R.id.uploadsinglefile);
            download1file=(Button) findViewById(R.id.dowloadsiglefile);
            upload1file.setOnClickListener(new OnClickListener(){

                  public void onClick(View v) {
                        UploadFile();
                  }

            });


            download1file.setOnClickListener(new OnClickListener(){

                  public void onClick(View v) {
                        Intent                                  it=new
Intent(AWSMainScreen.this,DataView.class);
                        it.putExtra("un", username);

                        startActivity(it);
```

```
                                              }

              });

        private boolean verify() {
              Date d=new Date();
              //String mydate="14-Sep-11";
               DateFormat formatter ;
              String ss=GetbucketName.getDate();

              formatter = new SimpleDateFormat("dd-MM-yy");
              try {

                    Date mydate2=formatter.parse(ss);
                    int dd=mydate2.getDate();
                    int mm=mydate2.getMonth();
                    int yy=mydate2.getYear();
                    String ssdf=GettingFiles.verifyDate(dd, mm, yy);
                    Log.d("age ad", ssdf);

                    System.out.print(dd+mm+yy);
                    if(d.compareTo(mydate2)<0){

  Toast.makeText(getBaseContext(),  "Today  Date  is  Lesser  than  my  Date",
  20).show();
                       return true;
                    }
                    else if(d.compareTo(mydate2)>0){
  Toast.makeText(getBaseContext(),  "Today  Date  is  Greater  than  my  Date",
  20).show();
                             return false;
                          }
                    else{
  Toast.makeText(getBaseContext(), "Both all equal", 20).show();
                             return true;
                          }
              } catch (ParseException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }
              return false;

        }


        private void startGetCredentials() {
        Thread t = new Thread() {

              public void run(){
                try {
                    Properties properties = new Properties();
        properties.load(                      getClass().getResourceAsStream(
  "AwsCredentials.properties" ) );

                    String accessKeyId = properties.getProperty( "accessKey"
  );
                    String  secretKey  =  properties.getProperty(  "secretKey"
  );
```

```
                  if ( ( accessKeyId == null ) || ( accessKeyId.equals( ""
) ) ||
                         (   accessKeyId.equals(   "CHANGEME"   )   )   ||(
secretKey == null )   ||
                        ( secretKey.equals( "" ) ) || ( secretKey.equals(
"CHANGEME" ) ) ) {
                      Log.e(  "AWS",  "Aws  Credentials  not  configured
correctly." );
                        credentials_found = false;
                  } else {
                  credentials        =        new        BasicAWSCredentials(
properties.getProperty( "accessKey" ), properties.getProperty( "secretKey"
) );
                    credentials_found = true;
                  }

              }
              catch ( Exception exception ) {
                  Log.e( "Loading AWS Credentials", exception.getMessage()
);
                  credentials_found = false;
              }

            }
      };
      t.start();
    }


      protected void downloadsinglefile() {
            if(haveInternet()){
      dialog.show();
              Thread newThred=new Thread(){
                       public void run(){

                              try{
                                   //String
bname=GetbucketName.bucketName;


      //GettingFiles.getFiles("shivashiva");

                                   Intent                              it=new
Intent(AWSMainScreen.this,TabViewDownload.class);
                                   //Intent                            it=new
Intent(AWSMainScreen.this,View.class);

                                   startActivity(it);
                                   dialog.dismiss();
                                   } catch(Throwable e){
                                         e.printStackTrace();

                                   }

                             }
                         };
                         newThred.start();
      }
```

```
		else{
		Toast.makeText(getBaseContext(),     "Network     Unavailable",
Toast.LENGTH_LONG).show();

	}
	}
	protected void UploadFile() {
		Intent it=new Intent(this,TabMenu.class);
		it.putExtra("un", username);
		startActivity(it);

	}

	protected void viewFiles(String res2) {

		Intent   bucketViewIntent   =   new   Intent(AWSMainScreen.this,
S3BucketView.class);
		bucketViewIntent.putExtra( "bucketname", "mynameisshiva" );
		startActivity(bucketViewIntent);


	}


	protected void onDestroy() {

		super.onDestroy();
	}

	@Override
	protected void onPause() {
		//
		super.onPause();
	}

	@Override
	protected void onRestart() {
		//
		super.onRestart();
	}

	@Override
	protected void onResume() {
		//
		super.onResume();
	}

	@Override
	protected void onStart() {
		//
		super.onStart();
	}

	@Override
	protected void onStop() {
		//
		super.onStop();
	}
	@TargetApi(Build.VERSION_CODES.ECLAIR)
	@Override
	public void onBackPressed() {
		// TODO Auto-generated method stub
		super.onBackPressed();
	finish();
	}
	  private boolean haveInternet(){
		   info=
((ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE)).getA
ctiveNetworkInfo();
		    if (info==null || !info.isConnected()) {
			    return false;
		    }
		    else
		    return true;
	}

}
```

APPENDIX-B
Sample Code for Open SDcard Device in Mobile:

```
public class TabMenu extends TabActivity {
	String user_name;
	public void onCreate(Bundle savedInstanceState) {
	    super.onCreate(savedInstanceState);
	    setContentView(R.layout.tabxml);
	    user_name=getIntent().getStringExtra("un");
	    System.out.println("tab menu              "+user_name);

	    Resources  res  =  getResources();  //  Resource  object  to  get
Drawables
	    TabHost tabHost = getTabHost();  // The activity TabHost
	    TabHost.TabSpec spec;  // Resusable TabSpec for each tab
	    Intent intent;  // Reusable Intent for each tab

	    //  Create  an  Intent  to  launch  an  Activity  for  the  tab  (to  be
reused)
	    intent          =           new          Intent().setClass(this,
ImageTab.class).putExtra("u", user_name);



	    // Initialize a TabSpec for each tab and add it to the TabHost
	    spec = tabHost.newTabSpec("image").setIndicator("Images",
	                      res.getDrawable(R.drawable.ic_tab_artists))
	                  .setContent(intent);
	    tabHost.addTab(spec);

	    // Do the same for the other tabs
	    intent          =           new          Intent().setClass(this,
SongsTab.class).putExtra("u", user_name);
	    spec = tabHost.newTabSpec("songs").setIndicator("Songs",
	                      res.getDrawable(R.drawable.ic_tab_example))
	                  .setContent(intent);
	    tabHost.addTab(spec);
```

```
          intent          =          new          Intent().setClass(this,
VideoTab.class).putExtra("u", user_name);
          spec = tabHost.newTabSpec("video").setIndicator("Videos",
                            res.getDrawable(R.drawable.ic_tab_video))
                        .setContent(intent);
          tabHost.addTab(spec);

        intent = new Intent().setClass(this, OtherTab.class).putExtra("u",
user_name);
          spec = tabHost.newTabSpec("other").setIndicator("Others",
                            res.getDrawable(R.drawable.ic_tab_other))
                        .setContent(intent);
          tabHost.addTab(spec);

          tabHost.setCurrentTab(0);
      }
}
```

APPENDIX-C

Sample Code for Images Tab in Mobile:

```
public class ImageTab extends Activity {

            public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.cameratab);
            usern=getIntent().getStringExtra("u");
            System.out.println("userrrrrrrrrrr        nameeeeeeeeeeeeeeee
"+usern);
            pp=new PublicPrivate();
            pp.createDomain();
            dialog = new ProgressDialog(this);
            dialog.setCancelable(true);
            dialog.setMessage("Uploading Image Plz Wait...");
            dialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);

            c                    =                    managedQuery(
MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                    null, // Which columns to return
                    null ,     // Return all rows
                    null,
                    MediaStore.Images.Media._ID);
           int i=c.getColumnIndex(MediaStore.Images.Media.DATA);
           columnIndex                                                =
c.getColumnIndexOrThrow(MediaStore.Images.Media._ID);

           if(c.moveToFirst())
           {do{

               al.add(c.getString(i));
            }while(c.moveToNext());
           }
            GridView gridview = (GridView) findViewById(R.id.sdcard);
            gridview.setAdapter(new MyAdapter(this));
            gridview.setOnItemClickListener(new OnItemClickListener()
            {
                  public void onItemClick(AdapterView<?> parent, View v,
int position,
                        long id) {

                        String[]              projection            =
{MediaStore.Images.Media.DATA};
                        Cursor        cursor        =       managedQuery(
MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                        projection, // Which columns to return
                        null,       // Return all rows

                        null,

                        null);

                        columnIndex                                 =
cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
                        cursor.moveToPosition(position);
                        imagePath = cursor.getString(columnIndex);
                  alertbox("Confirmation ! Select The  Level?",imagePath);
                  }
              });
           }
      protected void alertbox(String title,final String imagePath)
         {
            final CharSequence[] items = {"private", "public"};

            AlertDialog.Builder          builder          =          new
AlertDialog.Builder(ImageTab.this);
            builder.setTitle(title);

            builder.setSingleChoiceItems(items,          -1,          new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int item) {

                  final String ssss1[]=imagePath.split("/");
                  int index1=imagePath.lastIndexOf("/");
                        final String str1=imagePath.substring(index1+1);
                  switch(item)
                 {
                     case 0:   f=false;   break;
                     case 1:   f=true;    break;
                       default :
                          f=true;
                          pp.AddToTable(str1,usern, "public");
                          break;
                  }
                 if(f==false)
                        pp.AddToTable(str1,usern, items[item].toString());

                  else
                        pp.AddToTable(str1,usern, items[item].toString());

                 }
            });
           builder.setPositiveButton("Ok",                            new
DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int id) {
```

```
                        System.out.println("helloooo "+imagePath);

                        uploading(imagePath);
                  }
         });
         builder.show();
}

         protected void uploading(final String imagePath) {
             if(haveInternet()){
             final String ssss[]=imagePath.split("/");
              int index=imagePath.lastIndexOf("/");
                   final String str=imagePath.substring(index+1);

             System.out.println("strrrrrrrrrrrrrrrrrrrrrrrrrrr "+str);
                   c.close();

      System.out.println("hiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
"+imagePath);

                   final File f=new File(imagePath);
                   Log.d("file", GetbucketName.bucketName+imagePath);
                   System.out.println("fileeeeeeeeeeeeeeeee  "+f);

                   final int size=GettingFiles.updateSize1(f);
                   Log.d("size", Integer.toString(size));
             if(size>0){
                     final Thread newThred=new Thread(){
      public void run(){

              try{

                        flag=S3.createObjectForBucket(
"AndroidAmazon",GetbucketName.getFoldername()+"/"+str ,f );
                   //    insertDB(str);
                        System.out.println("str "+str);
                        System.out.println("flag  for  image  is-----------
"+flag);
                        FileOutputStream
fos=openFileOutput("awsincoign2",Context.MODE_PRIVATE);
                        GettingFiles.updateFile(fos,size);

                        //finish();
                   } catch(Throwable e){
                        e.printStackTrace();

                   }


             }
         };
         newThred.start();
         Thread t2=new Thread(){
             public void run(){
                   try {
                              newThred.join();
                        } catch (InterruptedException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                        if(flag==true){
                              DBAdapterSongs                         d=new
DBAdapterSongs(ImageTab.this,"AWSImages");
                        d.open();
                        long i=d.insertTitle(str);
                        if(i>0){
                              Log.d("DataBase", "Inserted Sucessfuly");
                        }
                        else{
                              Log.d("DataBase",       "NOt       Inserted
Sucessfuly");
                        }
                        d.close();
                        }else{
                              Intent                                it=new
Intent(ImageTab.this,AlertActivity.class);
                              startActivity(it);
                        }
            }
        };t2.start();

            }else{
                  Toast.makeText(getBaseContext(),       "No       space",
Toast.LENGTH_LONG).show();

            }
            }
            else{
                  Toast.makeText(getBaseContext(), "Network Unavailable",
Toast.LENGTH_LONG).show();

            }
        }



            private boolean haveInternet(){
            info=
((ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE)).getA
ctiveNetworkInfo();
           if (info==null || !info.isConnected()) {
                   return false;
           }
           else
           return true;
    }
            public class MyAdapter extends BaseAdapter {

            private Context mContext;

            public MyAdapter(Context c) {
                  // TODO Auto-generated constructor stub
                  mContext = c;
            }

            public int getCount() {
                  // TODO Auto-generated method stub
                  return al.size();
            }

            public Object getItem(int arg0) {
                  // TODO Auto-generated method stub
                  return al.get(arg0);
            }

            public long getItemId(int arg0) {
                  // TODO Auto-generated method stub
                  return arg0;
            }

            public View getView(int position, View convertView, ViewGroup 
parent) {
                  // TODO Auto-generated method stub

                  View grid;

                  if(convertView==null){
                        grid = new View(mContext);
                        LayoutInflater inflater=getLayoutInflater();
                        grid=inflater.inflate(R.layout.mygrid,     parent, 
false);
                  }else{
                        grid = (View)convertView;
                  }

                  ImageView                  imageView                   = 
(ImageView)grid.findViewById(R.id.imagepart);
                  //TextView                  textView                   = 
(TextView)grid.findViewById(R.id.textpart);
                  Bitmap bm = ShrinkBitmap(al.get(position), 300, 300);

                  imageView.setImageBitmap(bm);
                  imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
                imageView.setPadding(8, 8, 8, 8);


                  return grid;
            }

      }
        Bitmap ShrinkBitmap(String file, int width, int height){

            BitmapFactory.Options      bmpFactoryOptions      =      new 
BitmapFactory.Options();
               bmpFactoryOptions.inJustDecodeBounds = true;
               Bitmap     bitmap     =     BitmapFactory.decodeFile(file, 
bmpFactoryOptions);

               int                      heightRatio                    = 
(int)Math.ceil(bmpFactoryOptions.outHeight/(float)height);
               int                      widthRatio                     = 
(int)Math.ceil(bmpFactoryOptions.outWidth/(float)width);

               if (heightRatio > 1 || widthRatio > 1)
               {
                if (heightRatio > widthRatio)
                {
                 bmpFactoryOptions.inSampleSize = heightRatio;
                } else {
                 bmpFactoryOptions.inSampleSize = widthRatio;
                }
               }

               bmpFactoryOptions.inJustDecodeBounds = false;
               bitmap = BitmapFactory.decodeFile(file, bmpFactoryOptions);
            return bitmap;
           }

}
```

APPENDIX-D

Sample Code for Private Image View:

```
public class PrivateImageActivity1 extends ListActivity {
      String u_name, status;
      String imagename = "";
      ArrayList<String> al = new ArrayList<String>();
      Context context;
      Cursor c=null;
      String ite ;
      boolean dbFlag=false, flag=true;
      InputStream is;
      ProgressDialog dialog;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            u_name = getIntent().getStringExtra("u");
            status = getIntent().getStringExtra("status");
            System.out.println("privateal              " + u_name
                        + "  status   " + status);
            dialog = new ProgressDialog(this);
        dialog.setCancelable(true);
        dialog.setMessage("Downloading Image Plz Wait...");
        dialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
       ClassLoader                      classloader                    =
PrivateImagesActivity.class.getClassLoader();
        PublicPrivate pp = new PublicPrivate();
            List<Others> list = pp.getAllValues(u_name, status);
            int len = list.size();
            System.out.println("size @@@@@@@@@@@@@" + len);
            for (int i = 0; i < len; i++) {

                  Others oo = list.get(i);
                  imagename = oo.getName();

                  if(imagename.contains("jpg")||imagename.contains("png"))
                  {
                  al.add(imagename);
                  }
                  System.out.println("image names       " + oo.getName()
                              + "===========");
            }

            ListView lv = getListView();
            lv.setAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1, al));


            lv.setOnItemClickListener(new OnItemClickListener() {
                  public   void   onItemClick(AdapterView<?>   parent,   View
view,
                              int position, long id) {
                        ite= ((TextView) view).getText().toString();

                        alertbox("Confirmation ! Select The  Level?",ite,
position);

                  }


            });

      }
      protected void alertbox(String title,final String imagePath, final
int pos)
         {
             final CharSequence[] items = {"delete", "download"};

             AlertDialog.Builder          builder          =           new
AlertDialog.Builder(PrivateImageActivity1.this);
      builder.setTitle(title);

      builder.setSingleChoiceItems(items,             -1,              new
DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int item) {

       final String ssss1[]=imagePath.split("/");
       int index1=imagePath.lastIndexOf("/");
                  final String str1=imagePath.substring(index1+1);


          switch(item)
          {
              case 0: flag=false; break;
              case 1: flag=true; break;
                default : flag=true; break;

          }

          }
      });
     builder.setPositiveButton("Ok", new DialogInterface.OnClickListener()
{
    public void onClick(DialogInterface dialog, int id) {
      if(flag==true)
      {
            downloadfile(imagePath,pos);
      }
      else if(flag==false)
      {
            downloadfile1(imagePath,ite);
```

```
             Intent        it=new        Intent(PrivateImageActivity1.this,
PrivateImagesActivity.class);
       startActivity(it);

      }
                 System.out.println("hellooooooooooooooooooooooooooo
"+imagePath);

}
});
builder.show();
         }

      private void downloadfile1(String imagePath, String ite) {
            // TODO Auto-generated method stub
            AWSCredentials   credentials   =   new   BasicAWSCredentials(
"AKIAIVP2GAQDZR5IFIFQ","jSiXRQo9kEdeqxVqrGQfV+TmPuEciQqTs4dz7/4y" );
            AmazonSimpleDBClient  sdbClient  =  new  AmazonSimpleDBClient(
credentials);
            DeleteAttributesRequest          dar          =          new
DeleteAttributesRequest("ImageView", ite);
            sdbClient.deleteAttributes( dar );
      }
      protected void downloadfile(String images, int posi){
            dialog.cancel();
               dialog.show();
             is = S3.getImageForObject("AndroidAmazon",
                        "lizy" + "/"
                                    + al.get(posi).toString().trim());
             String filepath = "/sdcard/AWS/aws";
                 File imgdir = new File(filepath);
                 File file = new File(filepath
                              + al.get(posi).toString().trim());


                 dialog.dismiss();
                 System.out.println("@@@@@@@@@@@@@@@@@@@@"+is);
                 if(file.exists()==false){
                  String path=imgdir.toString().toLowerCase();
                  String name=imgdir.getName().toLowerCase();
                  ContentValues values = new ContentValues(7);
                    values.put(Images.Media.TITLE,
al.get(posi).toString().trim());
                    values.put(Images.Media.DISPLAY_NAME,
al.get(posi).toString().trim());
                  values.put(Images.Media.DATE_TAKEN,                 new
Date().getTime());
                    values.put(Images.Media.MIME_TYPE, "image/jpg/png");
                    values.put(Images.ImageColumns.BUCKET_ID,
path.hashCode());
                    values.put(Images.ImageColumns.BUCKET_DISPLAY_NAME,
name);
                    values.put("_data",            filepath            +
al.get(posi).toString().trim());

                    ContentResolver          contentResolver          =
getApplicationContext().getContentResolver();
                    Uri                      uri                      =
contentResolver.insert(Images.Media.EXTERNAL_CONTENT_URI, values);
```

```
                     System.out.println("uriiiiiiiiiiiiiiiiiiiiiiii
"+uri);

                     OutputStream outStream = null;
                        try {
                              outStream                              =
contentResolver.openOutputStream(uri);

                        } catch (FileNotFoundException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                        Toast.makeText(getApplicationContext(),
"Downloaded sucessfully", 190).show();
                     byte[] buffer = new byte[1024];
                     int count;
                     try {
                              while ((count = is.read(buffer)) != -1) {
                                  if (Thread.interrupted() == true) {
                                   String          functionName          =
Thread.currentThread().getStackTrace()[2].getMethodName() + "()";
                                   throw   new   InterruptedException("The
function " + functionName + " was interrupted.");
                                  }
                                  outStream.write(buffer, 0, count);
                                 }

                        } catch (IOException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        } catch (InterruptedException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }

                  }
      }


}
```

APPENDIX-E
Sample Code for Public Image View:

```
public class PublicImagesActivity extends ListActivity {
      String u_name,status,imagename="";
      ArrayList<String> al = new ArrayList<String>();
      InputStream is;
      ProgressDialog dialog;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
             u_name=getIntent().getStringExtra("u");
             status=getIntent().getStringExtra("status");
             System.out.println("publicimages                 "+u_name   +"
status   "+status);
```

```
                PublicPrivate pp = new PublicPrivate();
                dialog = new ProgressDialog(this);
            dialog.setCancelable(true);
            dialog.setMessage("Uploading Image Plz Wait...");
            dialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
                     List<Others> list = pp.getAllPublicValues(u_name);
                     int len = list.size();
                     System.out.println("size @@@@@@@@@@@@@" + len);
                     for (int i = 0; i < len; i++) {

                           Others oo = list.get(i);
                           imagename = oo.getName();

         if(imagename.contains("jpg")||imagename.contains("png"))
                           {
                           al.add(imagename);
                           }

                           System.out.println("image                      names
"+oo.getName() + "===========" );
                     }


                     ListView lv = getListView();
                     lv.setAdapter(new ArrayAdapter<String>(this,
                                 android.R.layout.simple_list_item_1, al));
                     lv.setOnItemClickListener(new OnItemClickListener() {
                           public  void  onItemClick(AdapterView<?>  parent,
View view,
                                       int position, long id) {
                                 dialog.cancel();
                                 dialog.show();
                                 String       ite        =        ((TextView)
view).getText().toString();

                                 is = S3.getImageForObject("AndroidAmazon",
                                             "lizy" + "/"
                                                         +
al.get(position).toString().trim());

                                 String filepath = "/sdcard/AWS/aws";
                                 File imgdir = new File(filepath);
                                 File file = new File(filepath
                                             +
al.get(position).toString().trim());


      System.out.println("@@@@@@@@@@@@@@@@@@@@"+is);

                                  if(file.exists()==false){
                                  String
path=imgdir.toString().toLowerCase();
                                  String name=imgdir.getName().toLowerCase();
                                  ContentValues      values       =       new
ContentValues(7);
                                    values.put(Images.Media.TITLE,
al.get(position).toString().trim());
                                    values.put(Images.Media.DISPLAY_NAME,
al.get(position).toString().trim());
```

```
                              values.put(Images.Media.DATE_TAKEN,       new
Date().getTime());
                                 values.put(Images.Media.MIME_TYPE,
"image/jpg/png");
                                 values.put(Images.ImageColumns.BUCKET_ID,
path.hashCode());

values.put(Images.ImageColumns.BUCKET_DISPLAY_NAME, name);
                                 values.put("_data",       filepath       +
al.get(position).toString().trim());
                                 ContentResolver     contentResolver      =
getApplicationContext().getContentResolver();
                                 Uri                 uri                  =
contentResolver.insert(Images.Media.EXTERNAL_CONTENT_URI, values);

                                 OutputStream outStream = null;
                                    try {
                                          outStream                       =
contentResolver.openOutputStream(uri);
                                    } catch (FileNotFoundException e) {
                                          //  TODO  Auto-generated   catch
block
                                          e.printStackTrace();
                                    }
                                    dialog.dismiss();

      Toast.makeText(getApplicationContext(),  "Downloaded  sucessfully",
30).show();
                                 byte[] buffer = new byte[1024];
                                 int count;
                                 try {
                                          while ((count = is.read(buffer))
!= -1) {
                                              if  (Thread.interrupted()  ==
true) {
                                               String    functionName     =
Thread.currentThread().getStackTrace()[2].getMethodName() + "()";
                                               throw                    new
InterruptedException("The   function   "   +   functionName   +   "   was
interrupted.");
                                              }
                                              outStream.write(buffer,   0,
count);
                                             }

                                    } catch (IOException e) {
                                          //  TODO  Auto-generated   catch
block
                                          e.printStackTrace();
                                    } catch (InterruptedException e) {
                                          //  TODO  Auto-generated   catch
block
                                          e.printStackTrace();
                                    }
                                }
                  }

                  });
```

7.2 LOGIN PAGE

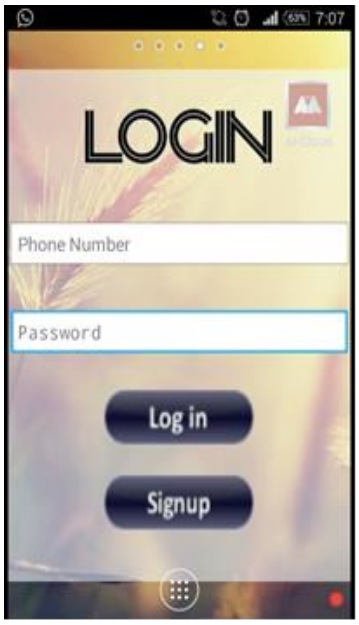


Figure 7.1 Screen Showing Login Page

The screen showing Login page which contains two fields such as phone number and password. The phone number and pasword fields are validated If user enters invalid details message box appears as "invalid Credentials".

7.3 REGISTRATION PAGE

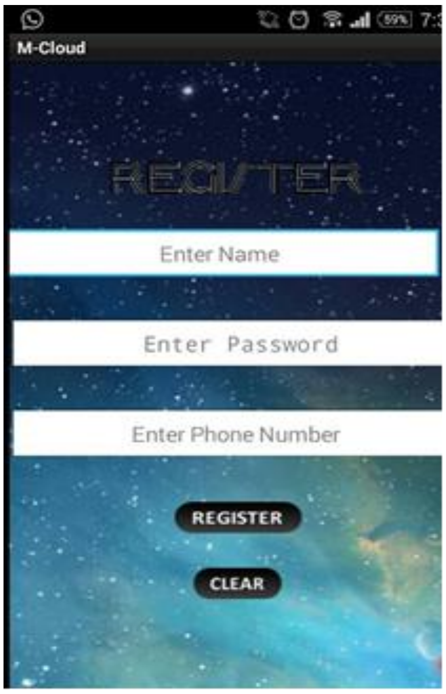


Figure 7.2 Screen ShowingRegistration Page

This Screen showing Registration page which contains fields like name,password, mobile number. All the fileds are validated. Password contains minimum two numeric values , minimum four special characters, And length should be more than thirteen characters.

7.4 USER PROFILE PAGE:

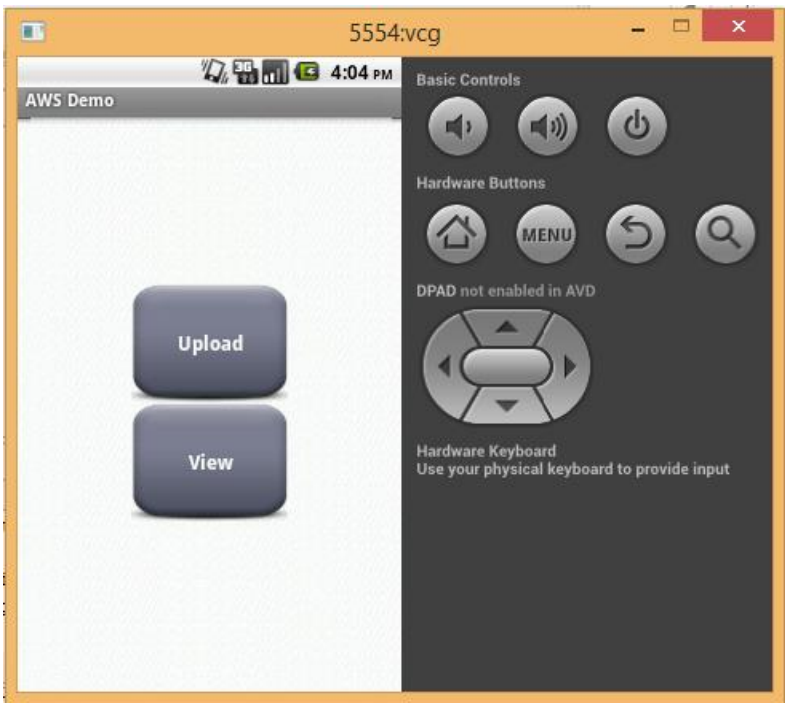
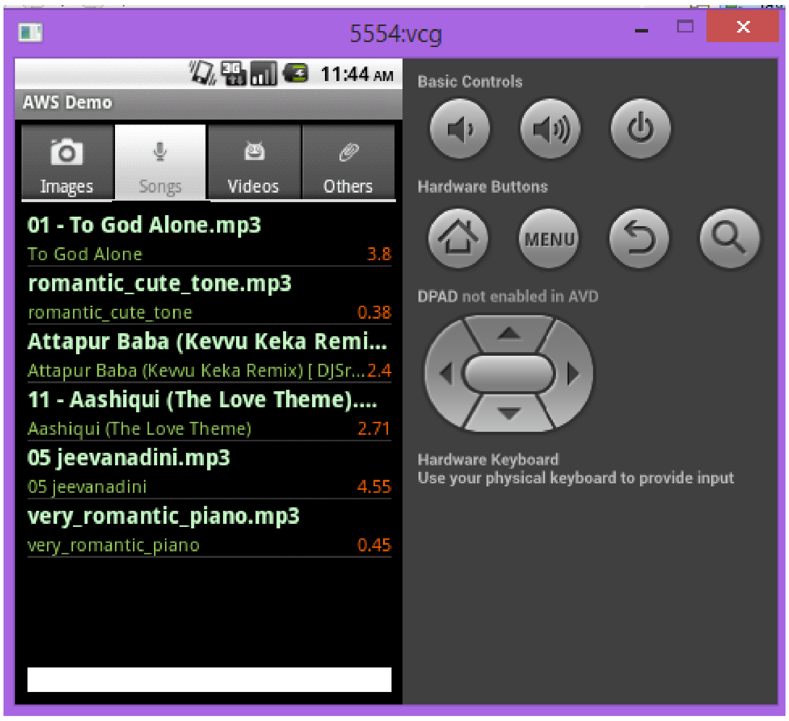


Figure 7.3 Screen Showing User Profile Page

The screen showing two buttons as "upload and "view". If user clicks upload button it directs to files storage page and user can upload files to cloud. If user clicks view button the cloud displays the files which are uploaded by the user.

7.5 FILE CONTENT PAGE:

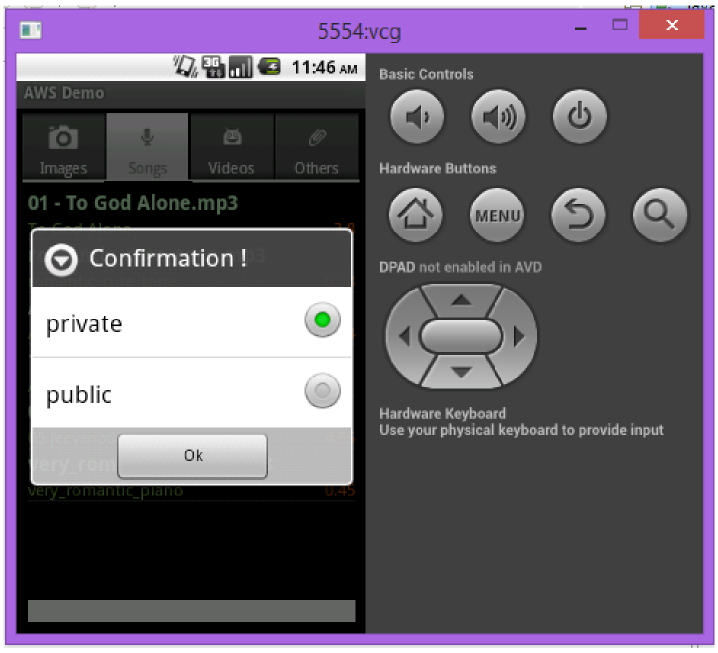


01 - To God Alone.mp3
To God Alone 3.8
romantic_cute_tone.mp3
romantic_cute_tone 0.38
Attapur Baba (Kevvu Keka Remi...
Attapur Baba (Kevvu Keka Remix) [DJSr...2.4
11 - Aashiqui (The Love Theme)....
Aashiqui (The Love Theme) 2.71
05 jeevanadini.mp3
05 jeevanadini 4.55
very_romantic_piano.mp3
very_romantic_piano 0.45

Figure 7.4 Screen ShowingFile Content Page on Mobile Device

The screen showing four fields like images,songs,videos and other content. The files are displayed by the cloud server.

7.6 TYPE OF UPLOADING PAGE



Confirmation !
private ●
public ○
Ok

Figure 7.5 Screen Showing Type of Uploading Page

The screen showing a confirmation box that contains two buttons such as "private" or "public". If user selects private then only private data shown to the user. If user selects public all public data is shown.

7.7 FILE UPLOAD PAGE

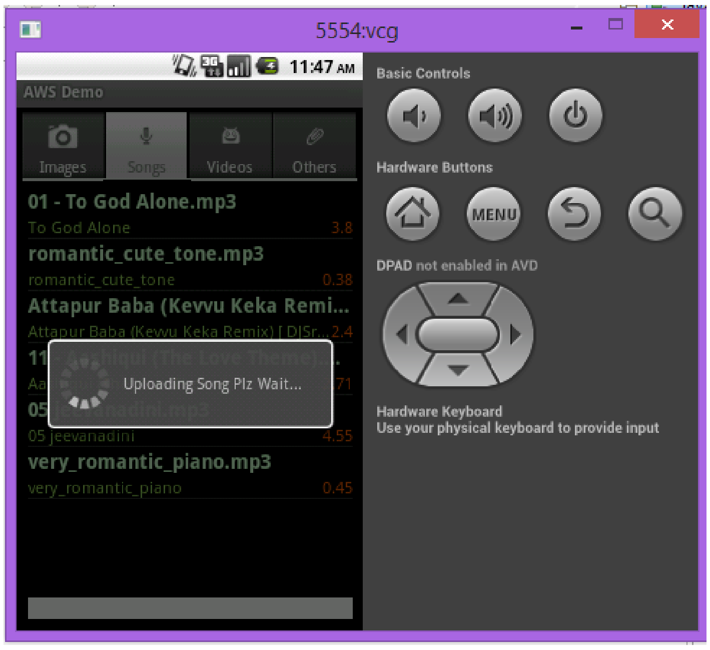


Figure 7.6 Screen Showing File Uploading Page

The screen showing that file is uploading to the cloud.

7.8 DISPLAY TYPE OF PROFILE PAGE

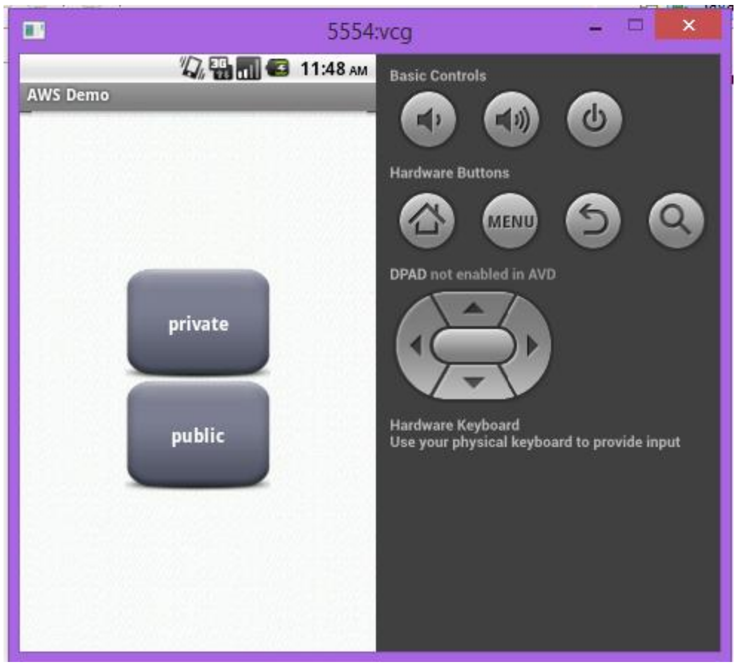


Figure 7.7 Screen Showing Display Profile Type Page

7.9 FILE ACCESS PAGE

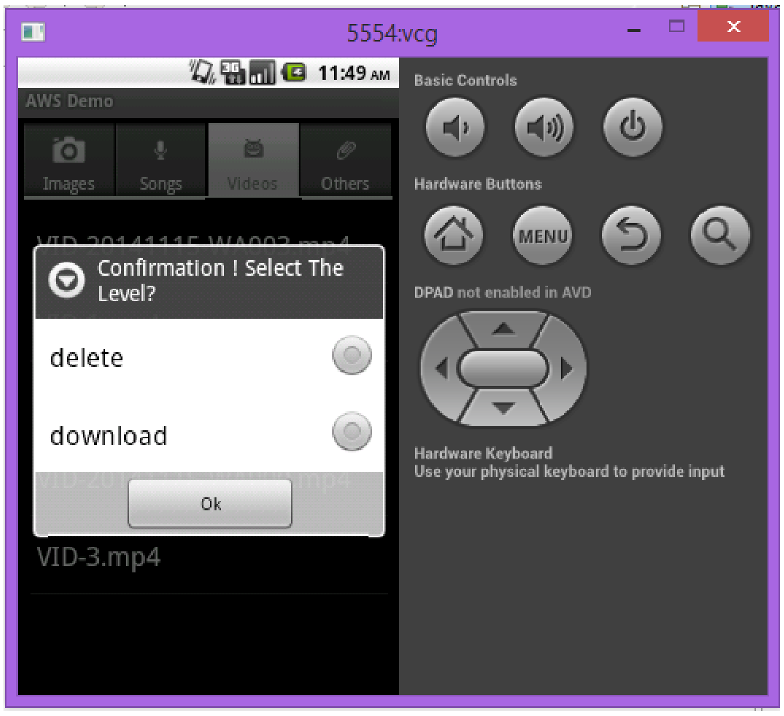


Figure 7.8 Screen Showing file Access Page

The screen showing that two fields delete and download which are shown only in the private profile. In this if user selects "delete" to delete the corresponding file is deleted. If users select download then, files are downloadeed directly to respective folder.

7.10 DATA VIEW PAGE

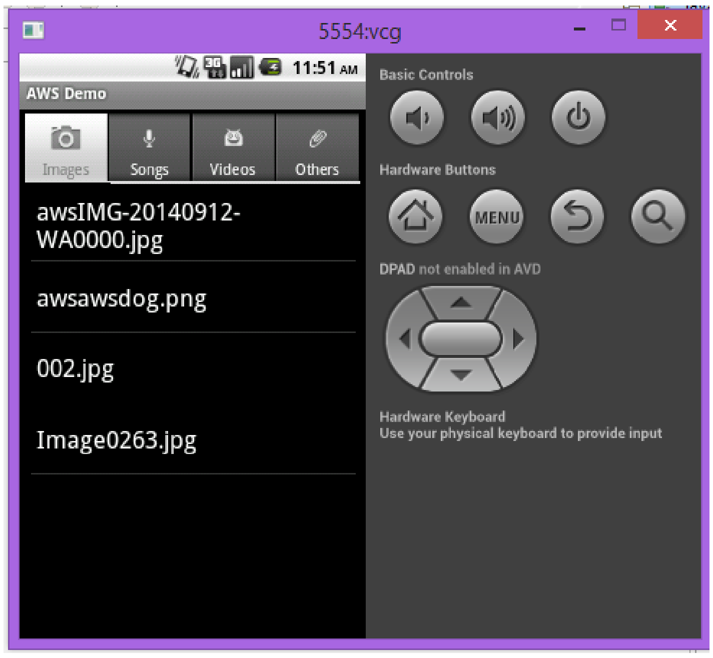


Figure 7.9 Screen Showing Data View Page

The screen showing files contents to the user. All the images, songs, videos etc are placed in separate folders.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

This problem solves the problem of sharing data to unwanted users. The cloud does not know the identity of the user who stores information, but only verifies the user's credentials. In mobile applications data is outsourced onto the cloud. Much of the data stored in cloud is highly sensitive like medical records and social networks. Security and privacy are thus very important issues in cloud computing. In this access control technique this project overcomes the limitation that the cloud knows the access policy for each record stored in the cloud. Access control in cloud is gaining attention because it is important that only authorized users have access to valid service. The proposed system employs access based policies to control the data and data security is ensured. The data is encrypted by the cloud provider for more secure, In future we would like to encrypt the data by user based.

.

BIBLIOGRAPHY

1. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," Proc. IEEE Symp. Security and Privacy, pp. 321-334, 2007.
2. X. Liang, Z. Cao, H. Lin, and D. Xing, "Provably Secure and Efficient Bounded Ciphertext Policy Attribute Based Encryption,"Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS), pp 343-352, 2009.
3. A.B. Lewko and B. Waters, "Decentralizing Attribute-Based Encryption," Proc. Ann. Int'l Conf. Advances in Cryptology (EUROCRYPT),pp. 568-588, 2011.
4. H.K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-Based Signatures: Achieving Attribute-Privacy and Collusion-Resistance," IACR Cryptology ePrint Archive, 2008.

5. Dan Boneh and BrentWaters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, TCC 2007, volume 4392 of Lecture Notes in Computer Science, pages 535–554. Springer, 2007.

6. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption upporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, EUROCRYPT 2008, volume 4965 of Lecture Notes in Computer Science, pages 146–162. Springer, 2008.
7. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption upporting disjunctions, polynomial equations, and inner products. J. ryptology, 26(2):191–224, 2013.
8. Kwangsu Lee and Dong Hoon Lee. Improved hidden vector encryption with short ciphertexts and tokens. Des. Codes Cryptography, 58(3):297–319, 2011.

9. Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, ASIACRYPT 2009, volume 5912 of Lecture Notes in Computer Science, pages 214–231. Springer, 2009.
10. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 591–608. Springer, 2012.

11. Jong Hwan Park. Inner-product encryption under standard assumptions. Des. Codes Cryptography, 58(3):235–257, 2011.
12. Jong Hwan Park, Kwangsu Lee, Willy Susilo, and Dong Hoon Lee. Fully secure hidden vector encryption under standard assumptions. Inf. Sci., 232:188–207, 2013.

13. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damg°ard, Leslie Ann Goldberg, Magn´us M. Halld´orsson, Anna Ing´olfsd´ottir, and Igor Walukiewicz, editors, ICALP 2008, volume 5126 of Lecture Notes in Computer Science, pages 560– 578. Springer, 2008.