

A
Mini Project Report On
**“CHARON A SECURE CLOUD OF CLOUDS SYSTEM FOR STORING
AND SHARING BIG DATA”**

Submitted to JNTUH in partial fulfillment of the
Requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In
COMPUTER SCIENCE & ENGINEERING

By

MOHAMMAD ASHFAQ

20N61A0583

Under the Guidance of
Dr. Dabbu Murali
Professor & Principal



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIVEKANANDA INSTITUTE OF TECHNOLOGY AND SCIENCE

(Approved by AICTE New Delhi & Affiliated to JNTU, Hyderabad)

An ISO 9001:2015 certified Institute)

KARIMNAGAR-505501

2023-2024

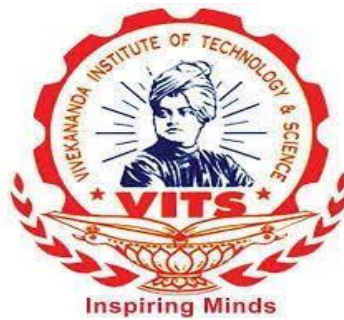
VIVEKANANDA INSTITUTE OF TECHNOLOGY & SCIENCE(N6)

(Approved by AICTE New Delhi & Affiliated to JNTU, Hyderabad)

An ISO 9001:2015 Certified Institution

KARIMNAGAR-505001

CERTIFICATE



This is to certify that the mini-project report titled “**CHARON A SECURE CLOUD OF CLOUDS SYSTEM FOR STORING AND SHARING BIG DATA**” is being submitted by **MOHAMMAD ASHFAQ 20N61A0583** in B. Tech IV-I semester, Computer Science & Engineering is a record bonafide work carried out by him. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide

Dr. Dabbu Murali

Professor & Principal

Head of the Department

Dr. M. V. Hanumanthareddy

Dept of CSE

External Examiner

Principal

Dr. Dabbu Murali

Professor

VIVEKANANDA INSTITUTE OF TECHNOLOGY & SCIENCE

(Approved by AICTE New Delhi & Affiliated to JNTU, Hyderabad)

An ISO 9001:2015 Certified Institution

KARIMNAGAR-505501

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION



I **MOHAMMAD ASHFAQ 20N61A0583** by declare that the Project report entitled **“CHARON A SECURE CLOUD OF CLOUDS SYSTEM FOR STORING AND SHARING BIG DATA”** submitted in partial fulfillment of the requirements for the award of degree in B. Tech IV-I semester, Computer Science & Engineering This is a record bonafide work carried out me. The results embodied in this report have not been submitted to any other University for the award of any degree or diploma.

MOHAMMAD ASHFAQ

20N61A0583

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIVEKANANDA INSTITUTE OF TECHNOLOGY & SCIENCE, Karimnagar.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide, **Dr. Dabbu Murali Professor & Principal** whose knowledge and guidance has motivated us to achieve goals we never thought possible. She has consistently been a source of motivation, encouragement, and inspiration. The time we have spent working under her supervision has truly been a pleasure.

We thank our H.O.D **Dr. M. V. Hanumanthareddy** for his effort and indefatigable guidance rendered throughout the progress of project work. Thanks to programmers and nonteaching staff of CSE Department of VITS(N6).

We thank our Principal **Dr. Dabbu Murali** and Management for providing excellent facilities to carry out project work.

Finally, Special thanks to our parents for their support and encouragement throughout this course. And thanks to our friends and well wishers for their constant support.

MOHAMMAD ASHFAQ

20N61A0583

ABSTRACT

We present CHARON, a cloud-backed storage system capable of storing and sharing big data in a secure, reliable, and efficient way using multiple cloud providers and storage repositories to comply with the legal requirements of sensitive personal data. CHARON implements three distinguishing features: (1) it does not require trust on any single entity, (2) it does not require any client-managed server, and (3) it efficiently deals with large files over a set of geo-dispersed storage services. Besides that, we developed a novel Byzantine-resilient data-centric leasing protocol to avoid write-write conflicts between clients accessing shared repositories. We evaluate CHARON using micro and application-based benchmarks simulating representative workflows from bioinformatics, a prominent big data domain. The results show that our unique design is not only feasible but also presents an end-to-end performance of up to 2.5x better than other cloud-backed solutions.

INDEX

CHAPTER 1	PAGE NO
INTRODUCTION	1
1.1 Introduction	1
1.2 Existing System	6
1.3 Disadvantages of Existing System	7
1.4 Proposed System	8
1.5 Advantages of Proposed System	9
CHAPTER 2	
PRELIMINARY INVESTIGATION	10
2.1 System Design	10
2.2 Literature Survey	12
CHAPTER 3	
SYSTEM REQUIREMENTS SPECIFICATIONS	14
3.1 Hardware Requirements	14
3.2 Software Requirements	14
3.3 Feasibility Study	15
CHAPTER 4	
DESIGN METHODOLOGY	16
4.1 Architecture Diagram	16
4.2 Dataflow Diagram	17
4.3 UML Diagrams	18
CHAPTER 5	
IMPLEMENTATION METHODOLOGY	23

CHAPTER 6	
SOFTWARE ENVIRONMENT	24
6.1 Java Technology	24
6.2 ODBC	30
6.3 JDBC	32
6.4 Network Concept	35
6.5 J2ME	38
6.6 Tomcat Server	41
CHAPTER 7	
TESTING STRATEGY	43
7.1 Testing	43
7.2 Types of Tests	44
7.3 Validation	45
CHAPTER 8	
OUTPUT SCREENS	47
CHAPTER 9	
CONCLUSION	56
CHAPTER 10	
REFERENCES	57

LIST OF FIGURES

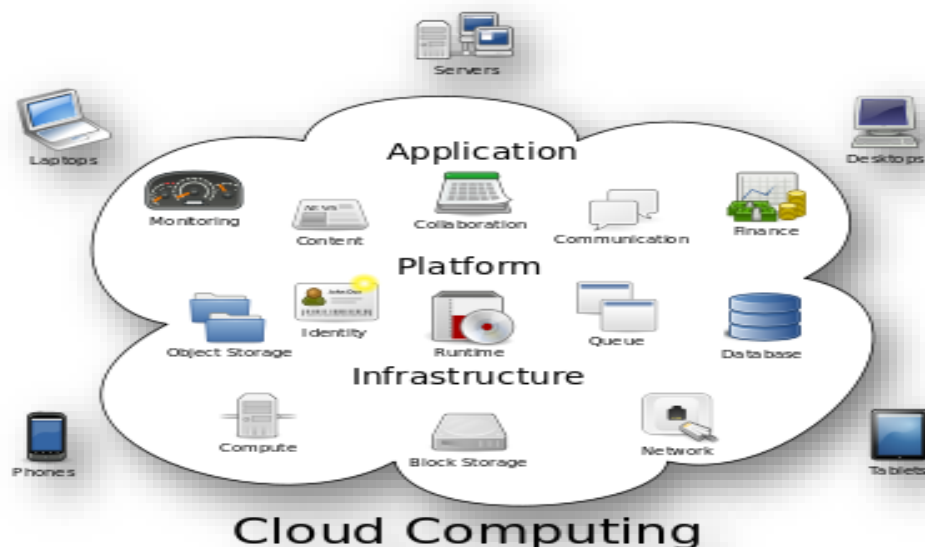
CONTENTS	PAGENO
Fig: 4.1 Architecture Diagram	16
Fig: 4.3.1 USE CASE Diagram	20
Fig: 4.3.2 Class Diagram	21
Fig: 4.3.3 Sequence Diagram	22
Fig: 6.1 Java Programming Structure	24
Fig: 6.1.2 Java Compiler CAN Run Any Where	25
Fig: 6.1.3 Java Platform Diagram	26
Fig: 6.1.4 Java SDK	28
Fig: 6.3.1 Java Overview	34
Fig: 6.5.1 J2ME Architecture	38
Fig: 6.6.1 Apache Tomcat 7.0	41
Fig: 8.1 Home Screen Login	47
Fig: 8.2 Cloud Server Login	48
Fig: 8.3 Threshold Details	49
Fig: 8.4 Workload Details	50
Fig: 8.5 Data Owner Registration	51
Fig: 8.6 Data Owner Login	52
Fig: 8.7 Upload Data	53

Fig: 8.8 End User Login	54
Fig: 8.7 Download File	55

1. INTRODUCTION

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.



Structure of cloud computing

How Cloud Computing Works?

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

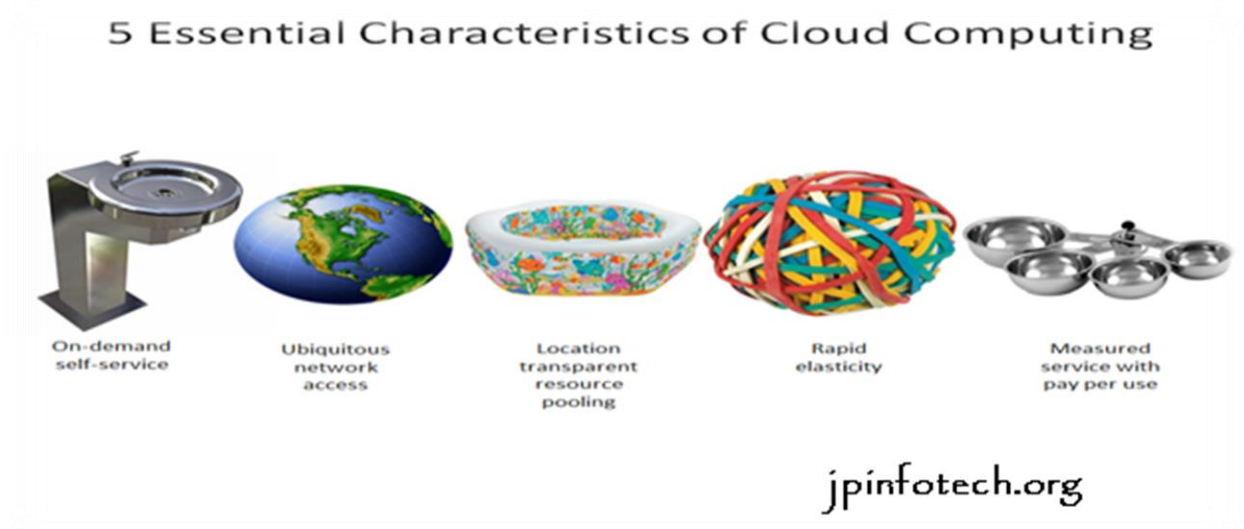
The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques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

1.2 Characteristics and Services Models:

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

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

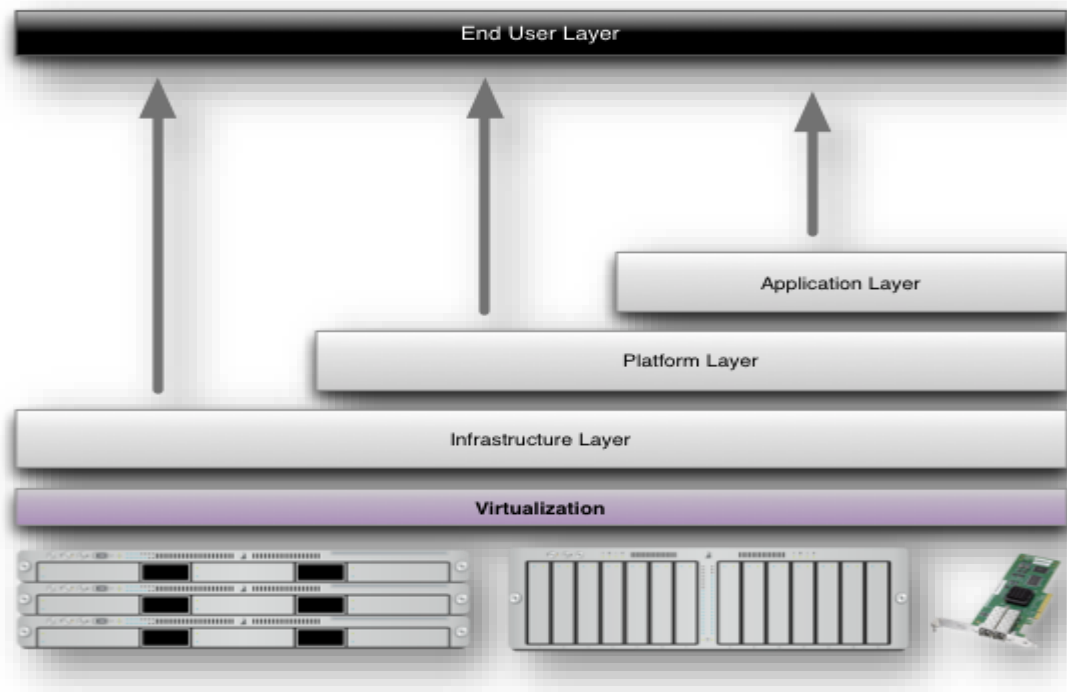
Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.



Characteristics of cloud computing

Services Models:

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



Structure of service models

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.

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 Existing System

- ❖ Byzantine disk Paxos [26] is a consensus protocol built on top of untrusted shared disks. More recently, an enhanced version of this protocol specifically designed to use file synchronization services (e.g., DropBox, Google Drive) instead of disks was published [21]. These algorithms could be used to implement mutual exclusion satisfying deadlock-freedom (a stronger liveness guarantee than obstruction-freedom). However, these solutions would require a much larger number of cloud accesses. Our lease protocol, on the other hand, requires only two to four cloud accesses for acquiring a lease.
- ❖ To the best of our knowledge, there are only two fault-tolerant data-centric lease algorithms in the literature [15], [39]. The lease algorithm of Chockler and Malkhi [39] has two important differences when compared with CHARON’s BFT composite lease. First, it does not provide an always-safe lease as it admits the existence of more than one process with valid leases. Second, it tolerates only crashes, requiring thus some trust on individual cloud providers. The BFT mutual exclusion algorithm from DepSky [15] is a natural candidate to regulate access contention in CHARON. However, our composite lease algorithm is 4×10^3 faster than DepSky’s (see x5.2), does not require clients to have synchronized clocks, and neither rely on weakly-consistent operations such as object storage’s list.
- ❖ Systems like Hybris [23], SCFS [24] and RockFS [70] employ a hybrid approach in which unmodified cloud storage services are used together with few computing nodes to store metadata and coordinate data access. The main limitation of these systems is that they require servers deployed in the cloud providers, which implies additional costs and management complexity. The same limitation applies to modern (single-provider) geo-replicated storage systems such as Spanner [71], SPANStore [25] and Pileus [72], if deployed in multiple clouds.

A slightly different kind of work proposes the aggregation of multiple file synchronization services (e.g., DropBox, Box, Google Drive) in a single dependable service [20], [21], [22]. CYRUS [20] does not implement any kind of concurrency control, allowing different clients to create different versions of files accessed concurrently.

1.3 Disadvantages of Existing system

- ❖ In the existing work, the system is less effective while dealing with shared files.
- ❖ The system has less security while working with data chunks.□

1.4 Proposed System

- ❖ The system proposes a CHARON which is a distributed file system that provides a near-POSIX interface to access an ecosystem of multiple cloud services and allows data transfer between clients. The preference for a POSIX interface rather than using data objects resorts to the fact the envisioned users are likely to be non-experts, and existent life sciences tools use files as their input most of the times. In particular, the system needs to (1) efficiently deal with multiple storage locations, (2) support reasonably big files, and (3) offer controlled file sharing. These challenges are amplified by our goals of excluding user-deployed servers and of requiring no modifications to existing cloud services (for immediate deployability).
- ❖ All techniques used in CHARON were combined considering two important design decisions. First, the system absorbs file writes in the client's local disk and, in the background, uploads them to their storage location. Similarly, prefetching and parallel downloads are widely employed for accelerating reads. This improves the usability of CHARON since transferring large files to/from the clouds take significant time (see x5). Second, the system avoids write-write conflicts, ruling out any optimistic mechanism that relies on users/applications for conflict resolution.
- ❖ The expected size of the files and the envisioned users justify this decision. More specifically, (1) solving conflicts manually in big files can be hard and time-consuming; (2) users are likely to be non-experts, normally unaware of how to repair such conflicts; and (3) the cost of maintaining duplicate copies of big files can be significant. For instance, collaborative repositories, such as the Google Genomics [31], require such control since they allow users to read data about available samples, process them, and aggregate novel knowledge on them by sharing the resulting derived data into the bucket containing the sample of interest.

1.5 Advantages of Proposed System

- ❖ Mutual Exclusion (safety): There are never two correct clients with a valid lease for the same resource.
- ❖ Obstruction-freedom (liveness): A correct client that tries to lease a resource without contention will succeed.
- ❖ Time-boundedness (liveness): A correct client that acquires a lease will hold it for at most T time units unless the lease was renewed.

2.PRELIMINARY INVESTIGATION

2.1 System Design

Input design

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

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

Objectives

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 follows:

Output design

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

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

The output form of an information system should accomplish one or more of the following objectives.

- ❖ Convey information about past activities, current status or projections of the Future.
- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

2.2 Literature Survey

Introduction

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating System and Language can be used for developing the tool

1) BYZANTINE DISK PAXOS: OPTIMAL RESILIENCE WITH BYZANTINE SHARED MEMORY

We present Byzantine Disk Paxos, an asynchronous shared-memory consensus algorithm that uses a collection of $n > 3t$ disks, t of which may fail by becoming non responsive or arbitrarily corrupted. We give two constructions of this algorithm; that is, we construct two different t -tolerant (i.e., tolerating up to t disk failures) building blocks, each of which can be used, along with a leader oracle, to solve consensus. One building block is a t -tolerant wait-free shared safe register. The second building block is a t -tolerant regular register that satisfies a weaker termination (liveness) condition than wait freedom: its write operations are wait-free, whereas its read operations are guaranteed to return only in executions with a finite number of writes. We call this termination condition finite writes (FW), and show that wait-free consensus is solvable with FW terminating registers and a leader oracle. We construct each of these tolerant registers from $n > 3t$ base registers, t of which can be non-responsive or Byzantine. All the previous t -tolerant wait-free constructions in this model used at least $4t + 1$ fault-prone registers, and we are not familiar with any prior FW terminating constructions in this model.

2) UNIDRIVE: SYNERGIZE MULTIPLE CONSUMER CLOUD STORAGE SERVICES

Consumer cloud storage (CCS) services have become popular among users for storing and synchronizing files via apps installed on their devices. A single CCS, however, has intrinsic limitations on networking performance, service reliability, and data security. To overcome these limitations, we present UniDrive, a CCS app that synergizes multiple CCSs (multi-cloud) by using only few simple public RESTful Web APIs. UniDrive follows a server-less, client-centric design, in which synchronization logic is purely implemented at client devices and all communication is conveyed through file upload and download operations. Strong consistency of the metadata is guaranteed via a quorum-based distributed mutual-exclusive lock mechanism.

UniDrive improves reliability and security by judiciously distributing erasure coded files across multiple CCSs. To boost networking performance, UniDrive leverages all available clouds to maximize parallel transfer opportunities, but the key insight behind is the concept of data block over-provisioning and dynamic scheduling. This suite of techniques masks the diversified and varying network conditions of the underlying clouds, and exploits more the faster clouds via a simple yet effective in-channel probing scheme. Extensive experimental results on the global Amazon EC2 platform and a real-world trial by 272 users confirmed significantly superior and consistent sync performance of UniDrive over any single CCS.

3. SYSTEM REQUIREMENT SPECIFICATION

3.1 Hardware Requirements

- System : core i3
- Hard Disk : 500GB
- RAM : 2GB

3.2 Software Requirements

- Operating System : Windows 7
- Front End : HTML, Java, JSP
- Scripts : JavaScript
- Database : MySQL
- Database connectivity : JDBC

3.3 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

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.

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.

4. DESIGN METHODOLOGY

4.1 Architecture Diagram

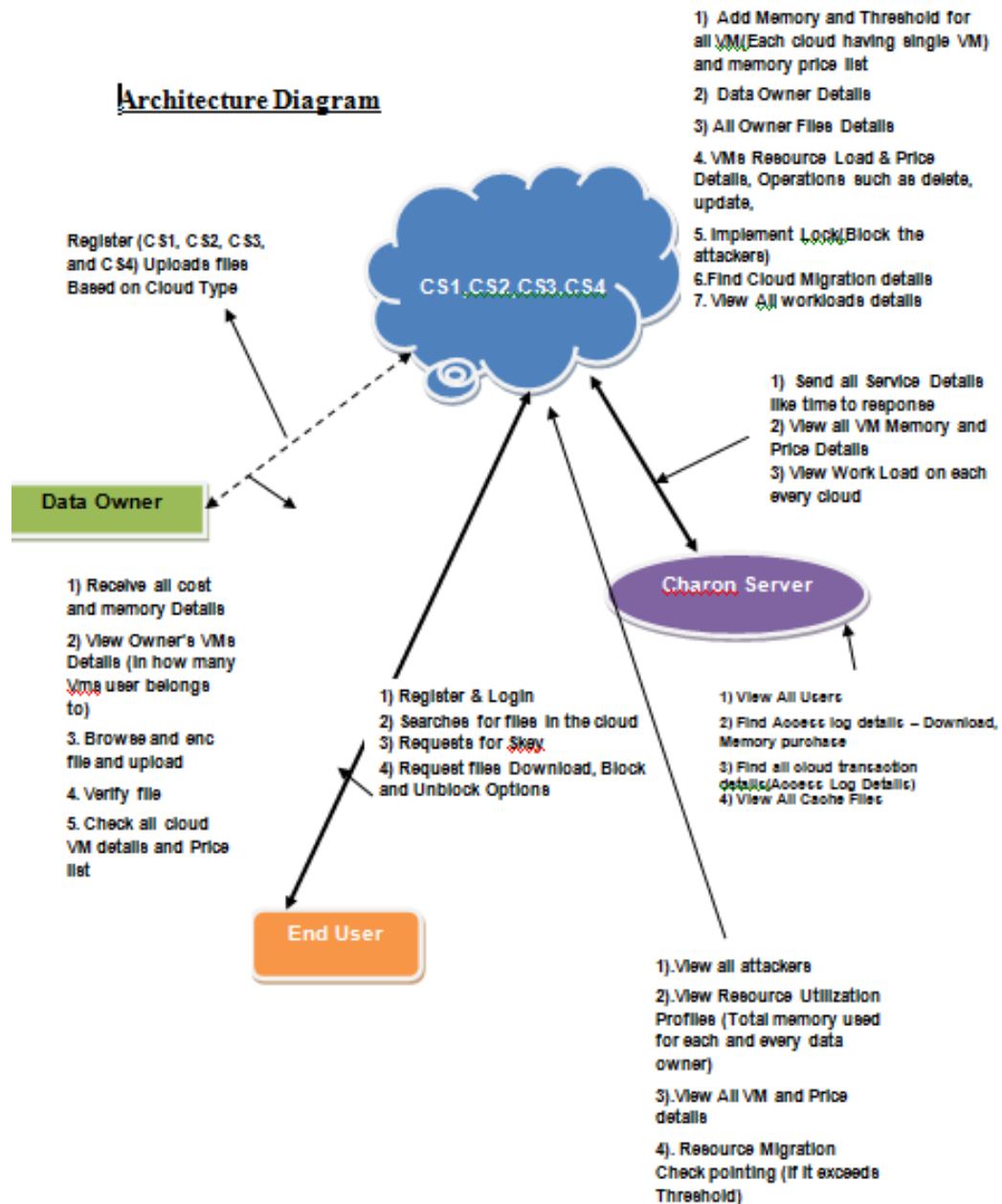


Fig: 4.1 Architecture Diagram

4.2 Dataflow Diagram

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

4.3 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.

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.

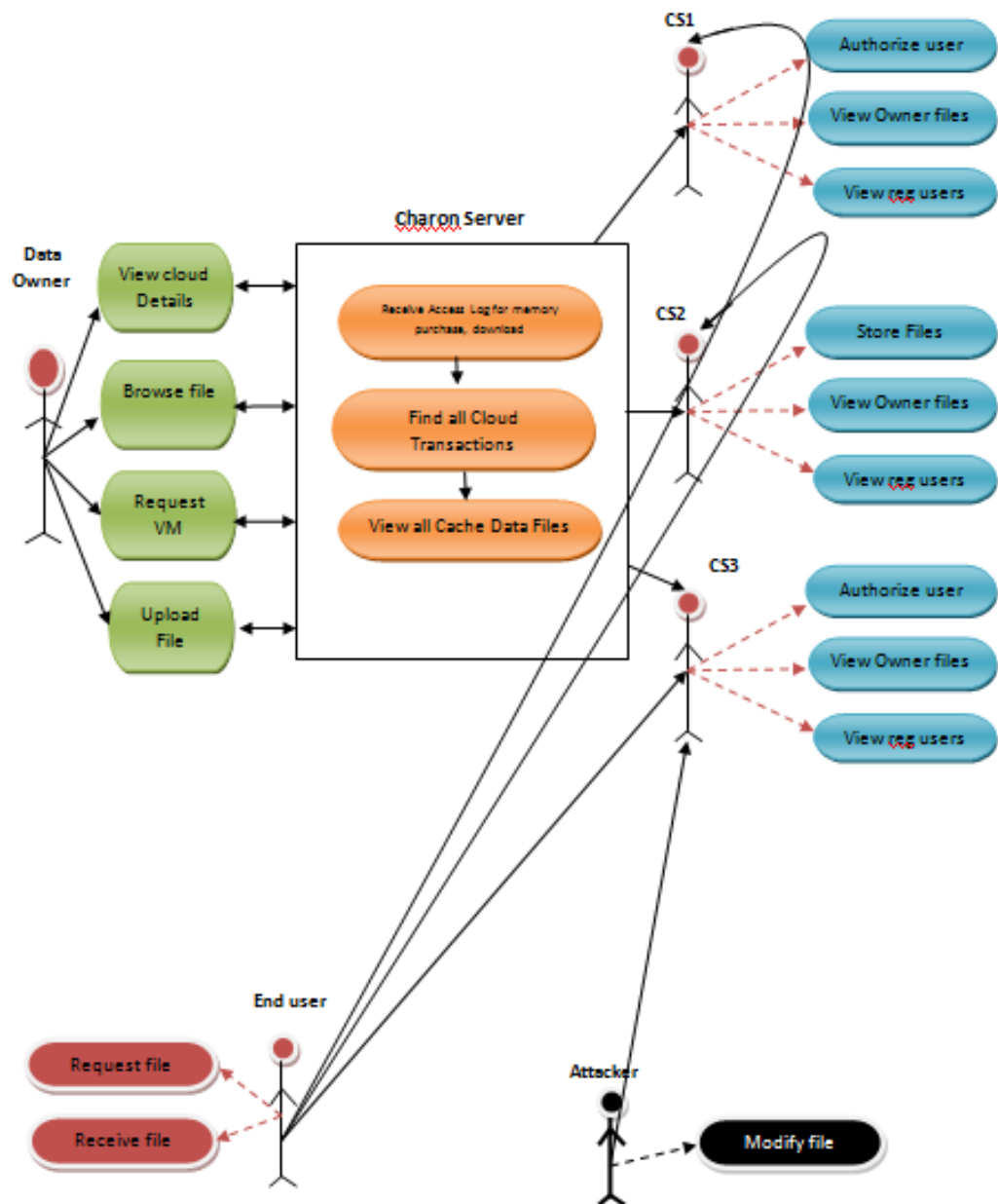


Fig: 4.3.1 USE CASE DIAGRAM

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.

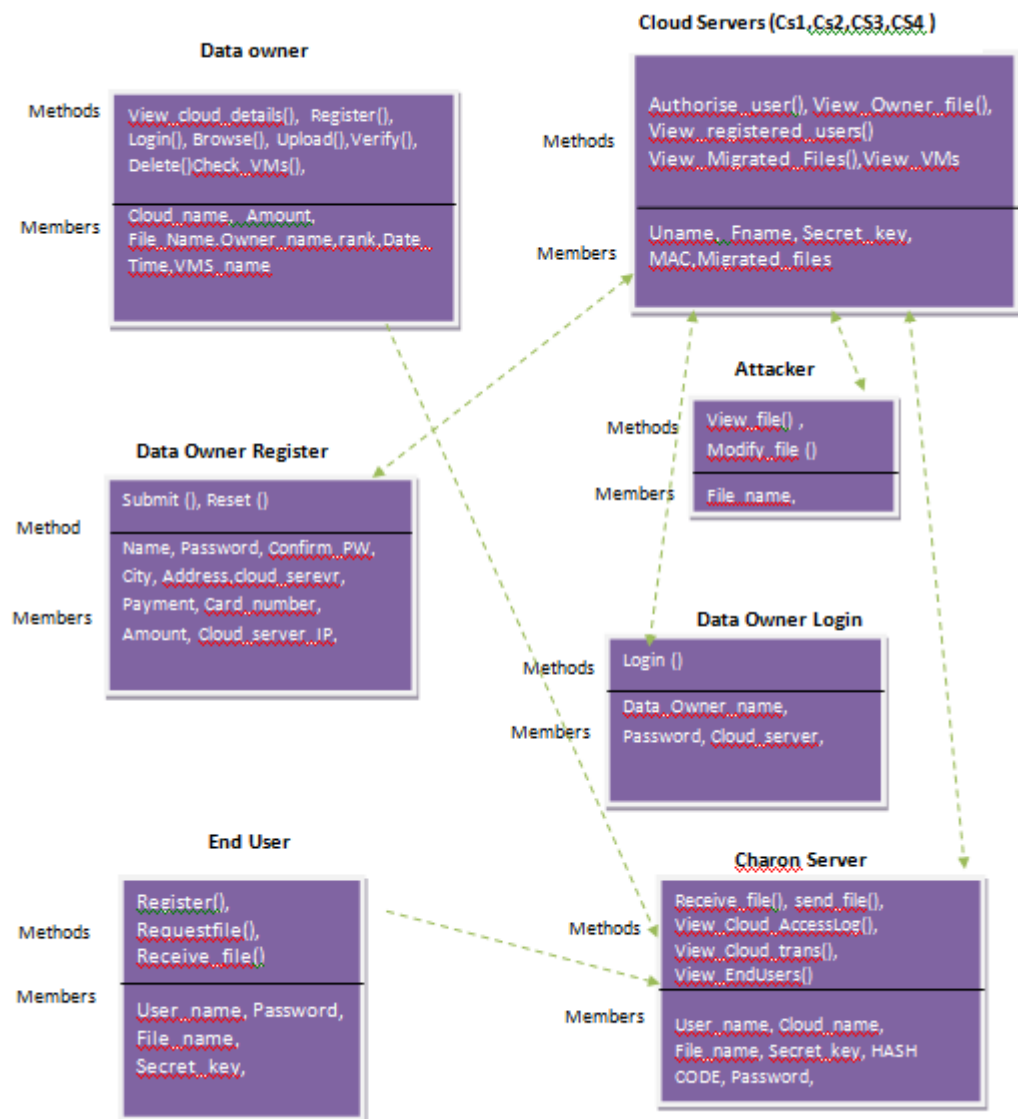


Fig: 4.3.2 CLASS DIAGRAM

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.

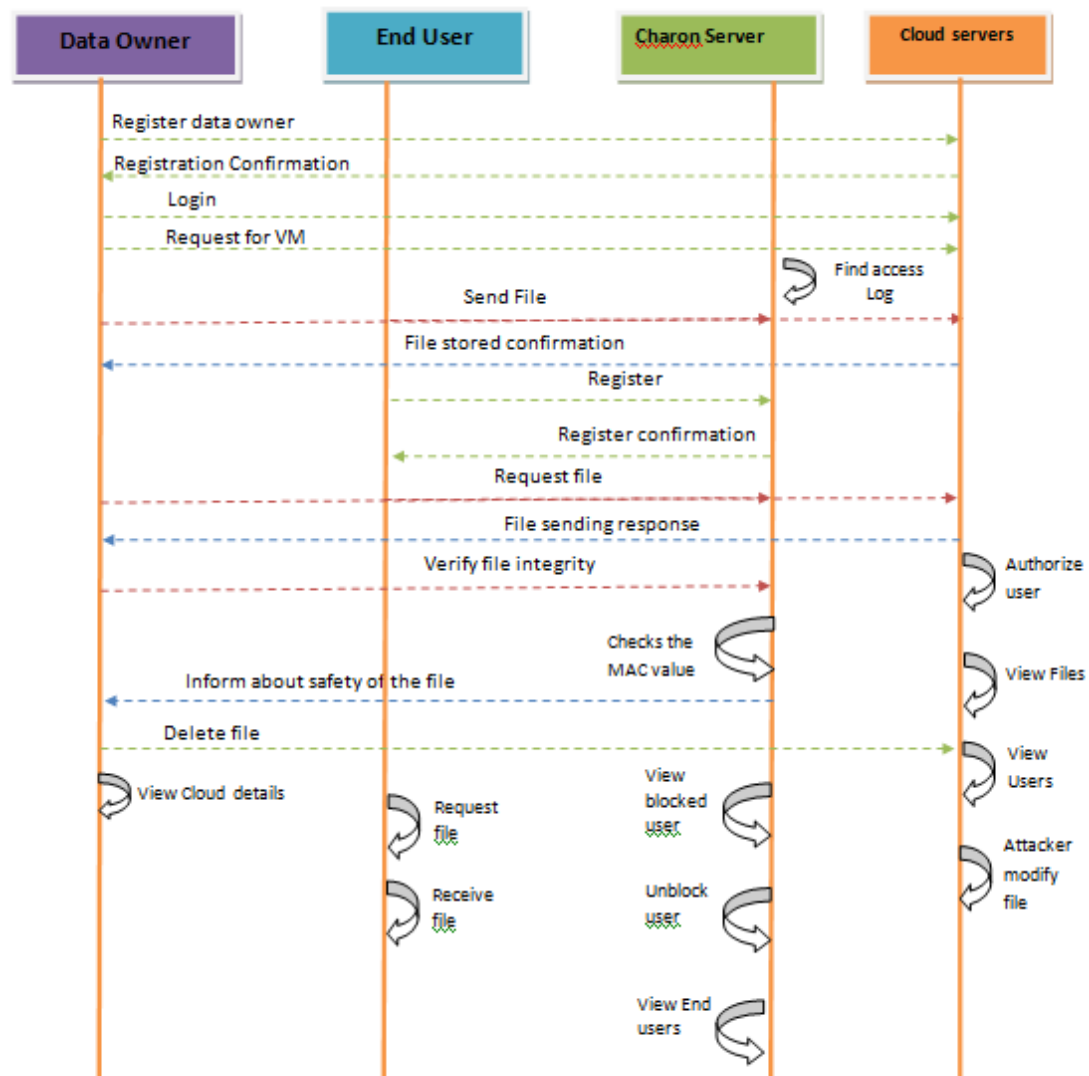


Fig: 4.3.3 SEQUENCE DIAGRAM

5.IMPLEMENTATION METHODOLOGY

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 it's 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 option.

It is the phase that focuses on 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.

The application is implemented in the Internet Information Services 5.0 web server under the windows 2000 Professional and accessed from various clients. An analysis of user training focuses on two factors

- User capabilities
- Nature of the system

Users range from the native to highly sophisticated. Hence they should be trained about the usage of software. The user should takes care to see that in the event of interruption due to power failure.

6. SOFTWARE ENVIRONMENT

6.1 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.

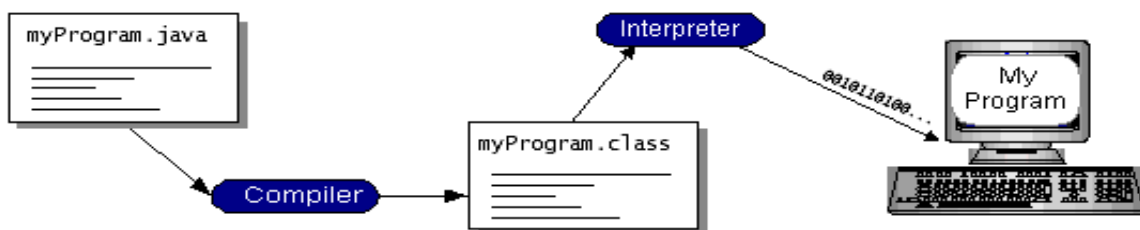


Fig: 6.1 Java Programmig Structure

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.

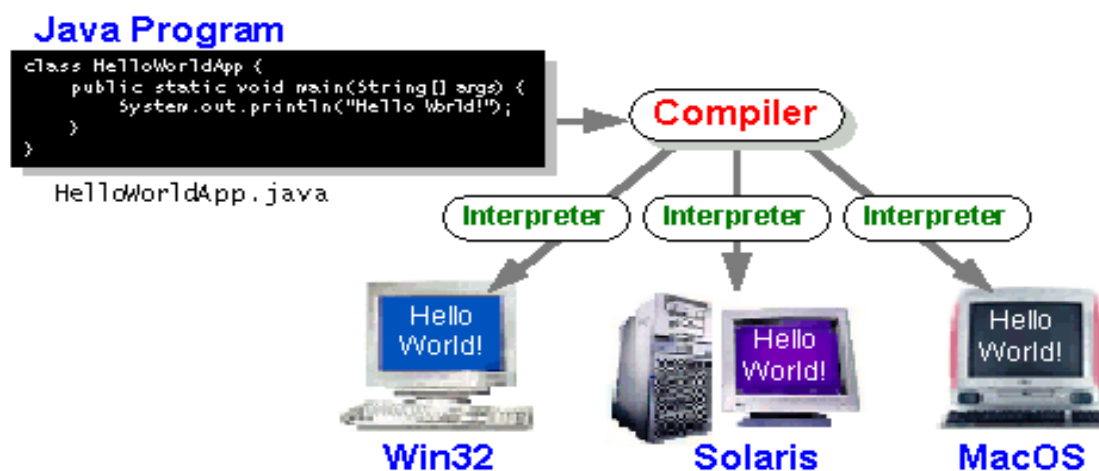


Fig: 6.1.2 Java Compiler Can Run Any Where

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 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.

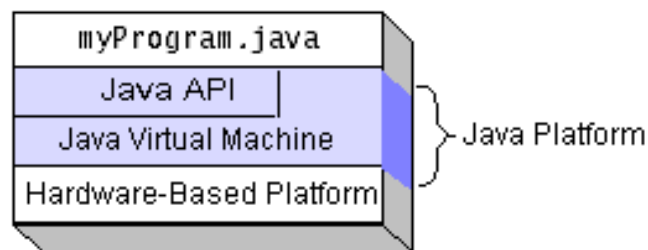


Fig: 6.1.3 Java Platform Diagram

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

What Can Java Technology Do?

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

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

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI

scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

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 JavaBeans, can plug into existing component architectures.

Object serialization:

Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Java Database Connectivity (JDBC™):

Provides uniform access to a wide range of relational databases.

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

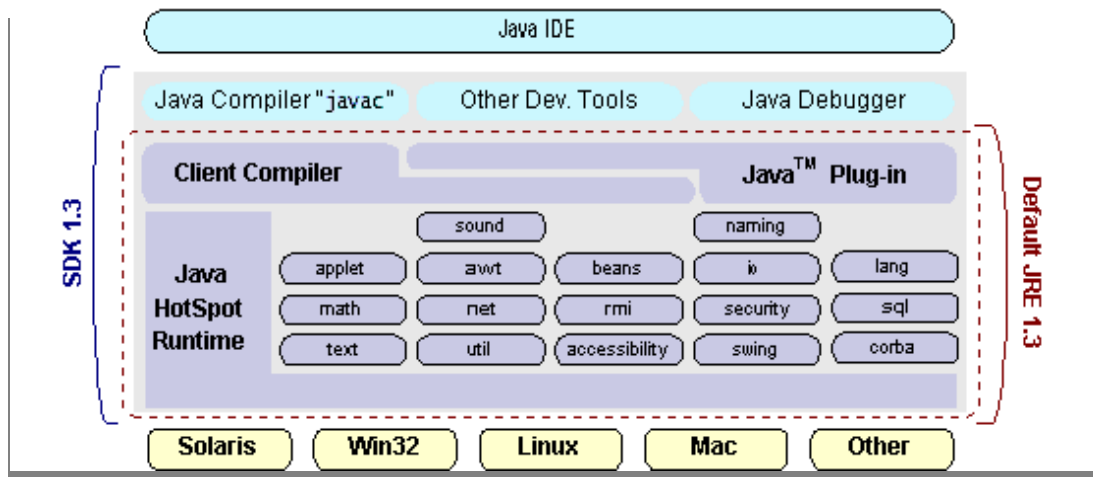


Fig: 6.1.4 Java SDK

How Will Java Technology Change My Life?

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

Get started quickly:

Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.

Write less code:

Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.

Write better code:

The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.

Develop programs more quickly:

Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.

Avoid platform dependencies with 100% Pure Java:

You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.

Write once, run anywhere:

Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.

6.2 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.

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.

6.3 JDBC

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

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

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after. The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

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

The seven 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 JavaNetworking. And for dynamically updating the cache table we go for MSAccess 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
- Object-oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture-neutral
- Portable
- High-performance
- multithreaded
- Dynamic

Java is also unusual in that each Java program is both compiled and interpreted. With a compile 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.

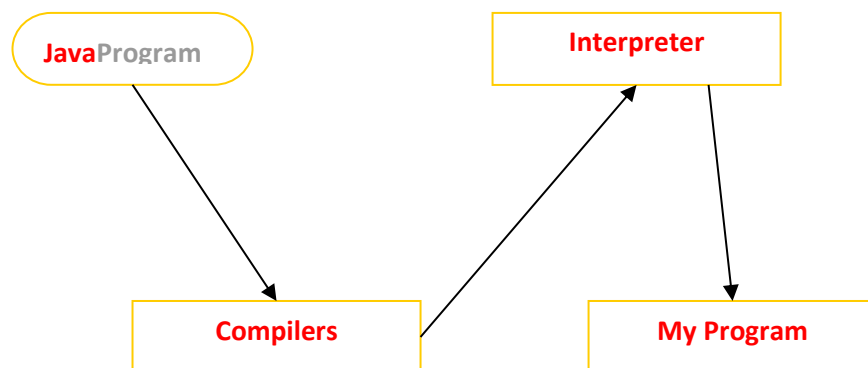
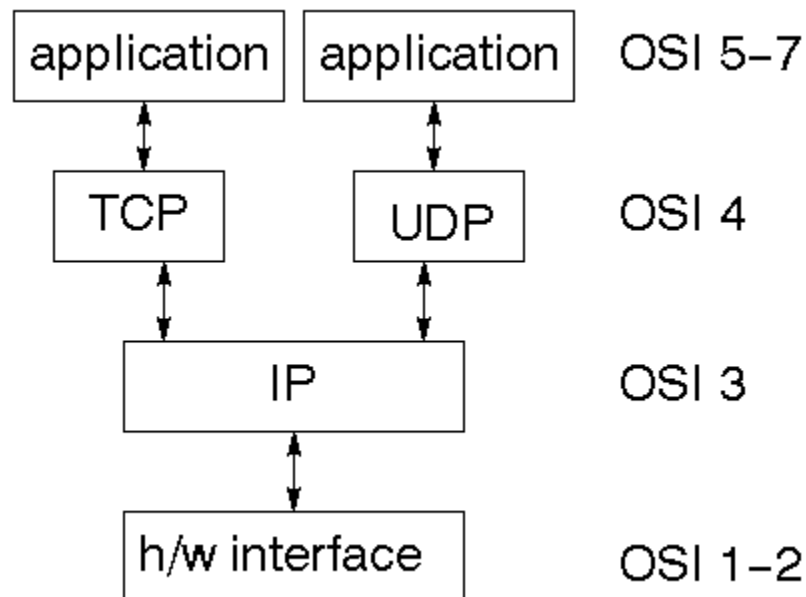


Fig: 6.3.1 Java overview

6.4 NETWORK CONCEPTS

TCP/IP stack

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



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

IP datagram's

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

UDP

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

TCP

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

Internet addresses

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

Network address

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

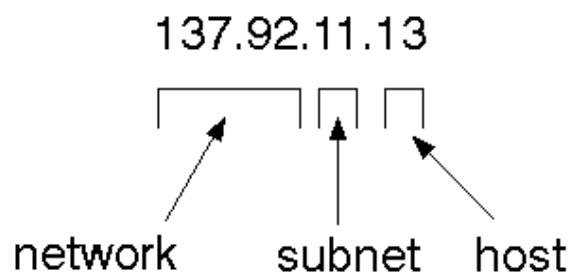
Subnet address

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

Host Address

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

Total address



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

Port addresses

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

Sockets

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

```
#include <sys/types.h>
#include <sys/socket.h>

int socket (int family, int type, int protocol);
```

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

6.5 J2ME

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

1. General J2ME architecture

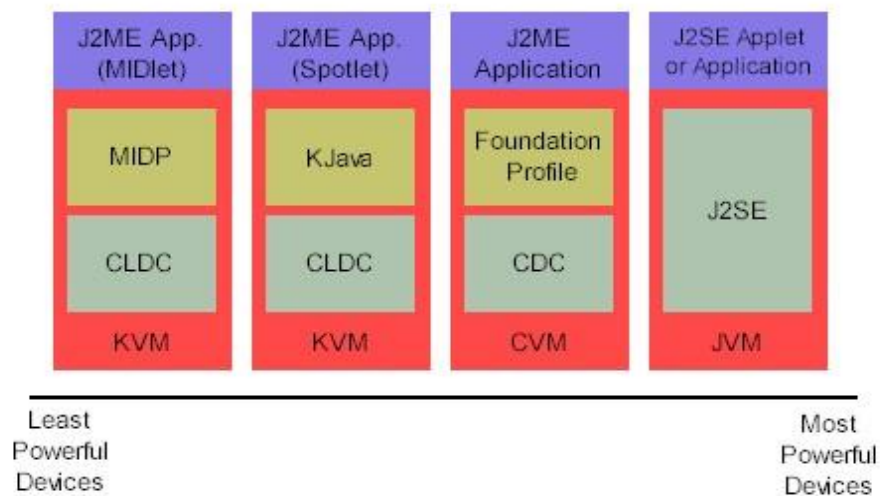


Fig: 6.5.1 J2ME Architecture

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the following graphic depicts the relationship between the different virtual machines, configurations, and profiles.

2. Developing J2ME applications

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

3. Design considerations for small devices

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

- * **Smaller is better.** This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.
- * **Minimize run-time memory use.** To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector.

4. Configuration overview

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

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

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

5. J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

6.6 TOMCAT SERVER

Tomcat is an open-source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs WebLogic, is one of the popular application servers).

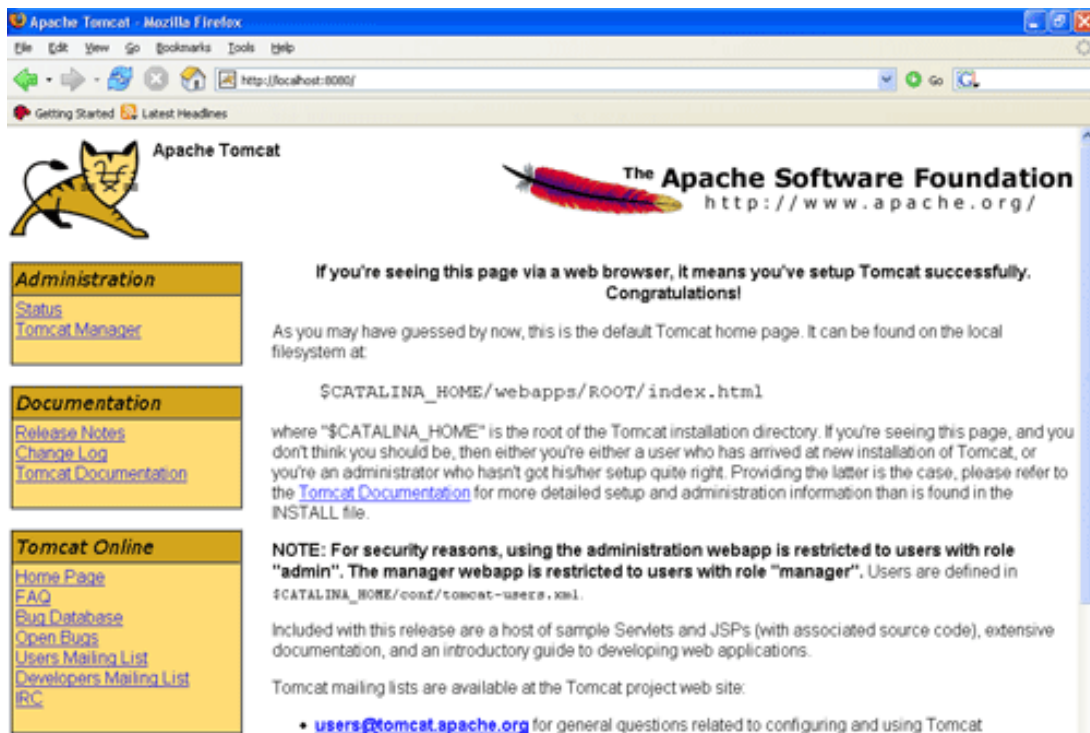


Fig: 6.6.1 Apache Tomcat 7.0

Tomcat's architecture is built upon a modular design, allowing it to function as a standalone server or be integrated seamlessly with other web servers like Apache HTTP Server. Its core components include the Catalina servlet container, which handles the execution of Java Servlets and JSP pages, and the Coyote connector, responsible for managing communication between Tomcat and external clients using various protocols such as HTTP.

Key Features:

Servlet and JSP Support:

Tomcat provides full support for Java Servlets and Java Server Pages, enabling developers to build dynamic and interactive web applications.

Open-Source Nature:

Being open-source, Tomcat is freely available, fostering a vibrant community of developers who contribute to its improvement and ensure its continual evolution.

Platform Independence:

Tomcat is platform-independent, making it compatible with various operating systems, including Windows, Linux, and macOS.

Scalability:

Tomcat offers scalability, allowing it to efficiently handle a growing number of requests by adding more resources or by clustering multiple Tomcat instances.

Security Features:

Security is a top priority for Tomcat, with features such as SSL/TLS support, authentication, and authorization mechanisms, safeguarding web applications from potential threats.

Management and Monitoring:

Tomcat includes a web-based management interface, the Tomcat Manager, facilitating easy deployment, undeployment, and monitoring of applications. Additionally, tools like JConsole and JVisualVM can be employed for in-depth performance analysis.

Deploying applications on Tomcat is a straightforward process. Developers can package their applications as WAR (Web Application Archive) files, which can then be deployed to Tomcat using the Tomcat Manager or by placing the WAR files directly into the designated deployment directory.

Apache Tomcat continues to be a stalwart choice for developers seeking a reliable, scalable, and open-source solution for hosting Java web applications.

7. TESTING AND VALIDATION

7.1 Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

The multifaceted nature of testing involves the exercise of software with the explicit intent of verifying its functionality and performance across diverse scenarios. This process encompasses various types of tests, each tailored to address specific testing requirements. Unit testing focuses on evaluating individual components, ensuring their proper functionality in isolation. Integration testing delves into the interaction between integrated components, validating the coherence of the software as a whole. System testing takes a broader perspective, examining the overall compliance of the software with stipulated requirements, while acceptance testing ensures that the software aligns with user acceptance criteria.

In essence, testing provides a robust mechanism for quality control and risk mitigation. By subjecting the software to a battery of tests, developers and quality assurance teams can identify and rectify defects before the software is deployed to end-users. This proactive approach not only enhances the reliability and performance of the software but also contributes to overall customer satisfaction. As an integral part of the software development life cycle, testing safeguards against potential issues, thereby fostering the delivery of a high-quality product that not only meets technical specifications but also surpasses user expectations

7.2 TYPES OF TESTS

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.

Integration Testing

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

Functional Testing

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

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.

7.3 Validation

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.

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.

White Box Testing

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

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.

Unit Testing:

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

Test strategy and approach

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

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

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

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g., components in a software system or one step up software applications at the company level – interact without error.

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

Acceptance Testing

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

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

8. OUTPUT SCREENS

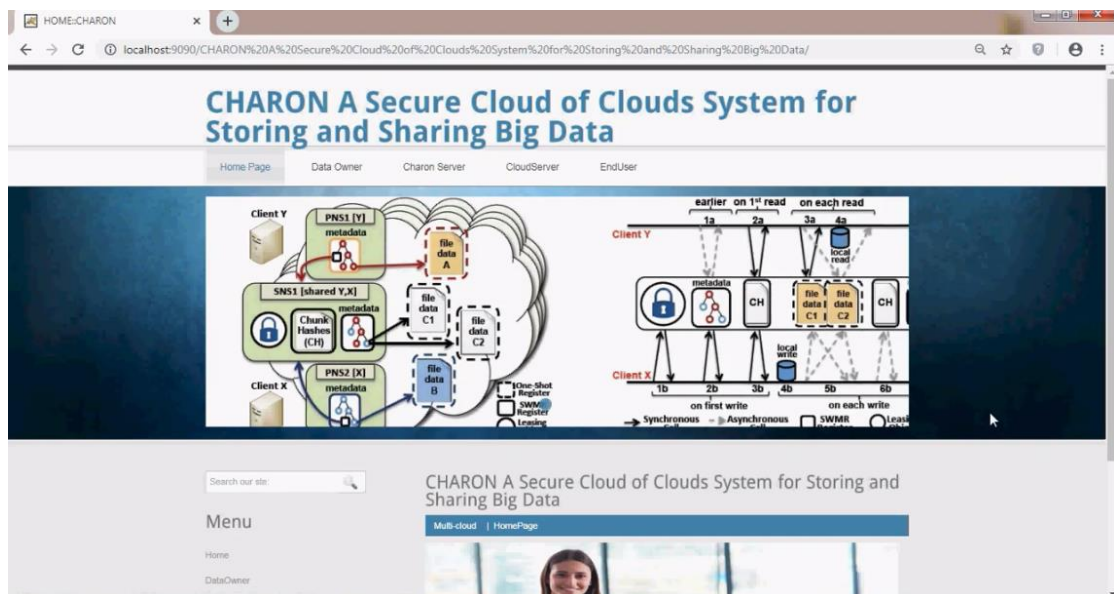


FIGURE 8.1 HOME PAGE LOGIN

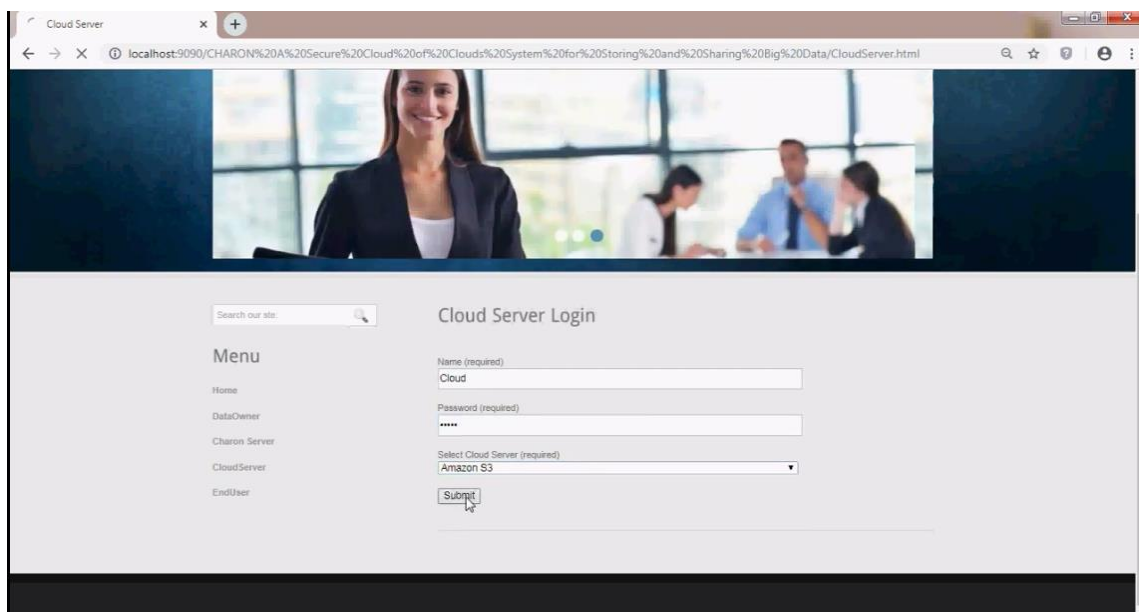


FIGURE 8.2 CLOUD SERVER LOGIN

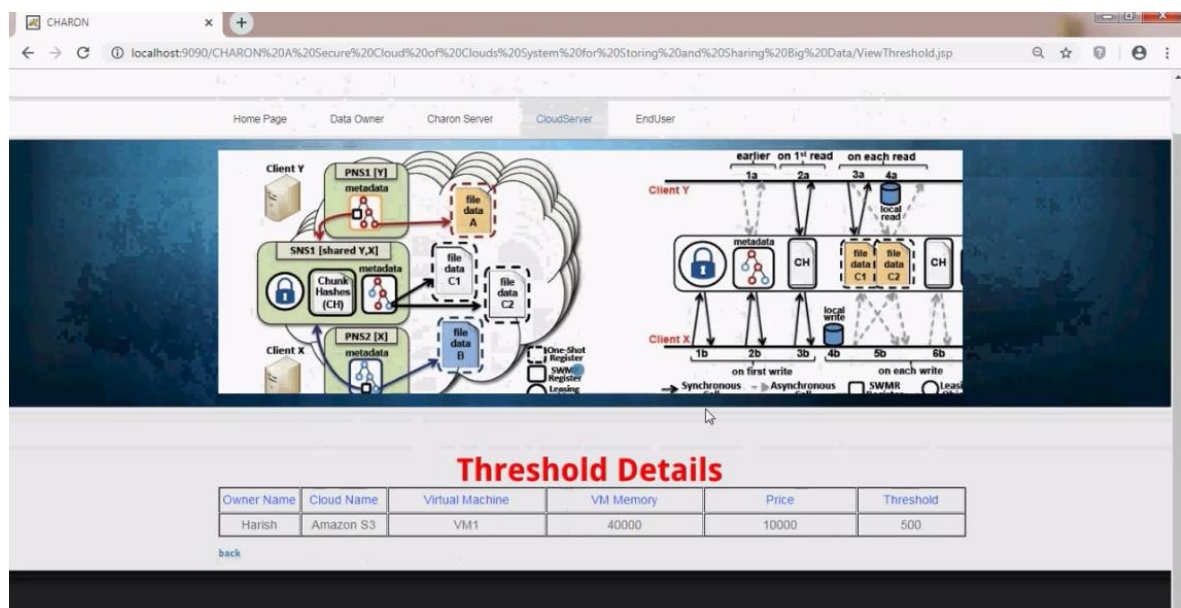


FIGURE 8.3 THRESHOLD DETAILS

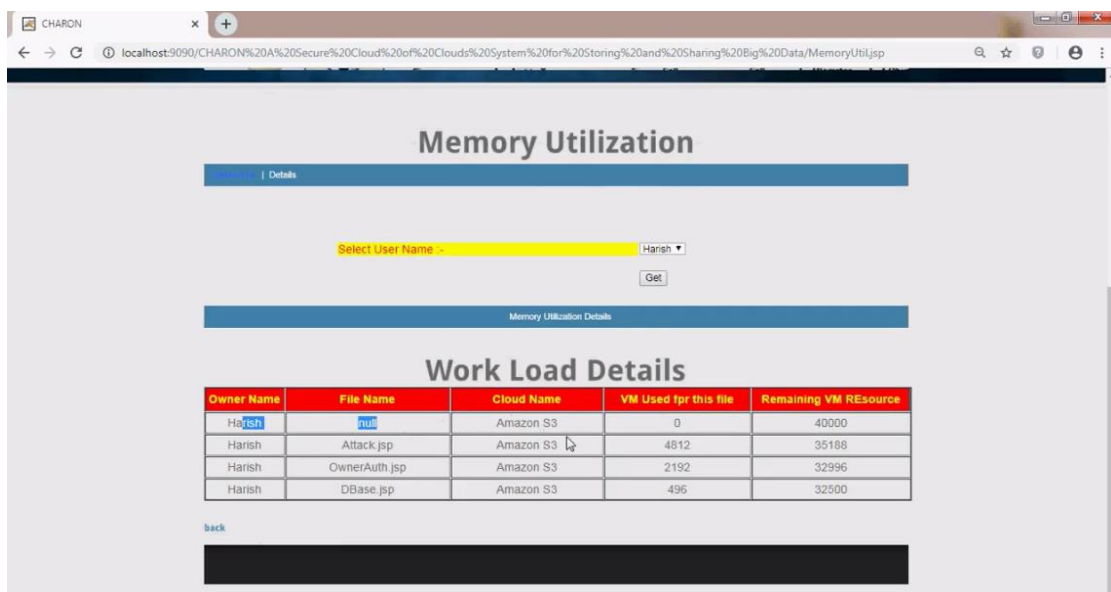
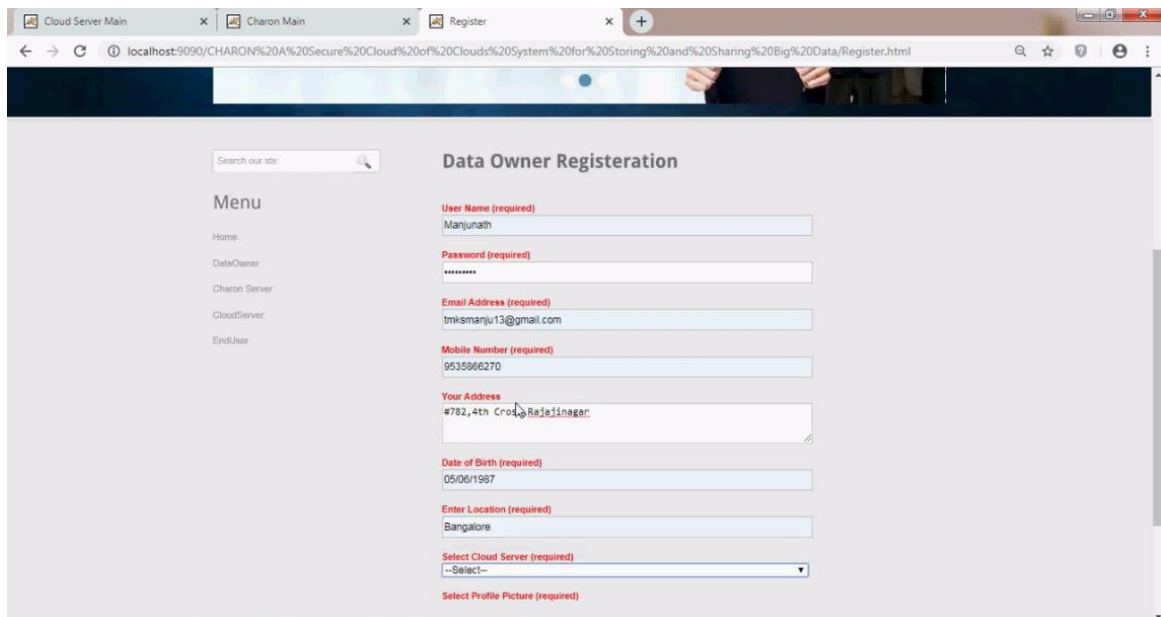


FIGURE 8.4 WORK LOAD DETAILS



Cloud Server Main x Charon Main x Register x

localhost:9090/CHARON%20A%20Secure%20Cloud%20of%20Clouds%20System%20for%20Storing%20and%20Sharing%20Big%20Data/Register.html

Search our site

Menu

- Home
- DataOwner
- Charon Server
- CloudServer
- EndUser

Data Owner Registration

User Name (required)
Manjunath

Password (required)

Email Address (required)
tnkismanju13@gmail.com

Mobile Number (required)
9535866270

Your Address
782, 4th Cross Rajalinggar

Date of Birth (required)
05/06/1987

Enter Location (required)
Bangalore

Select Cloud Server (required)
--Select--

Select Profile Picture (required)

FIGURE 8.5 DATA OWNER REGISTRATION

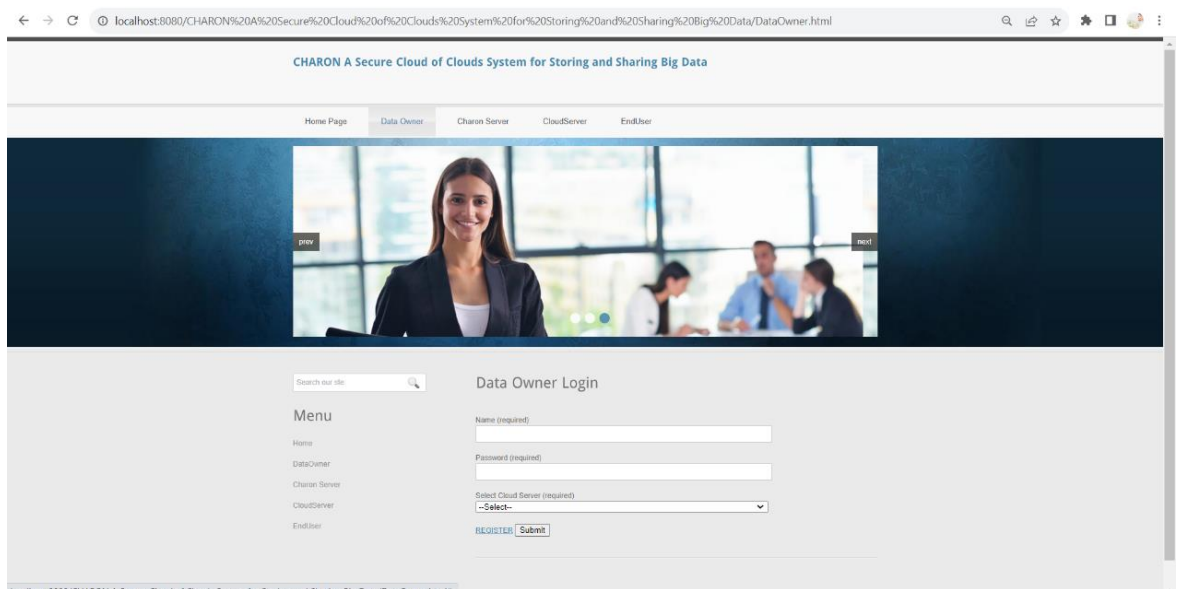


FIGURE 8.6 DATA OWNER LOGIN

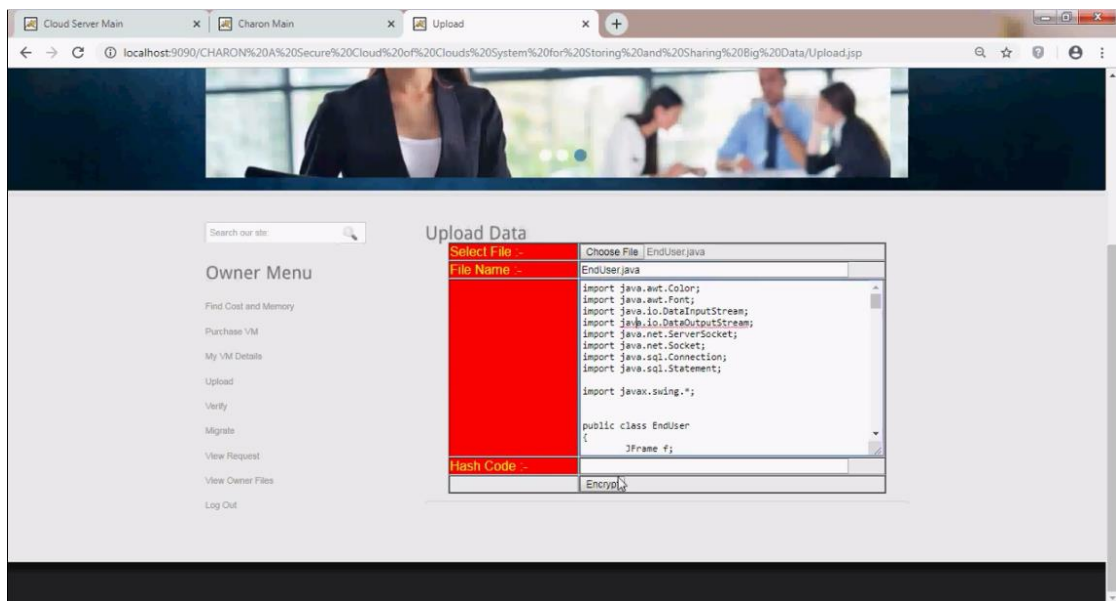


FIGURE 8.7 UPLOAD DATA

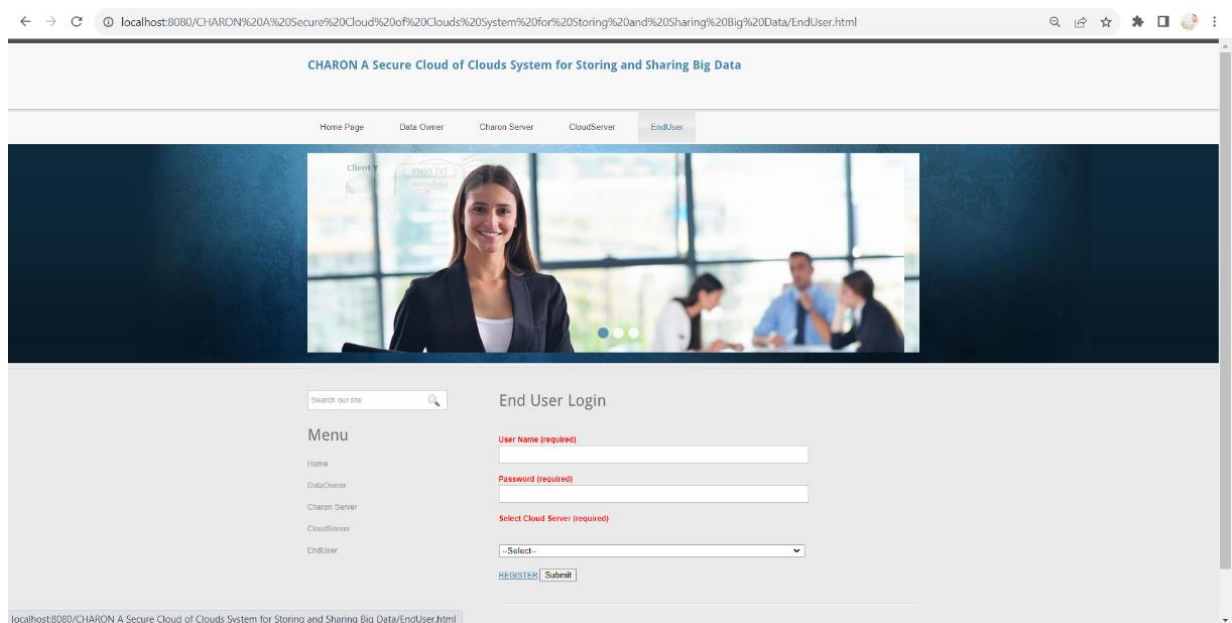


FIGURE 8.8 END USER LOGIN

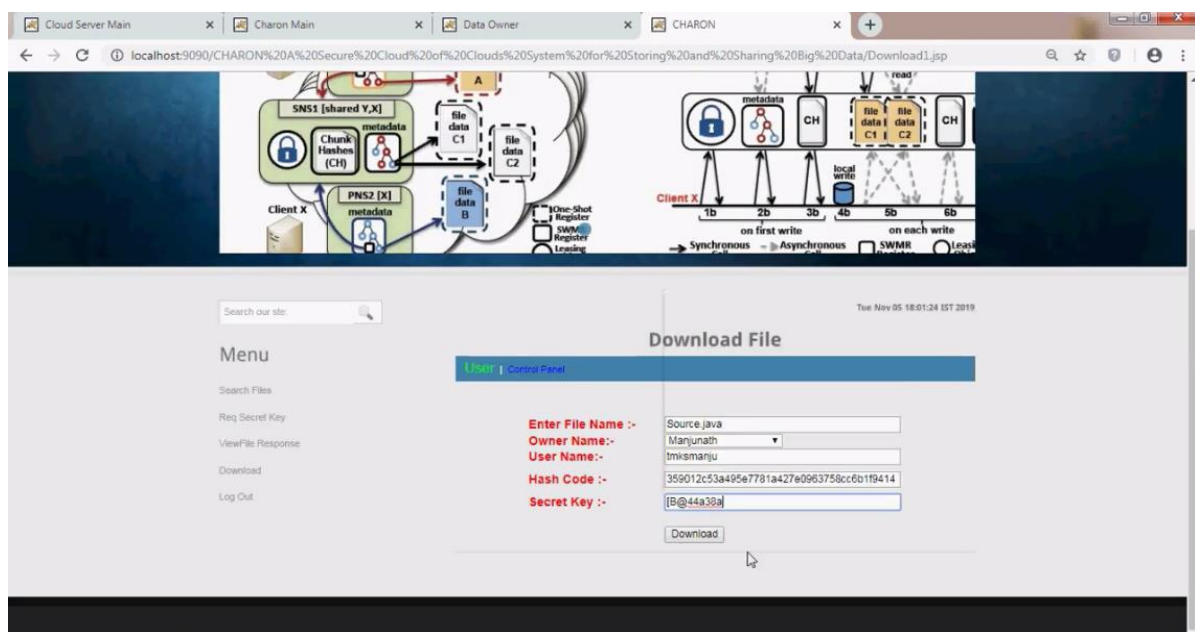


FIGURE 8.9 DOWNLOAD FILE

9. CONCLUSION

CHARON is a cloud-backed file system for storing and sharing big data. Its design relies on two important principles: files metadata and data are stored in multiple clouds, without requiring trust on any of them individually, and the system is completely data centric. This design has led us to develop a novel Byzantine resilient leasing protocol to avoid write-write conflicts without any custom server. Our results show that this design is feasible and can be employed in real-world institutions that need to store and share large critical datasets in a controlled way.

10. REFERENCES

- [1] Cloud Harmony, “Service Status,” <https://cloudharmony.com/status-of-storage-group-by-regions>, 2019.
- [2] Cloud Security Alliance, “Top Threats,” <https://cloudsecurityalliance.org/group/top-threats/>, 2016.
- [3] M. A. C. Dekker, “Critical Cloud Computing: A CIIP perspective on cloud computing services (v1.0),” European Network and Information Security Agency (ENISA), Tech. Rep., 2012.
- [4] H. S. Gunawi et al., “Why does the cloud stop computing?: Lessons from hundreds of service outages,” in Proc. of the SoCC, 2016.
- [5] European Commission, “Data protection,” <https://ec.europa.eu/info/law/law-topic/data-protection>, 2018.
- [6] G. Gaskell and M. W. Bauer, Genomics and Society: Legal, Ethical and Social Dimensions. Routledge, 2013.
- [7] A. Bessani et al., “BiobankCloud: a platform for the secure storage, sharing, and processing of large biomedical data sets,” in DMAH, 2015.
- [8] H. Gottweis et al., “Biobanks for Europe: A challenge for governance,” European Commission, Directorate-General for Research and Innovation, Tech. Rep., 2012.
- [9] P. E. Verissimo and A. Bessani, “E-biobanking: What have you done to my cell samples?” IEEE Security Privacy, vol. 11, no. 6, pp. 62–65, 2013.
- [10] P. R. Burton et al., “Size matters: just how big is big? Quantifying realistic sample size requirements for human genome epidemiology,” Int J Epidemiol, vol. 38, no. 1, pp. 263–273, 2009.
- [11] D. Haussler et al., “A million cancer genome warehouse,” University of Berkley, Dept. of Electrical Engineering and Computer Science, Tech. Rep., 2012.
- [12] R. W. G. Watson, E. W. Kay, and D. Smith, “Integrating biobanks: addressing the practical and ethical issues to deliver a valuable tool for cancer research,” Nature Reviews Cancer, vol. 10, no. 9, pp. 646–651, 2010.

- [13] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, “RACS: A case for cloud storage diversity.” SoCC, pp. 229–240, 2010.
- [14] C. Basescu et al., “Robust data sharing with key-value stores,” in Proc. of the DSN, 2012.
- [15] A. Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa, “DepSky: Dependable and secure storage in cloud-of-clouds,” ACM Trans. Storage, vol. 9, no. 4, pp. 12:1–12:33, 2013.
- [16] T. Oliveira, R. Mendes, and A. Bessani, “Exploring key-value stores in multi-writer Byzantine-resilient register emulations,” in Proc. of the OPODIS, 2016.
- [17] Amazon, “Amazon S3,” <http://aws.amazon.com/s3/>, 2019.
- [18] Microsoft, “Microsoft Azure Queue,” <http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-queues/>, 2019.
- [19] B. Martens, M. Walterbusch, and F. Teuteberg, “Costing of cloud computing services: A total cost of ownership approach,” in Proc. of the HICSS, 2012.
- [20] J. Y. Chung, C. Joe-Wong, S. Ha, J. W.-K. Hong, and M. Chiang, “CYRUS: Towards client-defined cloud storage,” in Proc. of the EuroSys, 2015.
- [21] S. Han et al., “MetaSync: File synchronization across multiple untrusted storage services,” in Proc. of the USENIX ATC, 2015.
- [22] H. Tang, F. Liu, G. Shen, Y. Jin, and C. Guo, “UniDrive: Synergize multiple consumer cloud storage services,” in Proc. of the Middleware, 2015.
- [23] D. Dobre, P. Viotti, and M. Vukolic, “Hybris: Robust hybrid cloud storage.” in Proc. of the SoCC, 2014.
- [24] A. Bessani et al., “SCFS: a shared cloud-backed file system,” in Proc. Of the USENIX ATC, 2014.
- [25] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, “SPANStore: Cost-effective geo-replicated storage spanning multiple cloud services,” in Proc. of the SOSP, 2013.