

A Project Report
on
SECURE DATA SHARING IN CLOUD COMPUTING
USING REVOCABLE STORAGE IDENTITY BASED
ENCRYPTION

in partial fulfillment of requirements
for the award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

V. BHARGAVI

(16091A0515)

G. HARSHA VARDHAN REDDY

(16091A0535)

K. ANUSHA

(16091A0505)

D. JAYASAI REDDY

(16091A0542)

Under the Guidance of

Dr. N. MADHUSUDHANA REDDY Ph.D.,
Professor, Dept. of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Affiliated to JNTU Anantapuramu, APPROVED BY A.I.C.T.E., NEWDELHI,
ACCREDITED BY NAAC of UGC, NEWDELHI with "A+" grade,
ACCREDITED BY N.B.A, NEWDELHI,
NANDYAL-518501, (Estd-1995)
YEAR: 2019-2020

Dr. SUBBA REDDY KUNAM

Dr. Subba Reddy Kunam, M.Tech., Ph.D., MIEE, MISTE, MCSI, FICTE, FIC(II)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist), A.P.

Rajeev Gandhi Memorial College of Engineering & Technology
(AUTONOMOUS)

(Affiliated to JNTU Anantapuramu)

APPROVED BY A.I.C.T.E., NEWDELHI, ACCREDITED BY N.B.A, NEWDELHI,
NANDYAL-518501, (Estd-1995)



(ESTD – 1995)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **V. BHARGAVI** (16091A0515), **G. HARSHA VARDHAN REDDY** (16091A0535), **K. ANUSHA** (16091A0505) and **D. JAYASAI REDDY** (16091A0542) of final year B.Tech, C.S.E, have carried out the major project work entitled “**SECURE DATA SHARING IN CLOUD COMPUTING USING REVOCABLE STORAGE IDENTITY BASED ENCRYPTION**” under the supervision and guidance of **Dr. N. MADHUSUDHANA REDDY** ^{PhD.}, Professor, CSE Department, in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology in Computer Science & Engineering** from **Rajeev Gandhi Memorial College of Engineering & Technology (Autonomous)**, Nandyal is a bonafied record of the work done by them during 2019-2020.

Project Guide

Dr. N. Madhusudhana Reddy ^{Ph.D.},

Professor, Dept. of CSE

Head of the Department

Dr. K. Subba Reddy ^{M.Tech, Ph.D.}

Professor, Dept. of CSE

Place: Nandyal

Date:

External Examiner

Dr. SUBBA REDDY KUNAM

^{BE, M.Tech, Ph.D.} ^{MCS, FIETE, FIE(I)}
Professor & Head of the Department of CSE
RGM College of Engineering & Technology (Autonomous)
NANDYAL-518501, N.T. DIST. A.P.

Candidate's Declaration

We hereby declare that the project work entitled "**SECURE DATA SHARING IN CLOUD COMPUTING USING REVOCABLE STORAGE IDENTITY ENCRYPTION**" was carried out and written by us under the guidance of **Dr. N. Madhusudhana Reddy Ph.D.**, Professor, Dept. of Computer Science & Engineering, R.G.M. College of Engineering & Technology, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology**" in **Computer Science & Engineering**. We have not submitted the matter presented in this report anywhere for the award of any other Degree.

V. Bhargavi (16091A0515)

G. Harsha Vardhan Reddy (16091A0535)

K. Anusha (16091A0505)

D. Jayasai Reddy (16091A0542)

Dept. of CSE,
RGM CET.

Name of Guide:

Dr. N. Madhusudhana Reddy Ph.D.,
Professor,
Dept. of CSE

14

Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)

Professor & Head of the Department of CSE

RGM College of Engg. & Tech., (Autonomous)

NANDYAL-518 501, Kurnool (Dist), A.P.

ACKNOWLEDGEMENT

At the outset, we would be failing in our duties if we do not express our sincere gratitude to our guide & supervisor, internal guide **Dr. N. Madhusudhana Reddy, Professor** for the guidance and assistance to us, which contribute to successful completion of this project.

Our special thanks to **Dr. K. Subba Reddy, Head of the Department (CSE), Rajeev Gandhi Memorial College of Engineering & Technology**, for providing all the facilities and guidelines, required for our academic pursuit.

Our special thanks to **Dr. T. JAYACHANDRA PRASAD, Principal, Rajeev Gandhi Memorial College of Engineering & Technology**, for providing all the necessary facilities, required for our academic pursuit.

We would like to express our sincere and grateful thanks to the management of **Rajeev Gandhi Memorial College of Engineering & Technology**, under the leadership of **Dr. M. SANTHIRAMUDU, Chairman** for providing us an opportunity to fulfill our Aspirations.

BY

V. Bhargavi	(16091A0515)
G. Harsha Vardhan Reddy	(16091A0535)
K. Anusha	(16091A0505)
D. Jayasai Reddy	(16091A0542)

Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist), A.P.

Dr. SUBBA REDDY KUNAM
MISTE, MCSI, FIETE, FIE(I)
Department of CSE
Autonomous

ABSTRACT

Cloud computing provides a flexible and convenient way for data sharing, which brings various benefits for both the society and individuals. But there exists a natural resistance for users to directly outsource the shared data to the cloud server since the data often contain valuable information. Thus, it is necessary to place cryptographically enhanced access control on the shared data. Identity-based encryption is a promising cryptographical primitive to build a practical data sharing system. However, access control is not static. That is, when some user's authorization is expired, there should be a mechanism that can remove him/her from the system. Consequently, the revoked user cannot access both the previously and subsequently shared data. To this end, we propose a notion called revocable-storage identity-based encryption (RS-IBE), which can provide the forward/backward security of ciphertext by introducing the functionalities of user revocation and ciphertext update simultaneously. Furthermore, we present a concrete construction of RS-IBE, and prove its security in the defined security model. The performance comparisons indicate that the proposed RS-IBE scheme has advantages in terms of functionality and efficiency, and thus is feasible for a practical and cost-effective data-sharing system. Finally, we provide implementation results of the proposed scheme to demonstrate its practicability.

14
Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEF, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist), A.P.

CONTENTS

Chapter No	Title	Page No
	List of Abbreviations	
	List of Figures	
	List of Tables	
1.	INTRODUCTION	
1.1	What is cloud computing?	1
1.1.1	How Cloud Computing Works?	2
1.1.2	Characteristics	2
1.1.3	Benefits of Cloud Computing	3
1.1.4	Advantages	4
1.2	Objectives	4
2.	LITERATURE SURVEY	
2.1	Introduction	6
2.2	Related work	6
2.3	Existing System	7
2.4	Disadvantages of Existing System	8
2.5	Proposed System	8
2.6	Advantages of Proposed System	9
3.	SYSTEM DESIGN	
3.1	Modules	10
3.2	System architecture	10
3.3	UML diagrams	11
3.3.1	Use Case Diagram	12
3.3.2	Class Diagram	13
3.3.3	Sequence Diagram	14
3.3.4	Activity Diagram	16


Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL (M.C. Road), (Dist), A.P.

4.	IMPLEMENTATION	
4.1	Technologies Used	17
4.1.1	Java Technology	17
4.1.2	What Can Java Technology Do?	19
4.1.3	ODBC	21
4.1.4	JDBC	23
4.1.5	JDBC Goals	23
4.2	Sample code	26
4.3	Results	35
5.	SYSTEM TESTING	
5.1	Types of Tests	51
5.1.1	Unit Testing	51
5.1.2	Integration Testing	51
5.1.3	Functional Testing	52
5.1.4	System Testing	52
5.1.5	White Box Testing	53
5.1.6	Black Box Testing	53
5.1.7	Acceptance Testing	53
6.	CONCLUSION	55
7.	REFERENCES	56

14

Dr. SUBBA REDDY KUNAM
 BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
 Professor & Head of the Department of CSE
 RGM College of Engg. & Tech., (Autonomous)
 NANDYAL-518 501, Kurnool (Dist), A.P.

LIST OF ABBREVIATIONS

BFC-Big File Cloud Storage

UDP-User Datagram Protocol

14
Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist). A.P.

LIST OF FIGURES

Fig. 1.1 BFC Architecture.....	2
Fig. 3.3.1 Use Case Diagram....	16
Fig 3.3.2 Class Diagram.....	17
Fig 3.3.3 Sequence Diagram....	18
Fig 3.3.4 Collaboration Diagram....	19
Fig 3.3.5 Component Diagram.....	20

14
Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist), A.P.

LIST OF TABLES

Table 5.3: Failure System Test Cases.....	53
Table 5.3: Pass System Test Cases....	54

14

Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist). A.P.

LIST OF ABBREVIATIONS

GUI- Graphical User Interface
IP- Internet Protocol
IBE- Identity Based Encryption
Java VM- Java Virtual Machine
Java API- Java Application Programming Interface
JDBC- Java Database Connectivity
NIST- National Institute of Standards and Terminology
ODBC- Open Database Connectivity
PKI- Public Key Infrastructure
RMI- Remote Method Invocation
RDBMS- Relational Database Management
RIBE- Revocable Identity Based Encryption
SQL- Structured Query Language
TCP- Transmission Control Protocol
UDP- User Datagram Protocol

14
Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D. MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist), A.P.

LIST OF FIGURES

Fig. 1.1 Structure of Cloud Computing.....	1
Fig. 3.1 Use Case Diagram	13
Fig. 3.2 Class Diagram.....	14
Fig 3.3 Sequence Diagram	15
Fig. 3.4 Activity Diagram.....	16
Fig. 4.1 Java Compilation	18
Fig. 4.2 Java Virtual Machine	18
Fig. 4.3 Java Platform	19
Fig. 4.4 Java Database Connectivity	21
Fig. 4.5 Java Platform Execution	26
Fig. 4.6 Home Page.....	35
Fig. 4.7 Registration Page.....	36
Fig. 4.8 Data Provider Login Page	37
Fig. 4.9 Uploading Page	38
Fig. 4.10 Cloud DriveHQ Page.....	39
Fig. 4.11 Folder Page in DriveHQ	40
Fig 4.12 Viewing the Uploaded File Details Page.....	41
Fig. 4.13 Updating the File Page	42
Fig. 4.14 User Login Page	43
Fig. 4.15 User Sending Request Page.....	44
Fig. 4.16 Auditor Login Page	45
Fig. 4.17 Viewing the Data Provider Details Page.....	46
Fig. 4.18 Viewing the User Details Page.....	47
Fig. 4.19 Viewing the User Request Page	48
Fig. 4.20 Sending Secret Key Page.....	49
Fig. 4.21 Page Where User Enters Secret Key	50

h

Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)

Professor & Head of the Department of CSE

PGSR College of Engg. & Tech., (Autonomous)

518 501, Kurnool (Dist), A.P.

CHAPTER- 1

INTRODUCTION

1.1 What is cloud computing?

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

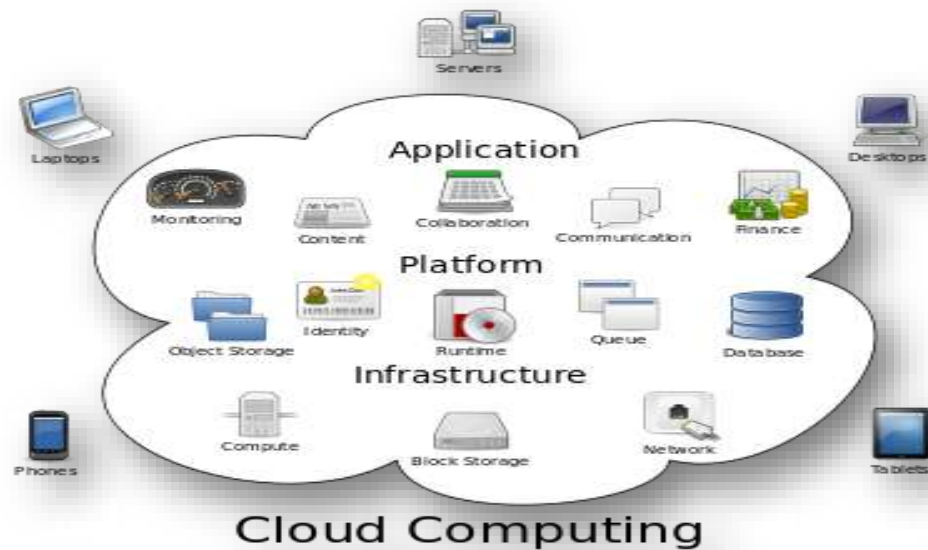


Figure 1.1: Structure of cloud computing

1.1.1 How Cloud Computing Works?

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

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

1.1.2 Characteristics:

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

- **On-demand self-service:** A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- **Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- **Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data

center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

- **Rapid elasticity:** Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- **Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

1.1.3 Benefits of cloud computing:

1. **Achieve economies of scale** – increase volume output or productivity with fewer people. Your cost per unit, project or product plummets.
2. **Reduce spending on technology infrastructure.** Maintain easy access to your information with minimal upfront spending. Pay as you go (weekly, quarterly or yearly), based on demand.
3. **Globalize your workforce on the cheap.** People worldwide can access the cloud, provided they have an Internet connection.
4. **Streamline processes.** Get more work done in less time with less people.
5. **Reduce capital costs.** There's no need to spend big money on hardware, software or licensing fees.
6. **Improve accessibility.** You have access anytime, anywhere, making your life so much easier!
7. **Monitor projects more effectively.** Stay within budget and ahead of completion cycle times.
8. **Less personnel training is needed.** It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.

9. **Minimize licensing new software.** Stretch and grow without the need to buy expensive software licenses or programs.
10. **Improve flexibility.** You can change direction without serious “people” or “financial” issues at stake.

1.1.4 Advantages:

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

1.2 OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

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

CHAPTER- 2

LITERATURE SURVEY

2.1 INTRODUCTION

Literature survey deals with the process of defining the functions of existing system. To create or develop a new system and study the prior system, Analysis difficult problems faced by that system. The disadvantages of existing system are discussed to prove the way of proposed system. Then the proposed system is defined for the problem and the advantages of the proposed system are also defined.

2.2 RELATED WORK

The concept of identity-based encryption was introduced by Shamir , and conveniently instantiated by Boneh and Franklin . IBE eliminates the need for providing a public key infrastructure (PKI). Regardless of the setting of IBE or PKI, there must be an approach to revoke users from the system when necessary, e.g., the authority of some user is expired or the secret key of some user is disclosed. In the traditional PKI setting, the problem of revocation has been well studied and several techniques are widely approved, such as certificate revocation list or appending validity periods to certificates. However, there are only a few studies on revocation in the setting of IBE. Boneh and Franklin first proposed a natural revocation way for IBE. They appended the current time period to the ciphertext, and non-revoked users periodically received private keys for each time period from the key authority. Unfortunately, such a solution is not scalable, since it requires the key authority to perform linear work in the number of non-revoked users. In addition, a secure channel is essential for the key authority and non-revoked users to transmit new keys. To conquer this problem, Boldyreva, Goyal and Kumar introduced a novel approach to achieve efficient revocation. They used a binary tree to manage identity such that their RIBE scheme reduces the complexity of key revocation to logarithmic (instead of linear) in the maximum number of system users. However, this scheme only achieves

selective security. Subsequently, by using the aforementioned revocation technique, Libert and Vergnaud proposed an adaptively secure RIBE scheme based on a variant of Water's IBE scheme, Chen et al. constructed a RIBE scheme from lattices. Recently, Seo and Emura proposed an efficient RIBE scheme resistant to a realistic threat called decryption key exposure, which means that the disclosure of decryption key for current time period has no effect on the security of decryption keys for other time periods. Inspired by the above work and, Liang et al. introduced a cloud-based revocable identity-based proxy re-encryption that supports user revocation and ciphertext update. To reduce the complexity of revocation, they utilized a broadcast encryption scheme to encrypt the ciphertext of the update key, which is independent of users, such that only non-revoked users can decrypt the update key. However, this kind of revocation method cannot resist the collusion of revoked users and malicious non-revoked users as malicious nonrevoked users can share the update key with those revoked users. Furthermore, to update the ciphertext, the key authority in their scheme needs to maintain a table for each user to produce the re-encryption key for each time period, which significantly increases the key authority's workload.

2.3 Existing System

- ❖ Boneh and Franklin first proposed a natural revocation way for IBE. They appended the current time period to the ciphertext, and non-revoked users periodically received private keys for each time period from the key authority.
- ❖ Boldyreva, Goyal and Kumar introduced a novel approach to achieve efficient revocation. They used a binary tree to manage identity such that their RIBE scheme reduces the complexity of key revocation to logarithmic (instead of linear) in the maximum number of system users.
- ❖ Subsequently, by using the aforementioned revocation technique, Libert and Vergnaud proposed an adaptively secure RIBE scheme based on a variant of Water's IBE scheme.
- ❖ Chen et al. constructed a RIBE scheme from lattices.

2.4 Disadvantages of Existing System:

- ❖ Unfortunately, existing solution is not scalable, since it requires the key authority to perform linear work in the number of non-revoked users. In addition, a secure channel is essential for the key authority and non-revoked users to transmit new keys.
- ❖ However, existing scheme only achieves selective security.
- ❖ This kind of revocation method cannot resist the collusion of revoked users and malicious non-revoked users as malicious non-revoked users can share the update key with those revoked users.
- ❖ Furthermore, to update the ciphertext, the key authority in their scheme needs to maintain a table for each user to produce the re-encryption key for each time period, which significantly increases the key authority's workload.

2.5 Proposed System:

- ❖ It seems that the concept of revocable identity-based encryption (RIBE) might be a promising approach that fulfills the aforementioned security requirements for data sharing.
- ❖ RIBE features a mechanism that enables a sender to append the current time period to the ciphertext such that the receiver can decrypt the ciphertext only under the condition that he/she is not revoked at that time period.
- ❖ A RIBE-based data sharing system works as follows:
- ❖ Step 1: The data provider (e.g., David) first decides the users (e.g., Alice and Bob) who can share the data. Then, David encrypts the data under the identities Alice and Bob, and uploads the ciphertext of the shared data to the cloud server.
- ❖ Step 2: When either Alice or Bob wants to get the shared data, she or he can download and decrypt the corresponding ciphertext. However, for an unauthorized user and the cloud server, the plaintext of the shared data is not available.

- ❖ Step 3: In some cases, e.g., Alice's authorization gets expired, David can download the ciphertext of the shared data, and then decrypt-then-re-encrypt the shared data such that Alice is prevented from accessing the plaintext of the shared data, and then upload the re-encrypted data to the cloud server again.

2.6 ADVANTAGES OF PROPOSED SYSTEM:

- ❖ We provide formal definitions for RS-IBE and its corresponding security model;
- ❖ We present a concrete construction of RS-IBE.
- ❖ The proposed scheme can provide confidentiality and backward/forward2 secrecy simultaneously
- ❖ We prove the security of the proposed scheme in the standard model, under the decisional ℓ -Bilinear Diffie-Hellman Exponent (ℓ -BDHE) assumption. In addition, the proposed scheme can withstand decryption key exposure
- ❖ The procedure of cipher text update only needs public information. Note that no previous identity-based encryption schemes in the literature can provide this feature;
- ❖ The additional computation and storage complexity, which are brought in by the forward secrecy, is all upper bounded by $O(\log(T)^2)$, where T is the total number of time periods.

CHAPTER- 3

SYSTEM DESIGN

3.1 MODULES:

- ❖ System Construction Module
- ❖ Data Provider
- ❖ Cloud User
- ❖ Key Authority (Auditor)

3.2 MODULES DESCRIPTION:

❖ System Construction Module

In the first module, we develop the proposed system with the required entities for the evaluation of the proposed model. The data provider (e.g., David) first decides the users (e.g., Alice and Bob) who can share the data. Then, David encrypts the data under the identities Alice and Bob, and uploads the cipher text of the shared data to the cloud server. When either Alice or Bob wants to get the shared data, she or he can download and decrypt the corresponding cipher text. However, for an unauthorized user and the cloud server, the plaintext of the shared data is not available.

❖ Data Provider

In this module, we develop the Data Provider module. The data provider module is developed such that the new users will Signup initially and then Login for authentication. The data provider module provides the option of uploading the file to the Cloud Server. The process of File Uploading to the cloud Server is undergone with Identity-based encryption format. Data Provider will check the progress status of the file upload by him/her. Data Provider provided with the features of Revocation and Ciphertext update the file. Once after completion of the process, the Data Provider logout the session.

❖ Cloud User

In this module, we develop the Cloud User module. The Cloud user module is developed such that the new users will Signup initially and then Login for authentication. The Cloud user is provided with the option of file search. Then cloud user feature is added up for send the Request to Auditor for the File access. After getting decrypt key from the Auditor, he/she can access to the File. The cloud user is also enabled to download the File. After completion of the process, the user logout the session.

❖ Key Authority (Auditor)

Auditor Will Login on the Auditor's page. He/she will check the pending requests of any of the above person. After accepting the request from the above person, he/she will generate master key for encrypt and Secret key for decrypt. After the complete process, the Auditor logout the session.

3.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.3.1 USE CASE DIAGRAM:

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

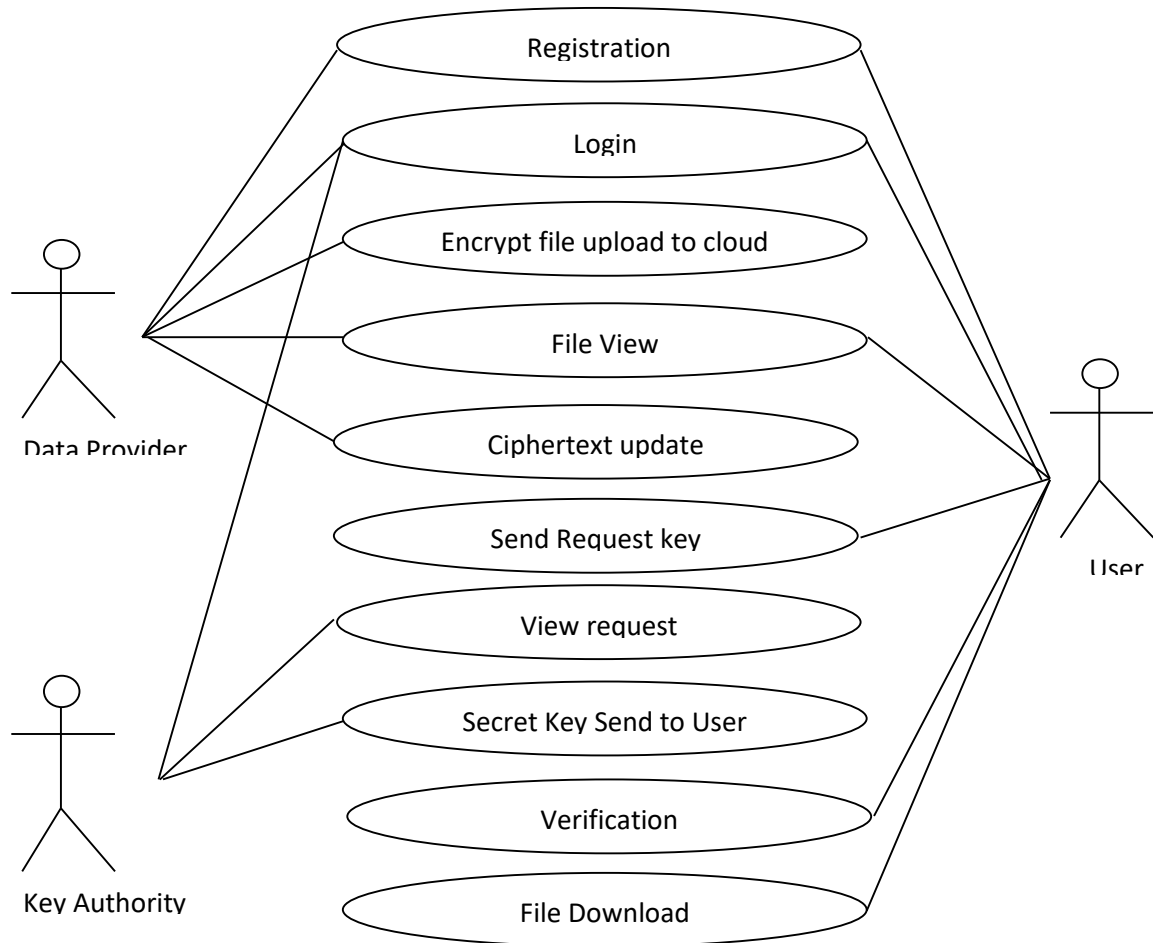


Figure 3.1: Use case diagram.

3.3.2 CLASS DIAGRAM:

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

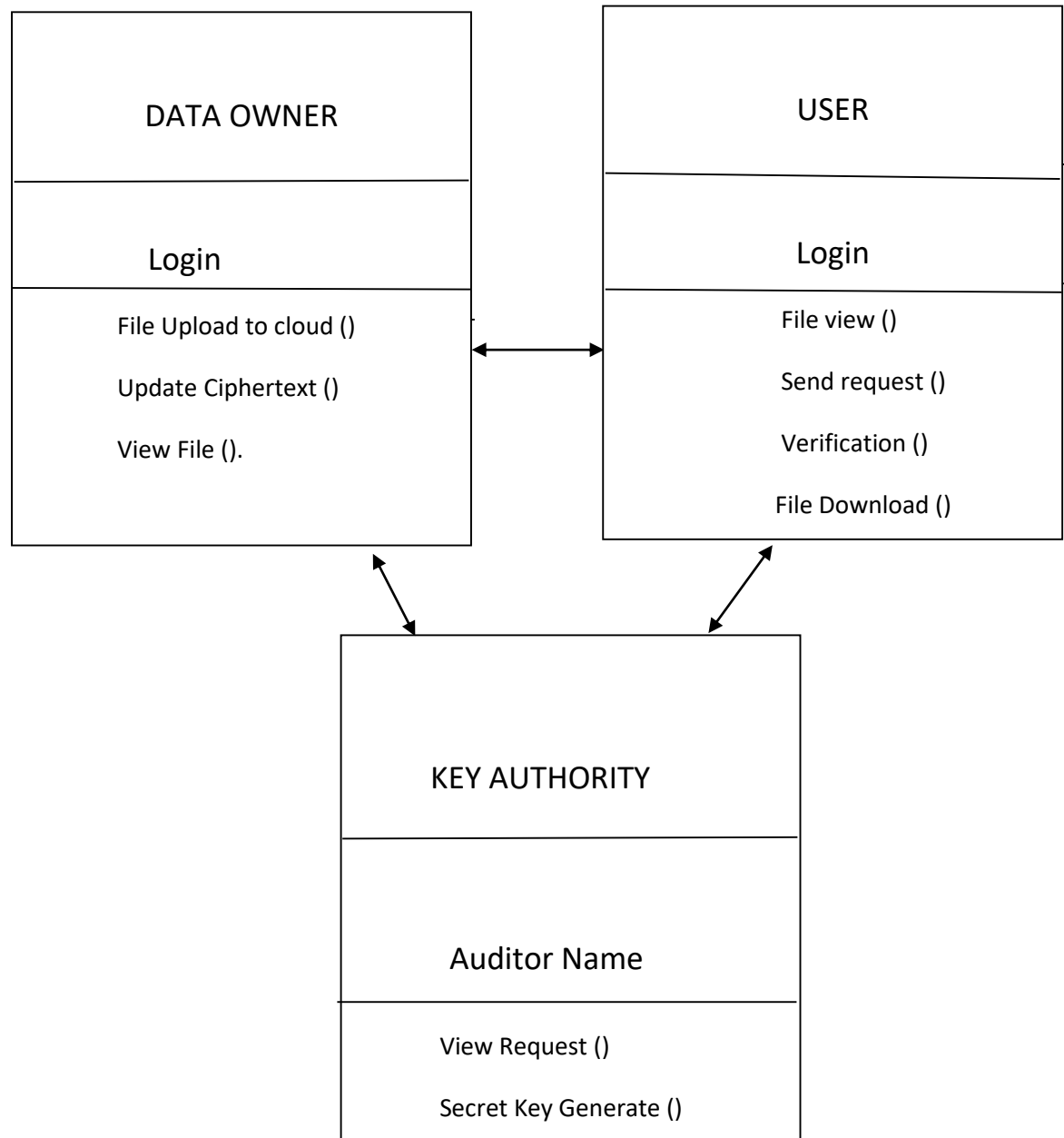


Figure 3.2: Class diagram.

3.3.3 SEQUENCE DIAGRAM:

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

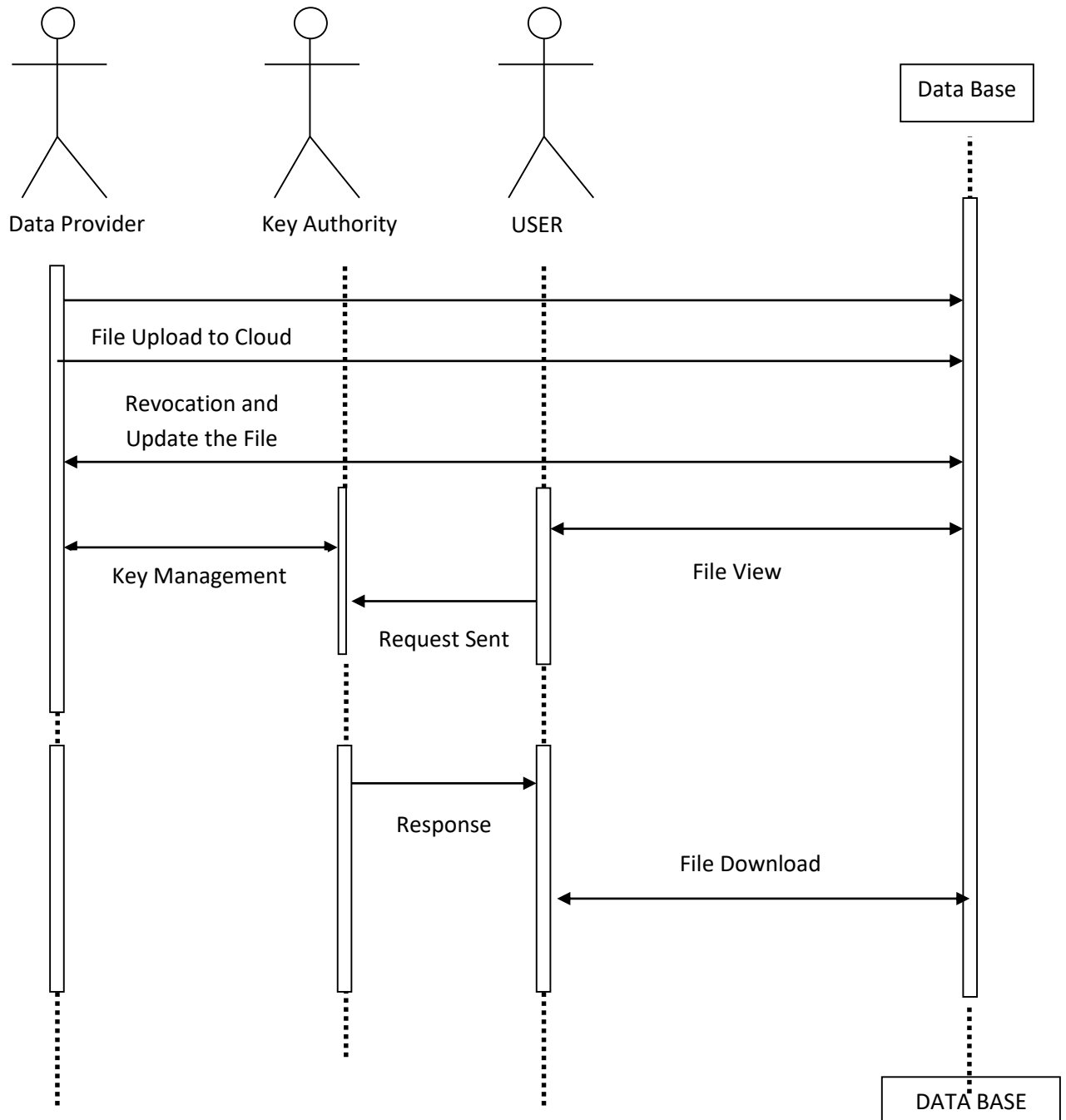


Figure 3.3: Sequence diagram.

3.3.4 ACTIVITY DIAGRAM:

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

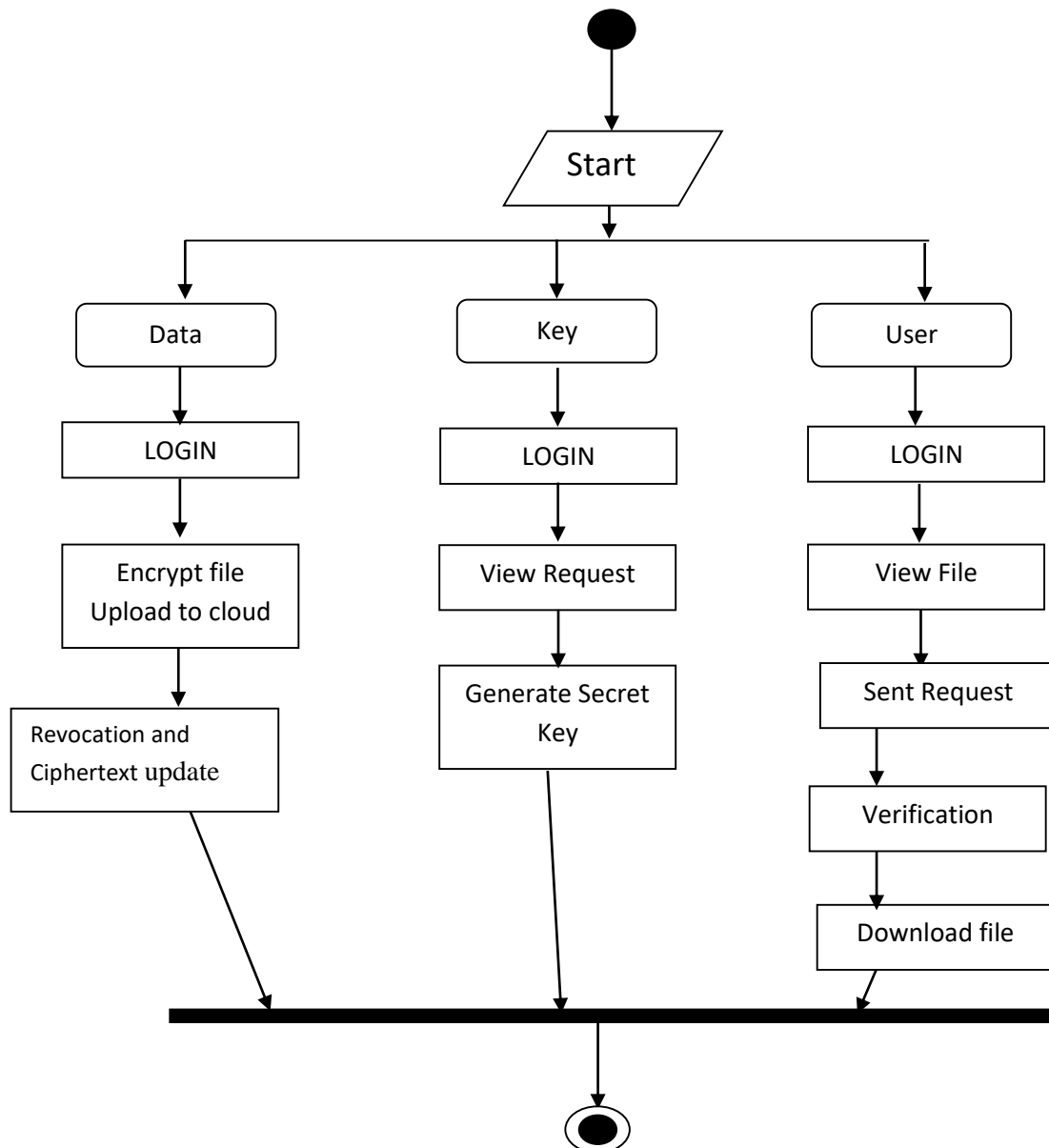


Figure 3.4: Activity diagram

CHAPTER- 4

IMPLEMENTATION

4.1 Technologies used:

4.1.1 Java Technology

Java technology is both a programming language and a platform.

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

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

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

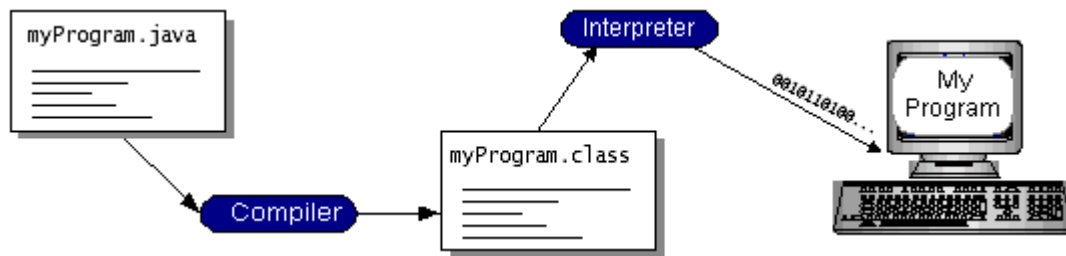


Figure 4.1: Java compilation

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

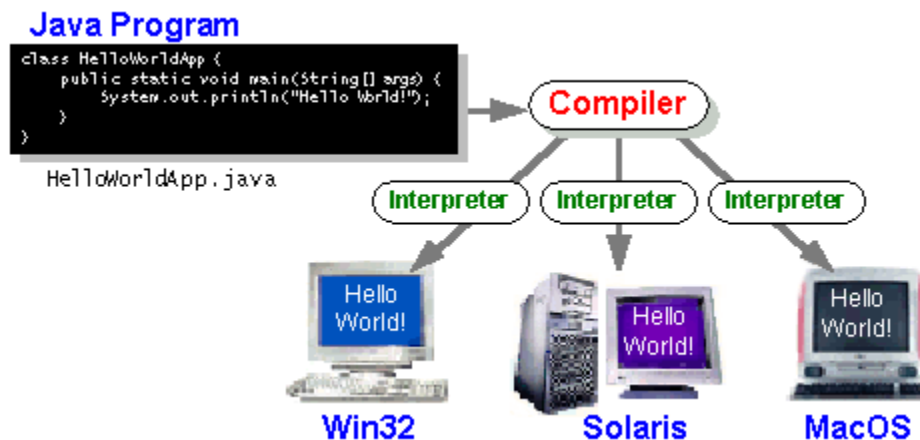


Figure 4.2: Java Virtual machine

The Java Platform:

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the

operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

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

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

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

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

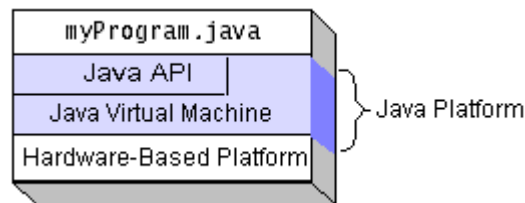


Figure 4.3: Java platform

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

4.1.2 What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar

with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

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

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

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

- **The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets:** The set of conventions used by applets.
- **Networking:** URLs, TCP (Transmission Control Protocol), UDP (User Datagram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security:** Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components:** Known as JavaBeansTM, can plug into existing component architectures.

- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™):** Provides uniform access to a wide range of relational databases.

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

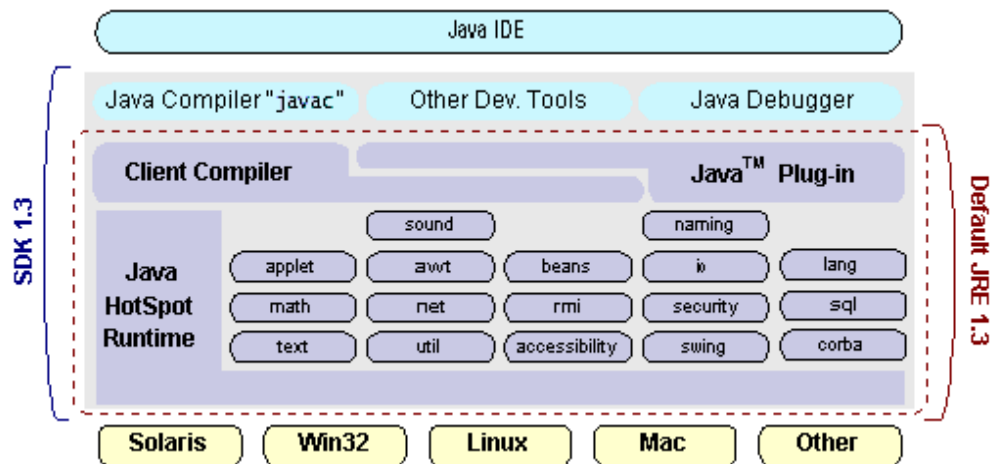


Figure 4.4: Java database connectivity.

4.1.3 ODBC

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

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will

lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources. From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

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

compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

4.1.4 JDBC

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

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

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

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

4.1.5 JDBC Goals

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

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

1. **SQL Level API**

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

2. **SQL Conformance**

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

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

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

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

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

5. **Keep it simple**

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

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

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

7. **Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database

Java has two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple	Architecture-neutral
Object-oriented	Portable
Distributed	High-performance
Interpreted	multithreaded
Robust	Dynamic
Secure	

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

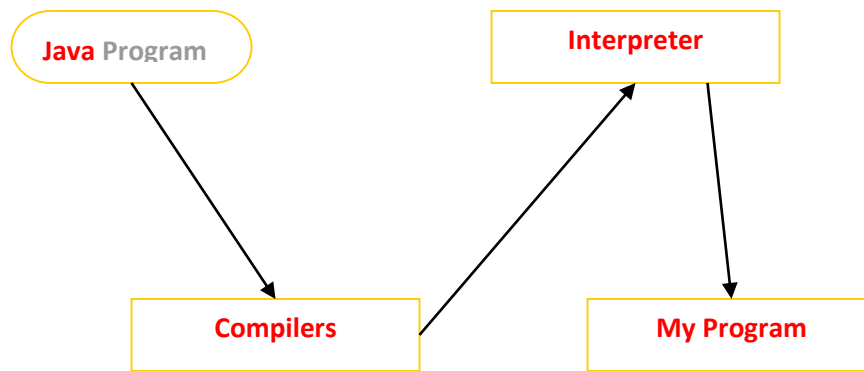


Figure 4.5: Java platform execution

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware. Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

4.2 SAMPLE CODE:

Reg.jsp:

```
<% @page contentType="text/html" pageEncoding="UTF-8"% >

<!DOCTYPE html>

<html>

<head>

<style>
```

```
.inputs {  
    background: #f5f5f5;  
  
    font-size: 0.8rem;  
  
    -moz-border-radius: 3px;  
  
    -webkit-border-radius: 3px;  
  
    border-radius: 3px;  
  
    border: none;  
  
    padding: 10px 10px;  
  
    width: 200px;  
  
    margin-bottom: 20px;  
  
    box-shadow: inset 0 2px 3px rgba( 0, 0, 0, 0.1 );  
  
    clear: both;  
  
}  
  
.inputs:focus {  
  
    background: #fff;  
  
    box-shadow: 0 0 0 3px #fff38e, inset 0 2px 3px rgba( 0, 0, 0, 0.2 ), 0px 5px 5px  
    rgba( 0, 0, 0, 0.15 );  
  
    outline: none;  
  
}  
  
.inputss {  
  
    background: #f5f5f5;  
  
    font-size: 0.8rem;
```

```
-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

border: none;

padding: 10px 10px;

width: 220px;

margin-bottom: 20px;

box-shadow: inset 0 2px 3px rgba( 0, 0, 0, 0.1 );

clear: both;

}


.inputss:focus {

    background: #fff;

    box-shadow: 0 0 0 3px #fff38e, inset 0 2px 3px rgba( 0, 0, 0, 0.2 ), 0px 5px 5px
    rgba( 0, 0, 0, 0.15 );

    outline: none;

}


.button {

    background-color: #0096da; /* Green */

    border: none;

    color: white;

    padding: 10px 23px;

    text-align: center;
```



```
text-decoration: none;

display: inline-block;

font-size: 16px;

}

</style>

<title>Registration</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/droid_sans_400-
droid_sans_700.font.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

</head>

<%
    if (request.getParameter("msg") != null) {
%>

<script>alert('Registration Succesfully');</script>

<%
    }
%>
```

```
<body>

<div class="main">

  <div class="header">

    <div class="header_resize">

      <div class="logo">

        <h1>Secure Data Sharing in Cloud Computing Using Revocable-Storage
Identity-Based Encryption</h1>

      </div>

      <div class="searchform">

        <form id="formsearch" name="formsearch" method="post" action="#">

          <span>

            <input name="editbox_search" class="editbox_search" id="editbox_search"
maxlength="80" value="Search our ste:" type="text" />

          </span>

          <input name="button_search" src="images/search.gif" class="button_search"
type="image" />

        </form>

      </div>

    <div class="clr"></div>

    <div class="menu_nav">

      <ul>

        <li><a href="index.jsp"><span>Home Page</span></a></li>

        <li><a href="pro.jsp"><span>Data Provider</span></a></li>

        <li><a href="user.jsp"><span>User</span></a></li>
```

```

        <li><a href="auditor.jsp"><span>Auditor</span></a></li>

        <li class="active"><a href="reg.jsp"><span>Registration</span></a></li>

    </ul>

</div>

<div class="clr"></div>

<div class="slider">

    <div id="coin-slider"> <a href="#"> </a> <a href="#"> </a> <a href="#"> </a> </div>

    <div class="clr"></div>

</div>

<div class="clr"></div>

</div>

<div class="content">

    <div class="content_resize">

        <div class="mainbar">

            <div class="article">

                <h2>Registration</h2><br>

                <div class="clr"></div>

                <!--star Body -->

                <form action="login.jsp" method="get" >

```

[illegible]

```
<!--End Body -->

<br></div>

</div>

<div class="sidebar">

  <div class="gadget">

    <h2 class="star"><span>Sidebar</span> Menu</h2>

    <div class="clr"></div>

    <ul class="sb_menu">

      <li><a href="index.jsp">Home</a></li>

      <li><a href="pro.jsp">Data Provider</a></li>

      <li><a href="user.jsp">User</a></li>

      <li><a href="auditor.jsp">Auditor</a></li>

      <li><a href="reg.jsp">Registration</a></li>

    </ul>

  </div>

</div>

<div class="clr"></div>

</div>

</div>

<div class="footer">

  <div class="footer_resize">

    <p class="lf">Copyright &copy; <a href="#">Jpinfotech</a></p>
```

<p class="rf">Design by Ajay</p>

<div style="clear:both;"></div>

</div>

</div>

</div>

</body>

</html>

4.3 RESULTS

4.3.1 SCREENS:



Figure 4.6: Home Page.

Description:

This is the home page of our project.

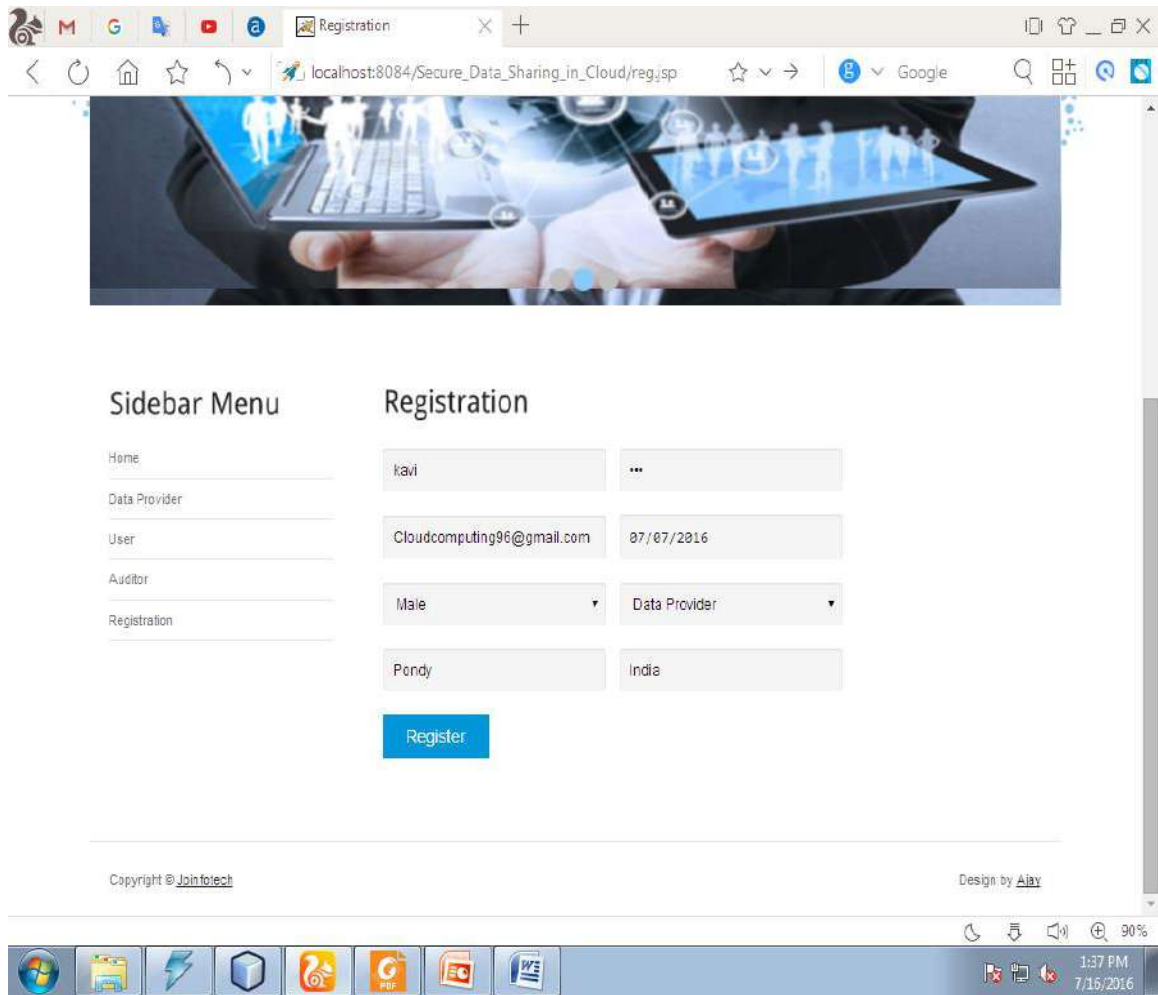


Figure 4.7 :Registration Page.

Description:

This is the page where user and data provider register.

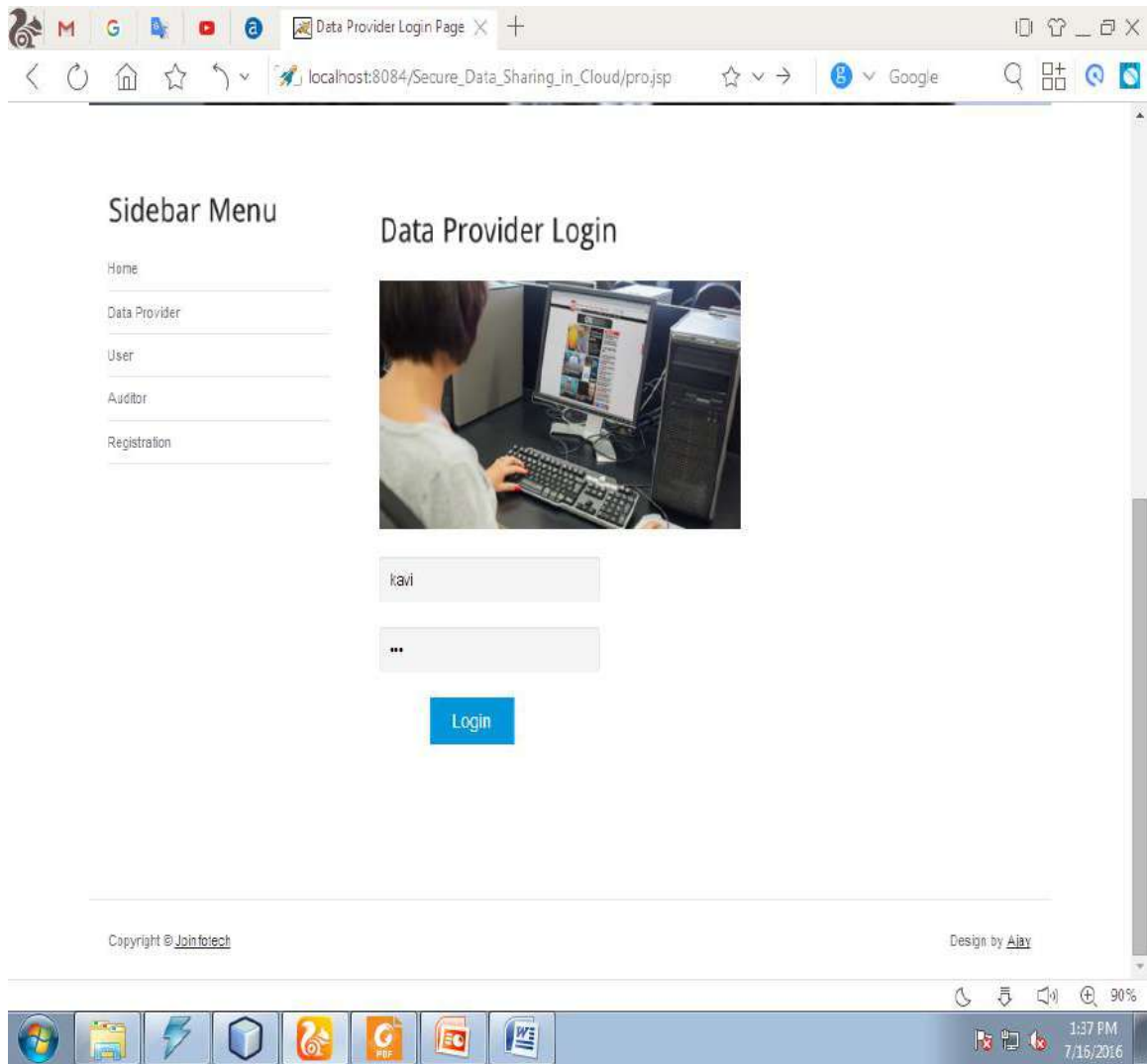


Figure 4.8: Data Provider Login Page.

Description:

This is the page where data provider login.

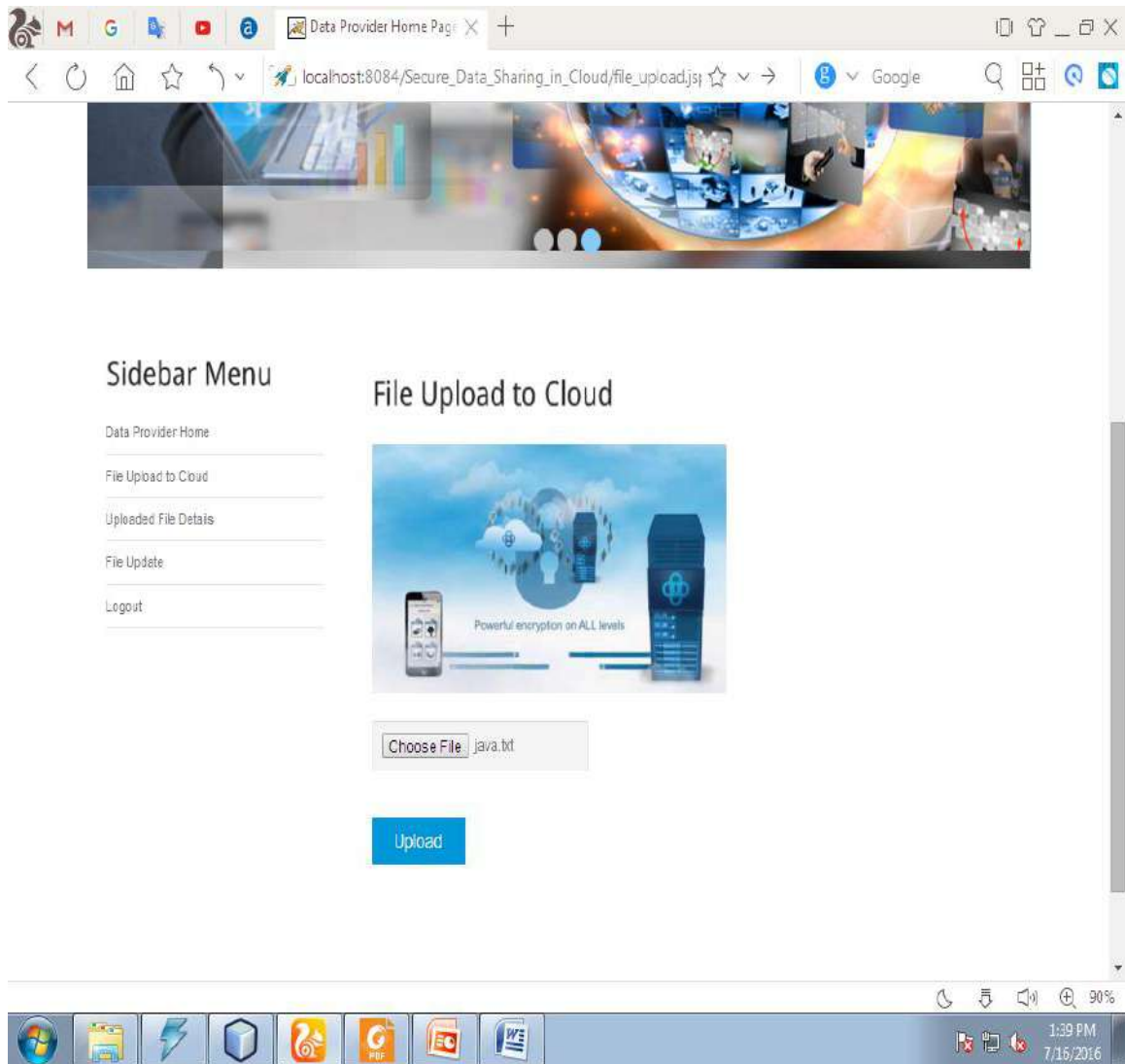


Figure 4.9: Uploading Page.

Description:

This is the page where data provider upload the files.

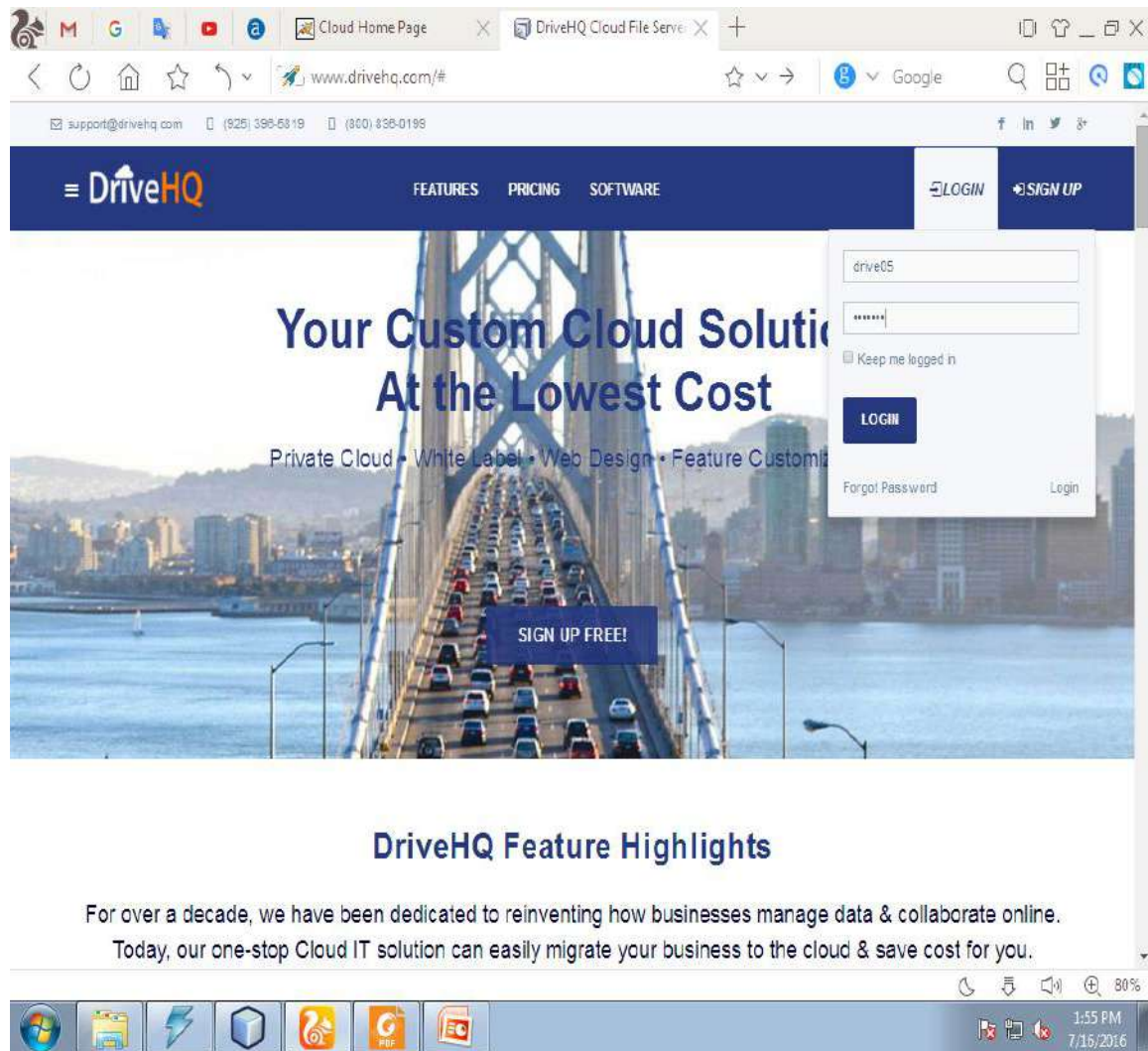


Figure 4.10: Cloud DriveHQ page.

Description:

This is the page where we store our cloud. The name of our cloud is DriveHQ

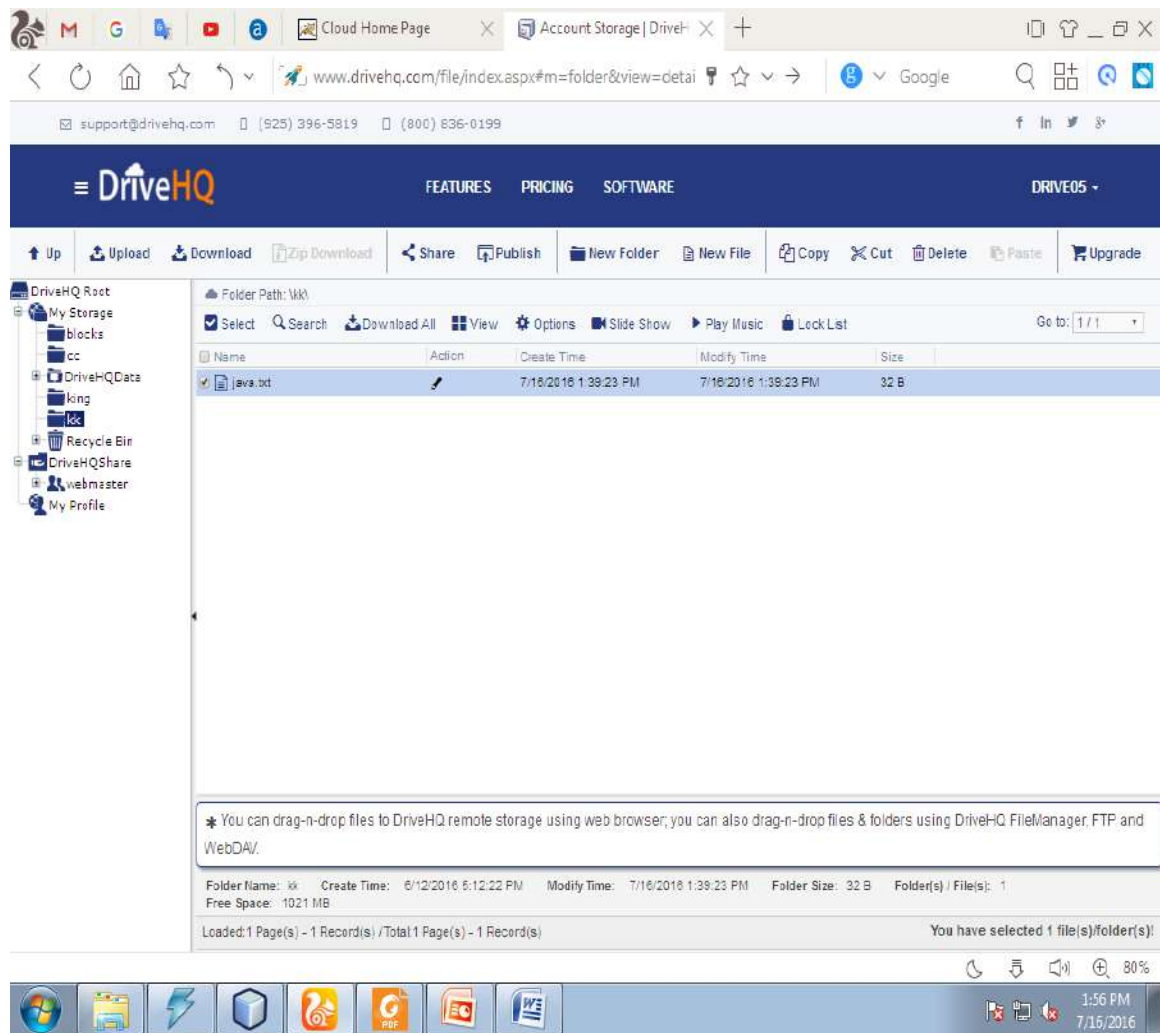


Figure 4.11: Folder page in DriveHq.

Description:

This is the page where our files are stored in DriveHQ cloud.



Figure 4.12: Viewing the uploaded file details page.

Description:

This is the page where we can see details of uploaded files.

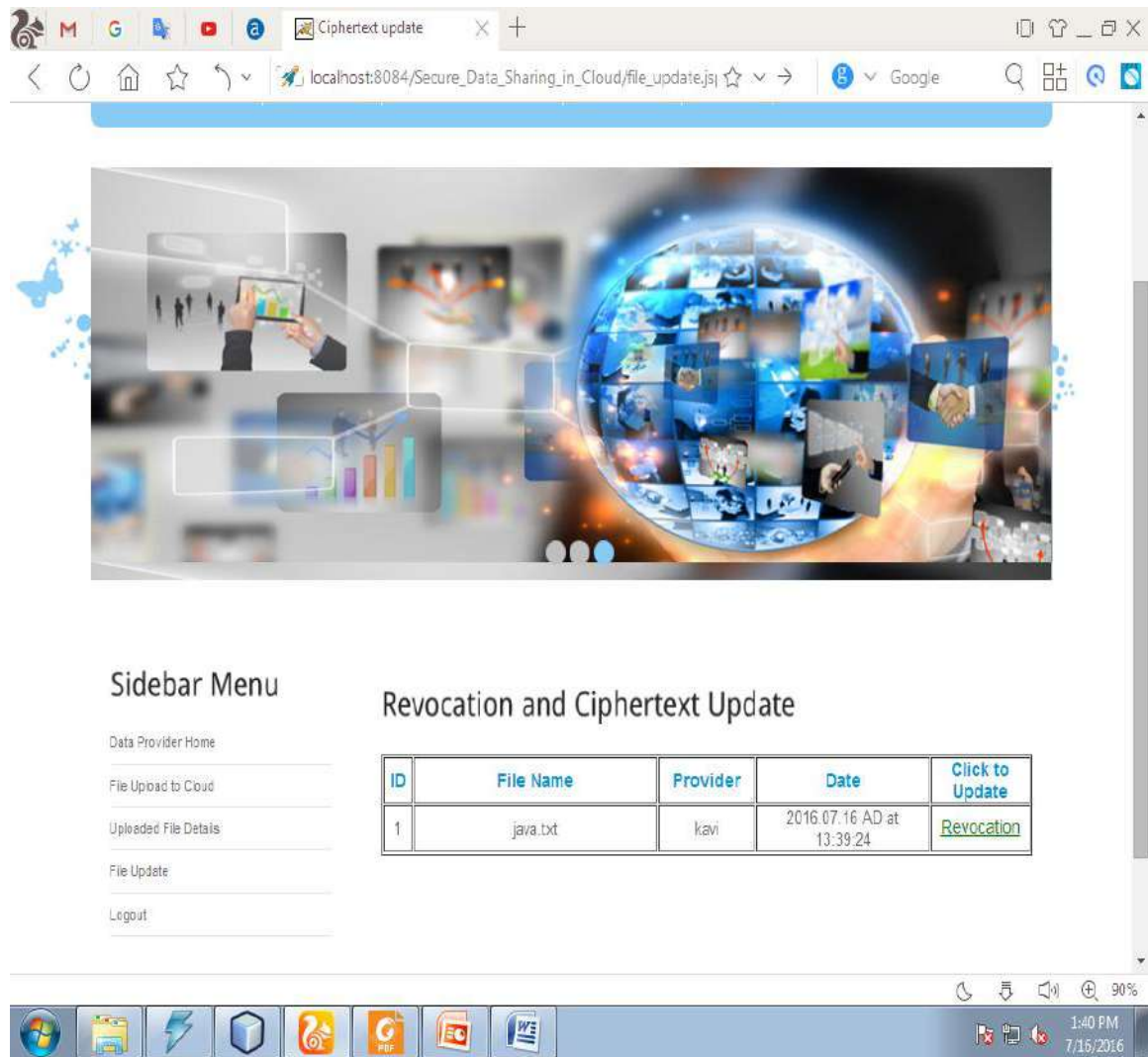


Figure 4.13: Updating the file page.

Description:

This is the page where data provider updates the page.

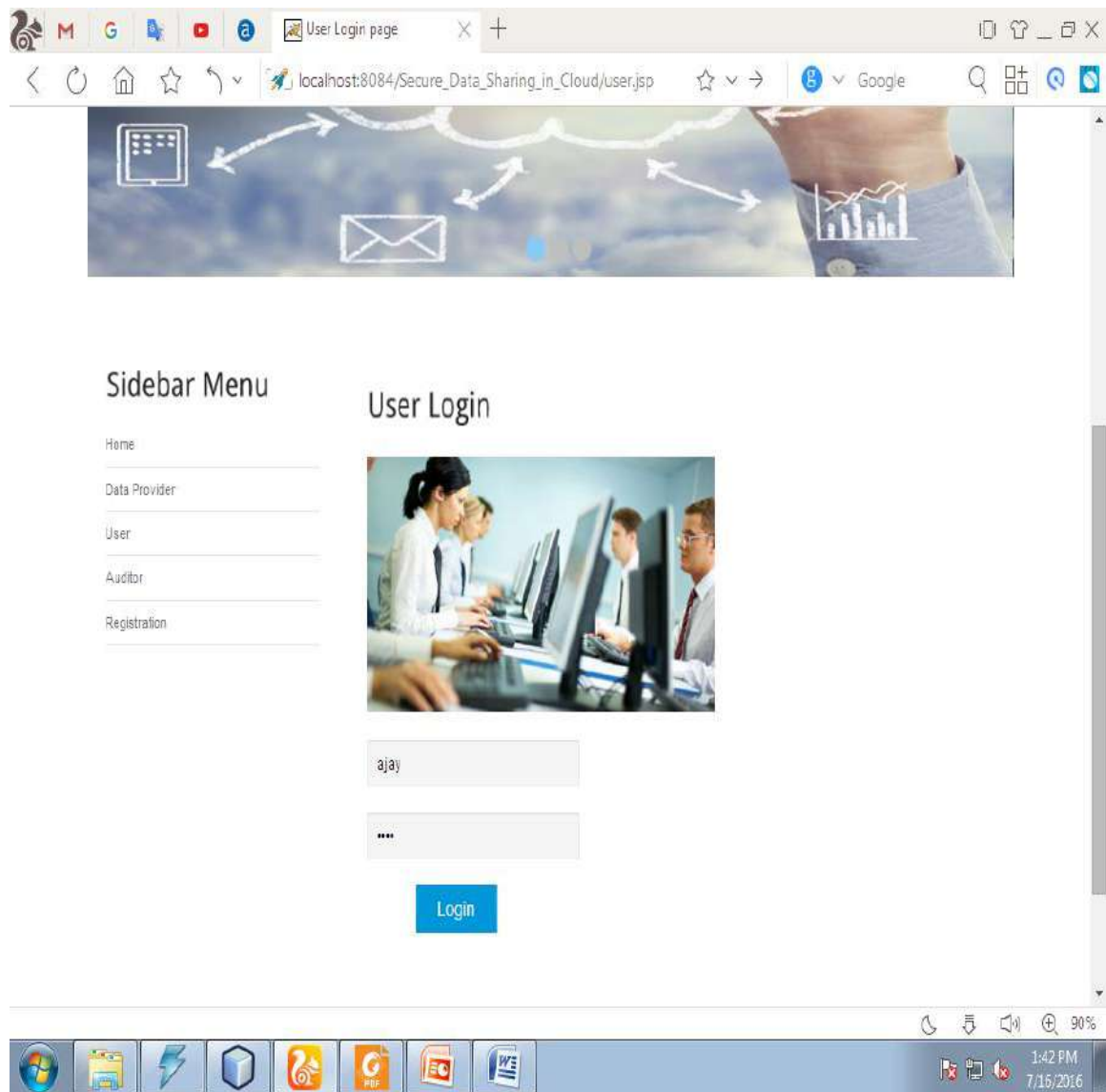


Figure 4.14: User Login Page.

Description:

This is the page where user login.

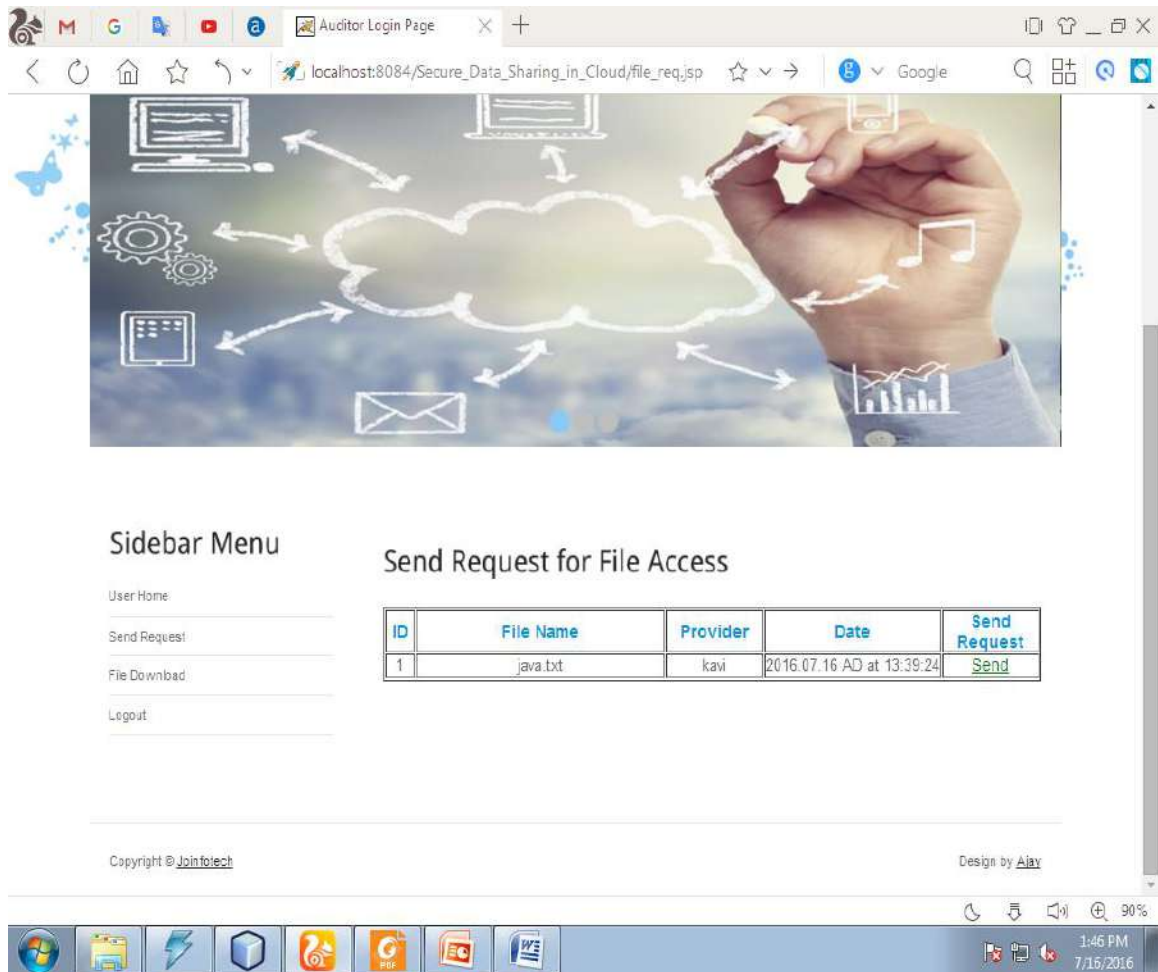


Figure 4.15: User sending request page.

Description:

This is the page where user sends request.

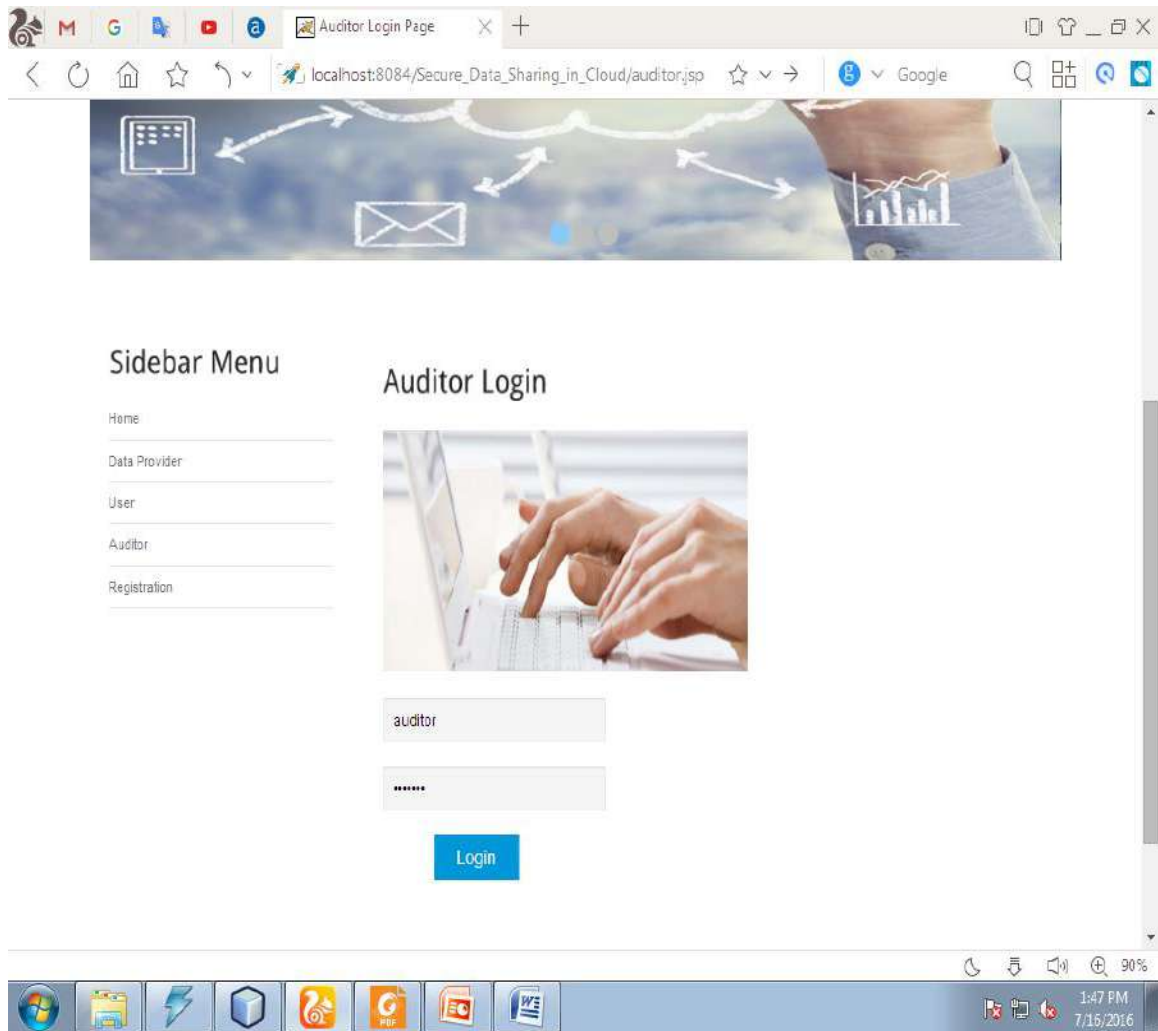


Figure 4.16: Auditor Login Page.

Description:

This is the page where auditor login.

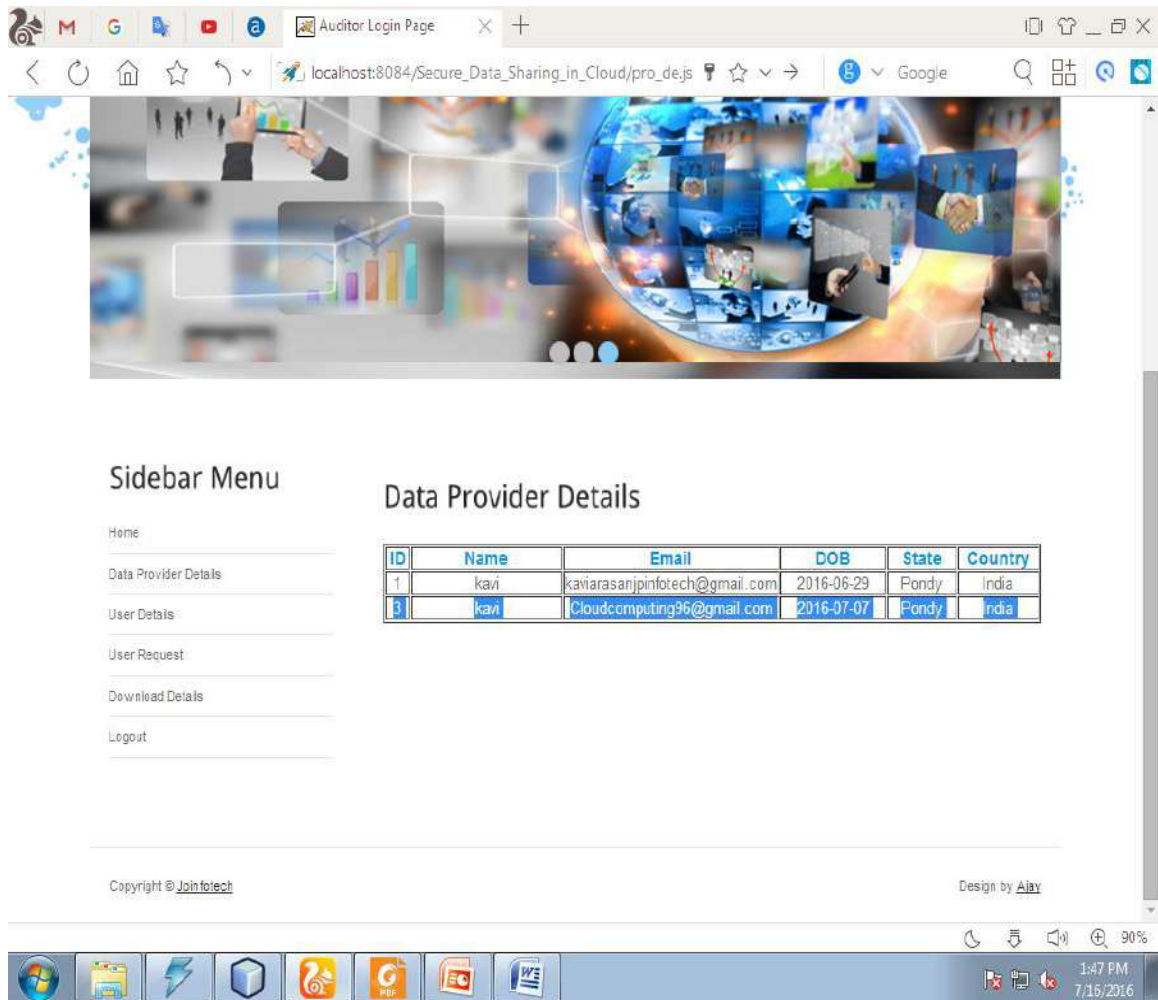


Figure 4.17: Viewing the data provider details page.

Description:

This is the page where auditor uploads details of data provider.

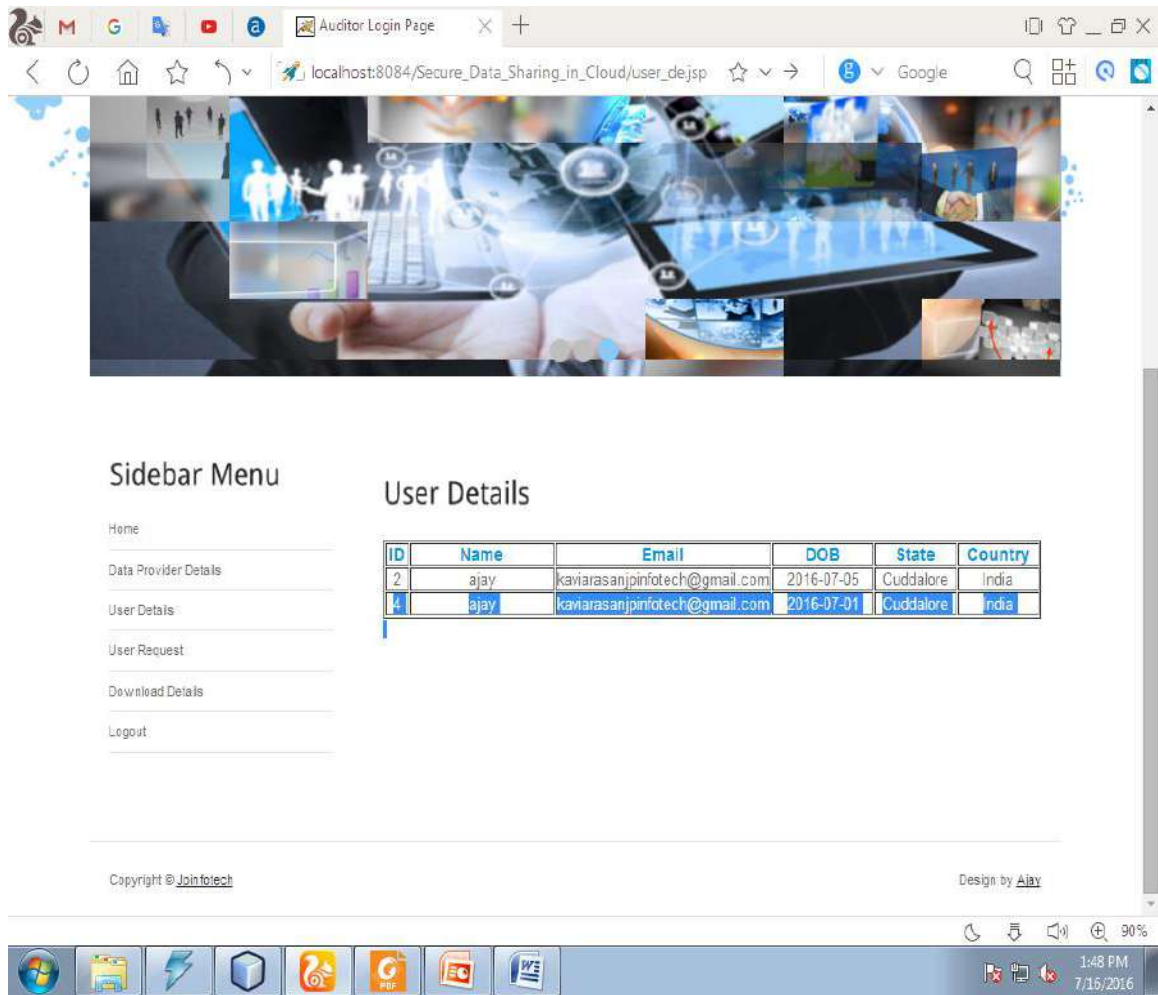


Figure 4.18: Viewing the user details page.

Description:

This is the page where auditor uploads details of user.

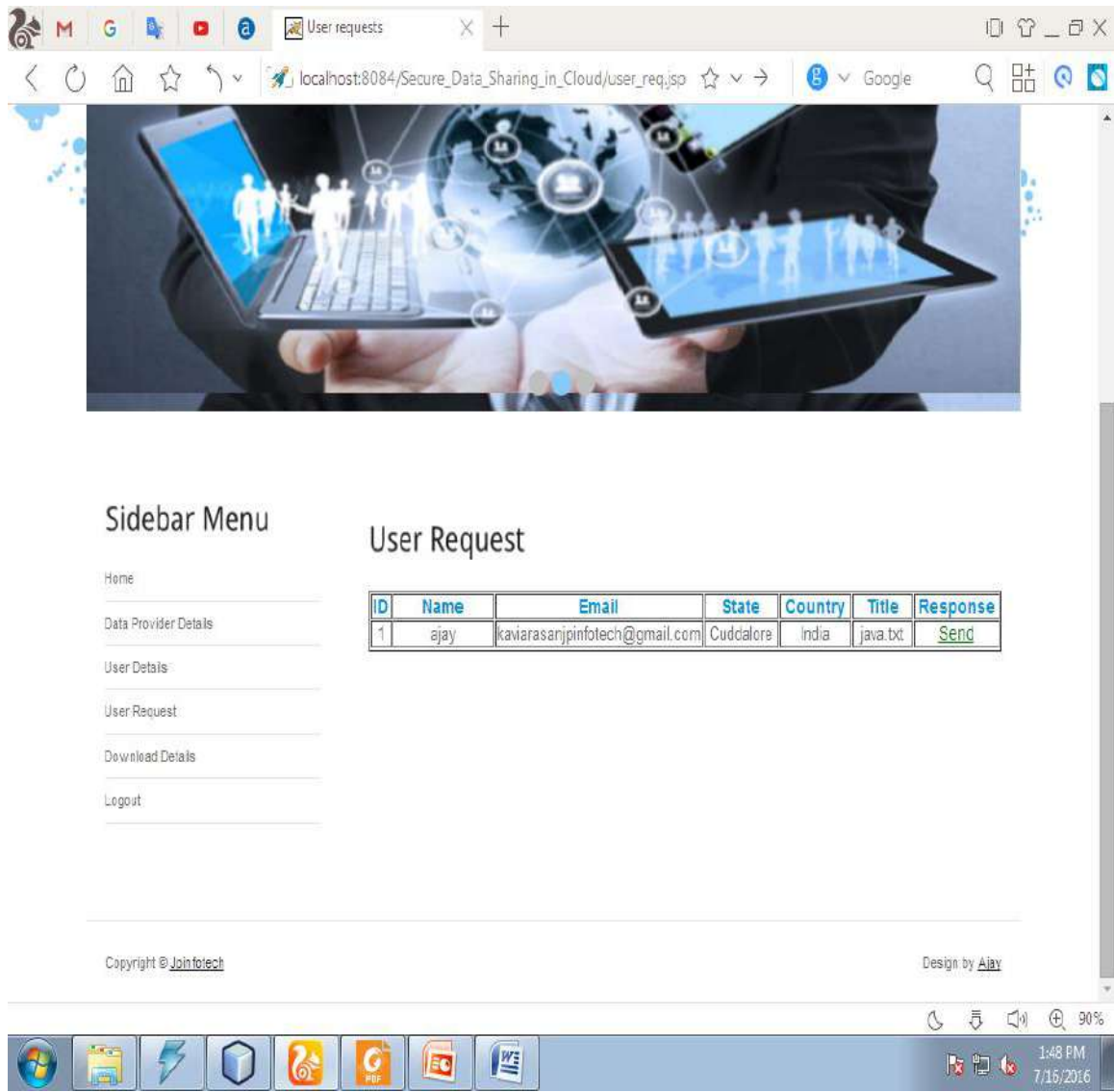


Figure 4.19: Viewing the user requests page.

Description:

This is the page where we can see the requests of the user.

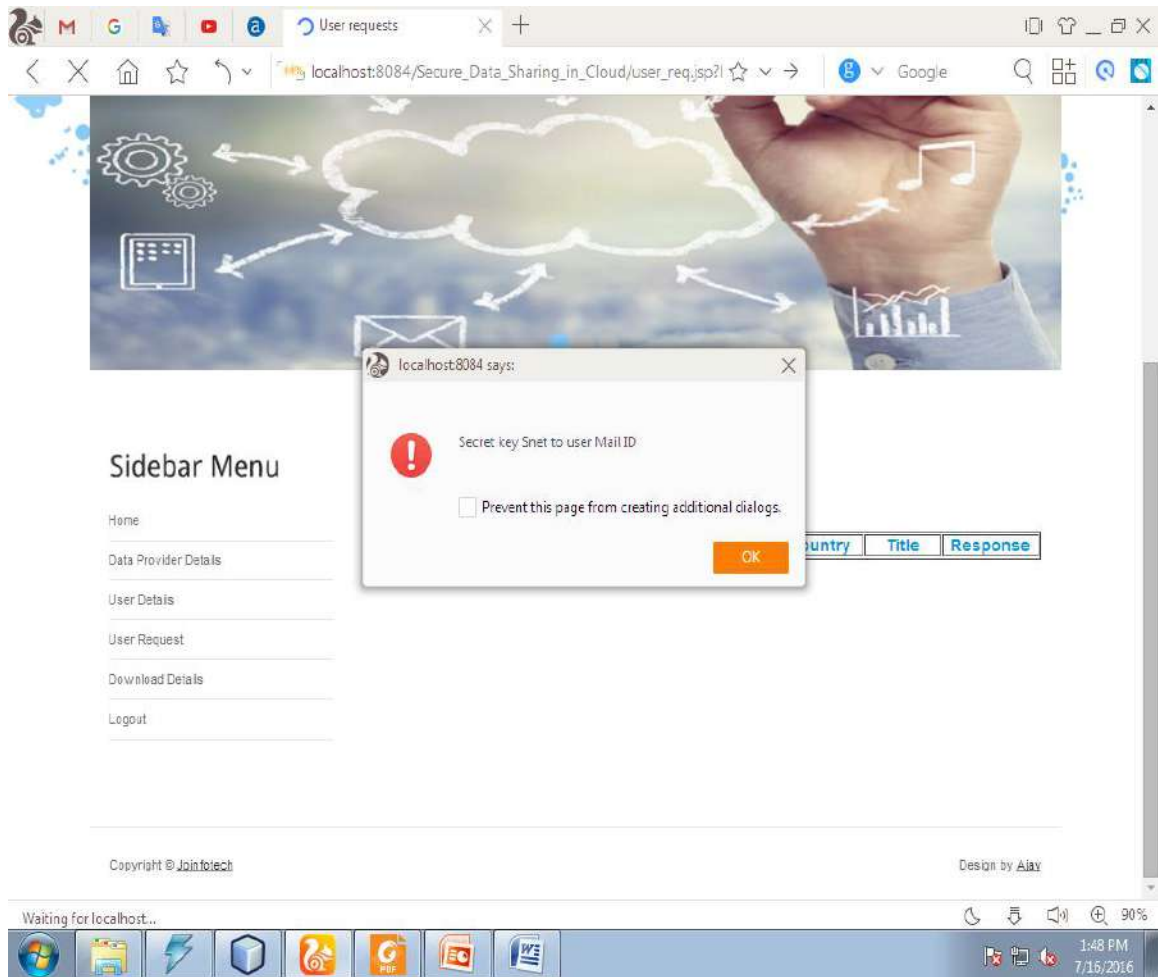


Figure 4.20: Sending secret key page.

Description:

This is the page where auditor sends the secret key to the respected mail.

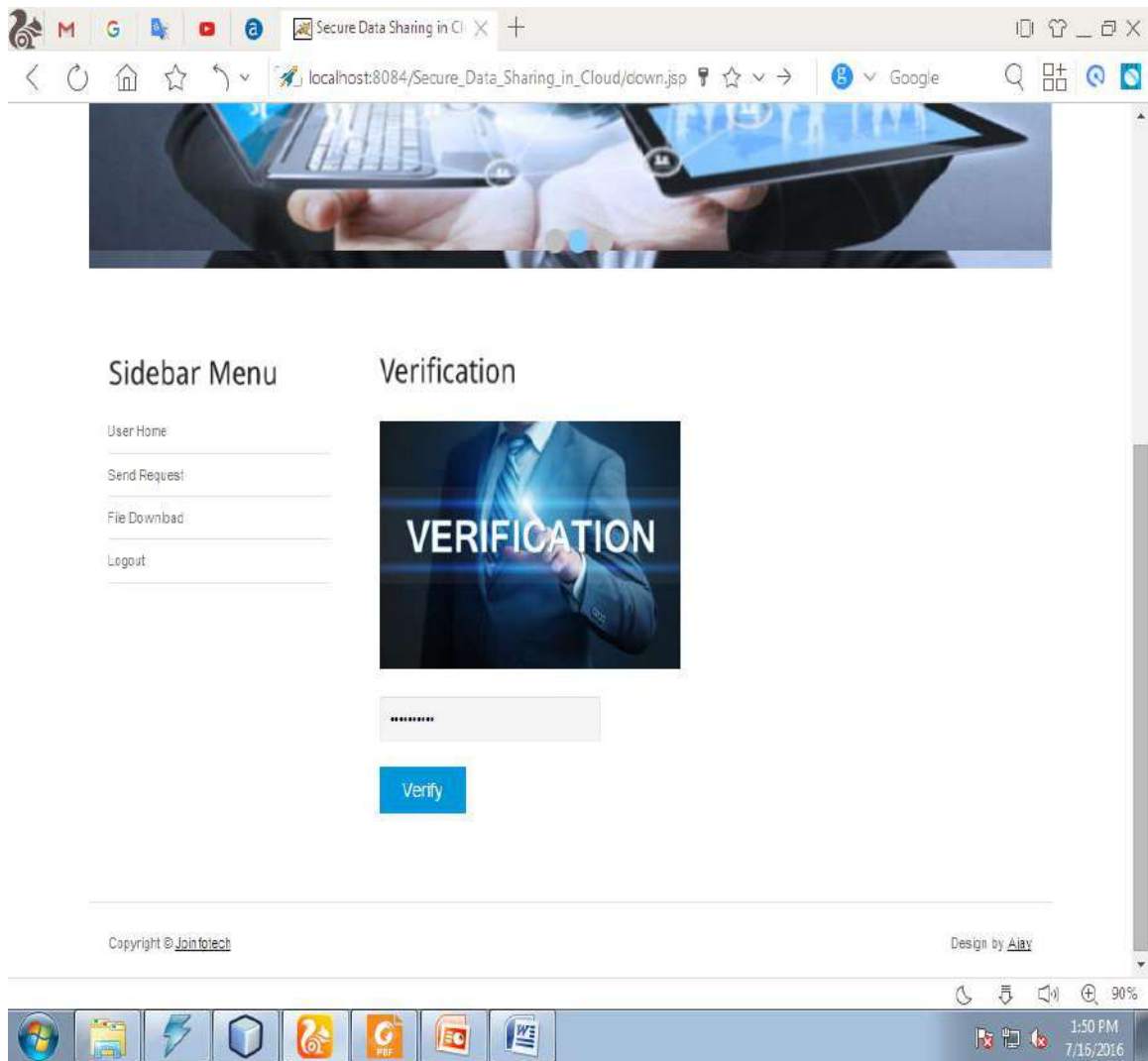


Figure 4.21: Page where user enters secret key for downloading or viewing file.

Description:

This is the page where user can download or view the file by using the secret key that has been sent to his mail.

CHAPTER- 5

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.1 TYPES OF TESTS

5.1.1 Unit testing

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

5.1.2 Integration testing

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

specifically aimed at exposing the problems that arise from the combination of components.

5.1.3 Functional test

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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

5.1.4 System Test

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

5.1.5 White Box Testing

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

5.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

5.1.7 Acceptance Testing

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

Test Case Name	Test Case Description	Test Steps			Test Case Status
		Step	Expected	Actual	
Upload file	User has to upload file	If we uploaded the file	Encrypted file uploaded successfully	Encrypted file uploaded successfully	Pass
Update file	User has to update file	If we updated the file	File is updated successfully	File is updated successfully	Pass
View file	User can view the file	Files are uploaded	User can view the file	User can view the file	Pass
Secret key	We get secret from auditor	Secret key should be entered	Secret key has been sent to email id	Secret key has been sent to email id	Pass

Verification of secret key	We has to enter secret key	Secret key has been sent to email id	Verified successfully	Verified successfully	Pass
Download file	If user want to download	User download the by entering secret key	Downloaded Successfully	Downloaded Successfully	Pass

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER - 6

CONCLUSION

Cloud computing brings great convenience for people. Particularly, it perfectly matches the increased need of sharing data over the Internet. In this paper, to build a cost-effective and secure data sharing system in cloud computing, we proposed a notion called RS-IBE, which supports identity revocation and ciphertext update simultaneously such that a revoked user is prevented from accessing previously shared data, as well as subsequently shared data. Furthermore, a concrete construction of RS-IBE is presented. The proposed RS-IBE scheme is proved adaptive-secure in the standard model, under the decisional ℓ -DBHE assumption. The comparison results demonstrate that our scheme has advantages in terms of efficiency and functionality, and thus is more feasible for practical applications.

CHAPTER - 7

REFERENCES

- [1] L. M. Vaquero, L. Roderio-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50–55, 2008.
- [2] iCloud. (2014) Apple storage service. [Online]. Available: <https://www.icloud.com/>
- [3] Azure. (2014) Azure storage service. [Online]. Available: <http://www.windowsazure.com/>
- [4] Amazon. (2014) Amazon simple storage service (amazon s3).[Online]. Available: <http://aws.amazon.com/s3/>
- [5] K. Chard, K. Bubendorfer, S. Caton, and O. F. Rana, “Social cloud computing: A vision for socially motivated resource sharing,” Services Computing, IEEE Transactions on, vol. 5, no. 4, pp. 551–563, 2012.
- [6] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy preserving public auditing for secure cloud storage,” Computers, IEEE Transactions on, vol. 62, no. 2, pp. 362–375, 2013.
- [7] G. Anthes, “Security in the cloud,” Communications of the ACM, vol. 53, no. 11, pp. 16–18, 2010.
- [8] K. Yang and X. Jia, “An efficient and secure dynamic auditing protocol for data storage in cloud computing,” Parallel and Distributed Systems, IEEE Transactions on, vol. 24, no. 9, pp. 1717–1726, 2013.

[9] B. Wang, B. Li, and H. Li, “Public auditing for shared data with efficient user revocation in the cloud,” in INFOCOM, 2013 Proceedings IEEE. IEEE, 2013, pp. 2904–2912.

[10] S. Ruj, M. Stojmenovic, and A. Nayak, “Decentralized access control with anonymous authentication of data stored in clouds,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 2, pp. 384–394, 2014.