

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELAGAVI-590018, KARNATAKA



**A Project Report
on**

**An Efficient and Secured OTP Enabled File Sharing Service
Over Big Data Environment**

*Submitted in partial fulfillment of the requirements for the VIII Semester of degree of
Bachelor of Engineering in Information Science and Engineering of Visvesvaraya
Technological University, Belagavi*

Submitted by

Chetan Balaji	1RN16IS026	Chetan N S	1RN16IS027
D Nanda Kishore	1RN16IS029	Manoj M	1RN16IS050

Under the Guidance of

Dr S Sathish Kumar
Professor
Department of ISE



Department of Information Science and Engineering

RNS Institute of Technology

**Dr. Vishnuvaradhana Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru-560098**

2019-2020

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhana Road, RajarajeshwariNagar post,
Channasandra,Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work entitled *An Efficient and Secured OTP Enabled File Sharing Service over Big Data Environment* has been successfully completed by **Chetan Balaji (1RN16IS026)**, **Chetan N S (1RN16IS027)**, **D Nanda Kishore (1RN16IS029)** and **Manoj M (1RN16IS050)**, bonafide students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of degree in **Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belgaum** during academic year **2019-2020**. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Dr S Sathish Kumar
Project Guide
Professor
Department of ISE

Dr. M V Sudhamani
Professor and HOD
Department of ISE
RNSIT

Dr. M K Venkatesha
Principal
RNSIT

Name of the Examiners

External Viva

Signature with Date

1. _____

1. _____

2. _____

2. _____

DECLARATION

We **Chetan Balaji** [USN: 1RN16IS026], **Chetan N S** [USN: 1RN16IS027], **D Nanda Kishore** [USN: 1RN16IS029] and **Manoj M** [USN: 1RN16IS050] students of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Project entitled *An Efficient and Secured OTP Enabled File Sharing service over Big Data Environment* has been carried out by us and submitted in partial fulfillment of the requirements for the *VIII Semester of degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belgaum* during academic year 2019-2020.

Place : Bengaluru

Date : 19-05-2020

Chetan Balaji (1RN16IS026)

Chetan N S (1RN16IS027)

D Nanda Kishore (1RN16IS029)

Manoj M (1RN16IS050)

ABSTRACT

The high volume, velocity, and variety of data being produced by diverse scientific and business domains challenge standard solutions of data management, requiring them to scale while ensuring security and dependability. A fundamental problem is where and how to store the vast amount of data that is being continuously generated. Private infrastructures are the first option for many organizations. However, creating and maintaining data centers is expensive, requires specialized workforce, and can create hurdles to sharing. Conversely, attributes like cost-effectiveness, ease of use, and (almost) infinite scalability make public cloud services natural candidates to address data storage problems.

File sharing has been an essential part of this century. Using various applications, files can be shared to large number of users. For the purpose of storage, the Hadoop Distributed File System (HDFS) can be used. The HDFS handles large size of files in a single server. Common sharing methods like removable media, servers or computer network, World Wide Web-based hyperlink documents.

In the proposed project, the files are merged using MapReduce programming model on Hadoop. This process improves the performance of Hadoop by rejecting the files which are larger than the size of Hadoop and reduces the memory size required by the Name Node.

ACKNOWLEDGMENT

At the very onset we would like to place our gratefulness to all those people who helped us in making this project a successful one.

Coming up, this project to be a success was not easy. Apart from the sheer effort, the enlightenment of our very experienced teachers also plays a paramount role because it is they who guide us in the right direction.

First of all, we would like to thank **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of project work.

In this regard, we express our sincere gratitude to the Director **Dr. H N Shivashankar** and the principal **Dr. M K Venkatesha**, for providing us all the facilities in this college.

We are extremely grateful to our own and beloved Professor and Head of Department of Information science and Engineering, **Dr. M V Sudhamani**, for having accepted to patronize us in the right direction with all her wisdom.

We place our heartfelt thanks to **DR S SATHISH KUMAR**, Professor, Department of information science and Engineering for having guided for project and all the staff members of our department for helping us out at all times.

We thank Mrs. **Kusuma S** and Mr. **Rajkumar R**, Project coordinators, Department of Information Science and Engineering. We thank our beloved friends for having supported us with all their strength and might. Last but not the least, we thank our parents for supporting and encouraging us throughout. We made an honest effort in this assignment.

Chetan Balaji

Chetan N S

D Nanda Kishore

Manoj M

TABLE OF CONTENTS

CERTIFICATE	
ABSTRACT	i
ACKNOWLEDGMENT	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	iv
LIST OF TABLES	v
ABBREVIATIONS	vi
1. INTRODUCTION	1
1.1 Background	1
1.2 Existing Systems and their Drawbacks	2
1.3 Proposed System	2
1.4 Advantages of the proposed system	4
2. LITERATURE REVIEW	5
3. ANALYSIS	23
3.1 Problem Identification	23
3.2 Objectives	26
3.3 Methodology	26
3.4 System Requirement Specification	28
3.4.1 Software Requirement Specification	28
3.4.2 Hardware Requirement Specification	30
3.4.3 Functional Requirement	30
3.4.4 Non-Functional Requirement	31
4. SYSTEM DESIGN	34

4.1	System Architecture	35
4.2	Detailed Design	35
4.2.1	High Level Design	35
4.2.2	Low Level Design	37
4.3	Data Flow Diagram	42
4.4	Use Case Diagram	44
4.5	Sequence Diagram	45
4.6	Class Diagram	46
5.	IMPLEMENTATION	49
5.1	Overview of System Implementation	49
5.2	Pseudocode	51
5.3	Implementation Support	60
6	TESTING	62
6.1	Introduction	62
6.2	Unit Testing	63
6.3	Integration Testing	65
6.4	System Testing	66
6.5	Validation Testing	66
6.6	User Acceptance Testing	67
7	DISCUSSION OF RESULTS	68
7.1	Snapshots	68
8.	CONCLUSION & FUTURE WORK	79
	REFERENCES	80

LIST OF FIGURES

Fig.No	Descriptions	Page
1.1	Map-Reduce Architecture	3
4.1	System Architecture Diagram	34
4.2	Slave Architecture Diagram	37
4.3	Profile Operation Module Interaction Diagram	39
4.4	File Upload Process Description	40
4.5	Managing files Servlet Interaction	41
4.6	Flow of access control on shared files	42
4.7	Dataflow Diagram	43
4.8	System Use Case	44
4.9	Sequence Diagram for OTP Generation	45
4.10	Sequence Diagram for OTP Verification	45
4.11	Class-Diagram 1(User Dao)	46
4.12	Class Diagram-2(File-Upload Servlet)	47
4.13	Class Diagram-3(Access Granting)	47
4.14	Class Diagram-4(OTP Authentication)	48
5.1	Directory Structure	52
7.1	Namenode Information	68
7.2	Index Page	69
7.3	Register Page	69
7.4	Login Page	70
7.5	Welcome Page	70
7.6	User Profile Operation	71
7.7	Update Profile Page	71
7.8	Change Password Page	72
7.9	HDFS Operations Page	72

7.10	Myfile Page	73
7.11	Add File Page	73
7.12	Manage File	74
7.13	Delete File	74
7.14	File Sharing	75
7.15	Change or Revoke access	75
7.16	Shared File Page	76
7.17	Getting Access	76
7.18	Verified OTP	77
7.19	Email OTP	77
7.20	Mobile OTP	78

LIST OF TABLES

Fig. No	Descriptions	Page
3.1	Software Requirements	29
3.2	Hardware Requirements	30
6.1	Unit testing	63
6.2	Integration Testing	65
6.3	System Testing	66

LIST OF ABBREVIATIONS

HTML	Hypertext Markup Language
HDFS	Hadoop Distributed File System
HPCC	High-Performance Computing Cluster
OTP	One-Time Password
YARN	Yet Another Resource Negotiator
W3C	World Wide Web Consortium
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing

Chapter 1

INTRODUCTION

1.1 Background

File sharing means sending and receiving of different types of file (audio, video, and picture) within the same network or different network. File sharing is done using techniques like file storage, distribution and transmission. File sharing is the act of circulating or giving access to carefully put away data, for example, PC programs, media (sound, pictures and video), reports, or electronic books. It might be actualized through an assortment of ways. Earlier file sharing applications have a drawback, once the file sharing link is discoverable, the file can be accessed by unauthorized user. The Hadoop system itself is for the most part written in the Java programming dialect, with some local code in C and charge line utilities composed as shell contents.

File sharing has been an essential part of this century. Using various applications, files can be shared to large number of users. For the purpose of storage, the Hadoop Distributed File System (HDFS) can be used.

HDFS is mainly used for the unstructured data analysis. The HDFS handles large size of files in a single server. Common sharing methods like removable media, servers or computer network, World Wide Web-based hyperlink documents. In the proposed project, the files are merged using MapReduce programming model on Hadoop. This process improves the performance of Hadoop by rejecting the files which are larger than the size of Hadoop and reduces the memory size required by the Name Node.

1.2 Existing Systems and their drawbacks

Previous works show that file sharing applications have a drawback, once the file sharing link is discoverable, the file can be accessed by unauthorized user. Hadoop is an open-source programming structure utilized for circulated capacity and handling of dataset of enormous information utilizing the MapReduce programming model. Managing big data is being a big problem to many organizations.

Due to this reason, file sharing faces many problems. Every day 2.5 Quintillion bytes are created. Hadoop can be easily handled by Hadoop File Distributed System (HDFS). Considering the drawbacks of the previous system, a new system can be proposed which has an advanced security i.e. OTP service.

The sender will send the required file to receiver. While opening the file, the receiver must enter an OTP number, which he will receive from the sender. This provides the system more security as the OTP is accessible to authorized user. The data owner will be capable of revoking the access to the users at any point of time. The system will also take care that the shared user will have no rights to further share the data with others. Also, this system is implemented for all the types of data, i.e., text, audio, video, images etc.

1.3 Proposed System

This proposed system has an advanced security i.e. OTP service. The sender will send the required file to receiver. While opening the file, the receiver must enter an OTP number, which he will receive from the sender. This provides the system more security as the OTP is accessible to authorized user. The data owner will be capable of revoking the access to the users at any point of time. The system will also take care that the shared user will have no rights to further share the data with others. Also, the system is implemented for all the types of data, i.e, text, audio, video, images etc

The main concept behind the Hadoop file system is the MapReduce. MapReduce is a concept which was developed by Google to sort the large data which is in petabytes by forming clusters of this data. MapReduce mainly has two parts i.e.-Map and Reduce.

Map is a transformation step in which the singular records are handled in parallel. Reduce is a summarization step in which all the related records are handled in a single entity. The concept of MapReduce is that initially all the data is divided into small parts called as chunks. These chunks are then individually processed by a map task. Later these chunks are physically partitioned and sorted accordingly.

Each sorted chunk is send to a reduce task. The figure below shows the MapReduce concept in detail. Each record will be split into several parts (chunks). A map task can run on any node in the computer and there maybe multiple map tasks running simultaneously. The map task converts each record into a key/value pair.

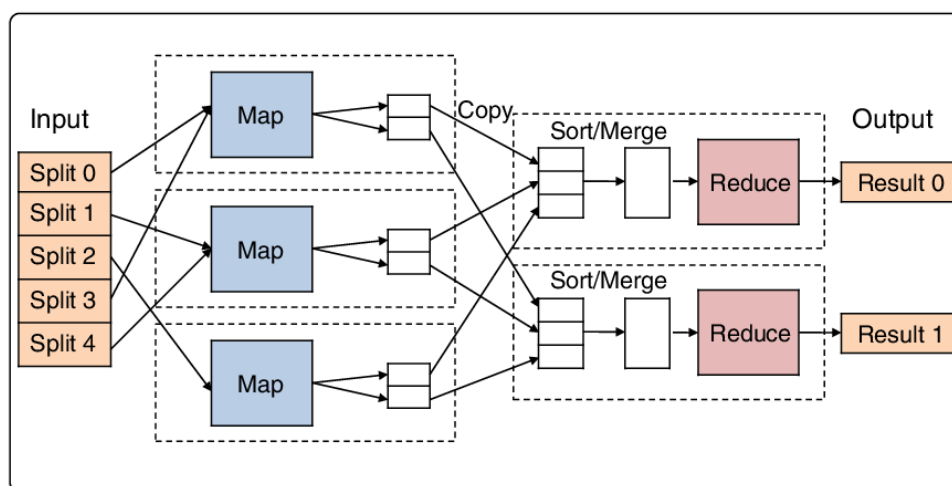


Figure 1.1: Map-Reduce Architecture

This key value pair will be partitioned and sorted. Each partition will then have a reduce task. Each partition's keys and values will have separate reduce tasks. There may be multiple reduce task running at a given time Any developer needs to provide to the Hadoop Framework 4 items i.e. a class which will read the input record and transform it into key/value pair, a map class, a reduce class and a class which will transform the key/value pair into output record. MapReduce requires a shared file system. Shared file system does not mean any file level system but it needs a distributed system with a plug in available to the framework. When HDFS will be used as a shared file system, Hadoop has an advantage that it will recognize which node has a physical copy of the input data and will check to read the data that is on the machine.

If the user doesn't want the reduce task, the user need not specify the reduce class. The framework will partition the input and schedule the map task. If needed the output of the map task will be sorted and given to the reduce task and the final output will be given to the user.

The framework has two processes that handle the management of the MapReduce, they are TaskTracker and JobTracker. The TaskTracker provides the individual execution of map and reduce tasks. The JobTracker provides job submission, monitoring and control. The Hadoop File Distributed System: Is the one which is designed for the MapReduce concept. The HDFS system uses the MapReduce concept to sort large chunks of data, sort them and provide them as large chunks of output files.

The HDFS services are provided using two main processes i.e. NameNode and DataNode. The NameNode manages the file system data by providing the management

and control services. The DataNode provides data storage and retrieval services. Below steps explain in brief how the OTP is used on Hadoop to make the file sharing work.

Step 1: Open the windows application

Step 2: If new user then registers else login with user id and password

Step 3: Select the files to be uploaded

Step 4: Select the file to be shared

Step 5: Enter the user to who file will be sent

Step 6: When the user has to access the shared file, the OTP will be sent to the User's mobile number and Email ID.

Step 7: The receiver will have to enter the OTP to open the file

Step 8: Repeat the steps 4 to 6 any number of time

Step 9: Stop

1.4 Advantages of the Proposed System

The advantages of the proposed system are:

- File sharing capability
- Secured file access and storage
- Access Revocation made available dynamically
- Non-transferrable access implementation
- Multiple file types addressed.

Chapter 2

LITERATURE REVIEW

A Literature Survey describes the concept of how the concept was emerged, how it has been implemented and what the current status is.

[1] Research and Implementation of Distributed Storage System Based on Big Data

According to the security policy research in distributed data and information storage, considered from the data access patterns, this paper designs a system with new storage model and query mechanism for distributed platform and big data. This system can provide data sharing integrity check, while giving a hot backup solution, to improve the safety and maintainability of the cluster, and take a viable option for data migration and recovery when the cluster fails.

Limitations:

- File sharing is not provisioned. It only deals with storing and processing big data.
- Efficiency of file read and write decreases as the number of nodes increases.

[2] JeCache: Just-Enough Data Caching for Just-in-Time Prefetching in Big Data Applications

In this paper, they propose a novel just-enough big data caching scheme for just-in-time block prefetching to improve the cache effectiveness of big data clusters. With just-in-time block prefetching, a block is cached in just before the task begins to process the block, rather than being cached in along with other blocks of the same dataset being processed. They monitor block accesses to measure the average processing time of data blocks, and then estimate the minimal number of blocks that should be kept in cache for a big dataset, so that the speed of data processing matches with that of data prefetching, and each upper level task can obtain its input blocks from cache just in time. Their experimental results show that the proposed cache method can restrain over-requirement of cache resources in big data applications, and provides the same performance improvement as when all data blocks are cached.

Limitations:

- No Reliability
- File sharing is not provisioned. It only deals with storing and processing big data

[3] Achieving Load Balance for Parallel Data Access on Distributed File Systems

In this paper, they propose novel methods, referred to as Opass, to optimize parallel data reads, as well as to reduce the imbalance of parallel writes on distributed file systems. Our proposed methods can benefit parallel data-intensive analysis with various parallel data access strategies. Opass adopts new matching-based algorithms to match processes to data so as to compute the maximum degree of data locality and balanced data access. Furthermore, to reduce the imbalance of parallel writes, Opass employs a heatmap for monitoring the I/O statuses of storage nodes and performs HM-LRU policy to select a local optimal storage node for serving write requests. Experiments are conducted on PROBE's Marmot 128-node cluster testbed and the results from both benchmark and well-known parallel applications show the performance benefits and scalability of Opass.

Limitations:

- Doesn't provide auto-scaling of meta data
- File sharing is not provisioned. It only deals with storing and processing big data

[4] Leveraging Distributed Big Data Storage Support in CLAAaaS for WINGS Workflow Management System

Cloud-based Analytics-as-a-Service (CLAAaaS) was developed by Zulkernine et al. with a goal to simplifying big data analytics users. It provides software-as-a-service access to a variety of back end analytics tools and data stores. One of the tools is the Workflow Instance Generation and Selection (WINGS). WINGS allows users to reuse predefined workflows and their components containing semantic meta-data to define new workflows. The goal of this project is to add support for big data storage systems to WINGS and validate the extensions using multiple data analytic workflows of different complexities with data residing in a variety of back end data sources. The extension allows the CLAAaaS users to create, validate and execute analytic workflows in a distributed environment and use data from multiple big data storage systems. They validate our work using four big data storage systems in WINGS workflows namely, Apache HBase, MongoDB, MySQL with a front-end interface.

Limitations:

- Access time for the files is huge
- File sharing is not provisioned. It only deals with storing and processing big data

[5] Quick Answer for Big Data in Sharing Economy: Innovative Computer Architecture Design Facilitating Optimal Service Demand Matching

In this paper, they design a novel computer architecture the accelerator based on optical network-on-chip (ONoC) to further speed up the matching between citizens offer and demand in sharing economy. Our ONoC based accelerator is able to quickly calculate the optimal service-demand matching by processing computation tasks on parallel cores, i.e., task-core mapping. In addition, to improve the accelerator reliability, the assorted task-core mapping algorithm is also designed. The extensive simulation results based on real trace file demonstrate the effectiveness of our system and algorithm.

Limitations:

- File sharing is not provisioned. It only deals with storing and processing big data
- Efficiency of file read and write decreases as the number of nodes increases

[6] The Integration of Shared Storages with the CephFS and Rados Gateway for Big Data Accessing

In this system, the data is stored on Hadoop Distributed File System (HDFS), and the data stored in memory virtual distributed store system that mentioned as Alluxio automatically. Then, the data would be processed through Hadoop Map Reduce method and the output would be inserted into Hadoop Distributed File System and Alluxio environment. The first experiment is to use S3 as application program interface that will connect to RADOS

Gateway stored data into Object Storage Daemon (OSD). If there is any problem, system will give warning or error responds to users automatically.

Limitations:

- No Reliability

- File sharing is not provisioned. It only deals with storing and processing big data.

[7] SLoG: Large-Scale Logging Middleware for HPC and Big Data Convergence

Cloud developers traditionally rely on purpose specific services to provide the storage model they need for an application. In contrast, HPC developers have a much more limited choice, typically restricted to a centralized parallel file system for persistent storage. Unfortunately, these systems often offer low performance when subject to highly concurrent, conflicting I/O patterns. This makes difficult the implementation of inherently concurrent data structures such as distributed shared logs. Yet, this data structure is key to applications such as computational steering, data collection from physical sensor grids, or discrete event generators. In this paper this issue is tackled. They present SLoG, shared log middleware providing a shared log abstraction over a parallel file system, designed to circumvent the aforementioned limitations. They evaluate SLoG's design on up to 100,000 cores of the Theta supercomputer: the results show high append velocity at scale while also providing substantial benefits for other persistent backend storage systems.

Limitations:

- Doesn't provide auto-scaling of meta data
- File sharing is not provisioned. It only deals with storing and processing big data

[8] Distributed File System for NDN: an IoT Application

Named Data Networking (NDN) has been introduced as a Future Internet Architecture. Adopting this new networking architecture implies a change in the way today applications are developed. NDN brings a transition from a host centric communication to an information centric communication, where data are the masterpiece. As Internet of Things (IoT) is a source of tremendous quantity of data shared and processed on the network, a new approach is needed to enable a smooth and optimized data sharing among the IoT devices introduces an approach of a network optimized, content oriented, and fully distributed file system that enables IoT devices and processing hosts to share both the raw data and the generated results.

Limitations:

- Access time for the files is huge

- File sharing is not provisioned. It only deals with storing and processing big data

[9] Efficient File-Share Reconstruction Scheme for Device Addition/Removal in Personal Area Network

In this paper, they propose a combinatorial-based file sharing scheme for secure and efficient file management in personal area networks (PAN) on wearable devices. Existing schemes do not account for efficiency problems owing to the addition and removal of devices specifically, when devices are added or removed, the distributively stored file shares should be newly generated. Thus, they propose a file share reconstruction scheme to reduce energy consumption when devices join and leave a PAN. In this scheme, the file shares stored in the devices of the PAN are reformed with minimum operation costs.

Limitations:

- File sharing is not provisioned. It only deals with storing and processing big data
- Efficiency of file read and write decreases as the number of nodes increase.

[10] UNS: A Portable, Mobile, and Exchangeable Namespace for Supporting Fetch-From Any where Big Data Eco-Systems

They design the proposed UNS to support big data computing on multiple geolocation-based shared storage systems. They describe the innovative concepts of the proposed UNS's architecture and demonstrate UNS's early performance results from various test cases. Finally, they summarize the benefits and advantages of using UNS in supporting remote data sharing for extreme scale computing and big data computing. Furthermore, they present the future design and developmental direction of UNS

Limitations:

- No Reliability
- File sharing is not provisioned. It only deals with storing and processing big data

[11] Enabling User Driven Big Data Application on Remote Computing Resources

Ad-hoc analysis routines can be described and preserved in a format that can be shared and re-used. Remote resources can also be described and implemented through configuration files to automatically bridge the application with remote resources and

facilitate migration with different resources in the future. Consequently, analysis tasks can be preserved through the configuration file for reproducibility. Here they propose application framework and its preliminary implementations. they demonstrated usage of this framework with a practical use case of aggregating and analyzing live tweets.

Limitations:

- Doesn't provide auto-scaling of meta data
- File sharing is not provisioned. It only deals with storing and processing big data

[12] A Low-Overhead Integrity Verification for Big Data Transfers

Fast Integrity Verification (FIVER) algorithm which overlaps checksum computation and data transfer operations of files to minimize the cost of integrity verification. Extensive experiments show that FIVER is able to bring down the cost from 60% by the state-of-the-art solutions to below 10% by concurrently executing transfer and checksum operations and enabling file I/O share between them. They also implemented FIVER-Hybrid to mimic disk access patterns of sequential integrity verification approach to capture possible data corruption that may occur during file write operations which FIVER may miss. Results show that FIVER-Hybrid is able to reduce execution time by 20% compared to sequential approach without compromising the reliability of integrity verification.

Limitations:

- Access time for the files is huge
- File sharing is not provisioned. It only deals with storing and processing big data

[13] An Open Sharing Pattern Design of Massive Power Big Data

The open sharing pattern of power big data were designed in detail from the following three aspects: the way of metadata management, the mode of user access control for data open sharing, and the route of sharing service gateway generation, and then, this article gave the complete process which describes the steps of an open sharing service building, and all of these can help to achieve open sharing of massive power big data. Finally this article analyzed the effectiveness of open sharing for power big data in data mining and

data transaction, summarized the open sharing pattern of massive power big data, and gave the prospects for the next step of data open sharing work.

Limitations:

- File sharing is not provisioned. It only deals with storing and processing big data
- Efficiency of file read and write decreases as the number of nodes increases

[14] Performance Factor Analysis and Scope of Optimization for Big Data Processing on Cluster

Performance-oriented technical analysis covering all relevant aspects is presented in the context of Terascale Big data processing on TeraFLOPS cluster PARAM-Kanchenjunga, with identification of major factors influencing the performance or sources of these overheads related to computation, communication or IPC, memory, I/O contention, scheduling, load imbalance, synchronization, latency and network jitter; by determining their impact. As existing approaches found insufficient, to achieve possible speedup advance methods with a variety of alternatives as RDMA enabled libraries, PFS, MPI-Integrated extensions, loop tiling, hybrid parallelization are provided to consider for optimization purposes. This paper will assist to prepare performance aware design of experiments and performance modelling.

Limitations:

- No Reliability
- File sharing is not provisioned. It only deals with storing and processing big data

[15] Performance Evaluation of Big Data Processing Strategies for Neuroimaging

This model acknowledges the fact that page caching provided by the Linux kernel is critical to the performance of Big Data applications. Results show that in memory computing alone speeds-up executions by a factor of up to 1.6, whereas when combined with data locality, this factor reaches Lazy evaluation strategies were found to increase the likelihood of cache hits, further improving processing time. Such important speed-up values are likely to be observed on typical image processing operations performed on images of size larger than 75GB. A ballpark speculation from our model showed that in-memory computing alone will not speed-up current functional MRI analyses unless

coupled with data locality and processing around 280 subjects concurrently. Furthermore, they observe that emulating in-memory computing using in-memory file systems (tmpfs) does not reach the performance of an in-memory engine, presumably due to swapping to disk and the lack of data cleanup. they conclude that Big Data processing strategies are worth developing for neuroimaging applications.

Limitations:

- Doesn't provide auto-scaling of meta data
- File sharing is not provisioned. It only deals with storing and processing big data

[16] Performance Analysis of Big Data Frameworks on Virtualized Clusters

Research on Big Data applications has become increasingly important for institutions and researchers worldwide. This trend is triggered by the increasingly use of systems and devices that leads to generate massive of electronic data each day. The implementation of conventional algorithms has been considered to be less efficient on managing and processing large datasets. In Big Data computation, Hadoop and Apache Spark are two open source frameworks that are commonly used and run on physical clusters. Since running these frameworks on a physical cluster costs more energy and rigid in management, in this research the performance on virtualized clusters was evaluated. Virtualization technology offers flexibility on managing cluster by sharing the resources to multiple instances. Our experiments show that in general Apache Spark is about 2-9 times better in execution time and throughput compared with Hadoop running on a virtualized environment

Limitations:

- Access time for the files is huge
- File sharing is not provisioned. It only deals with storing and processing big data

The Hadoop Distributed File System

HDFS is a file system that is designed for use for MapReduce jobs that read input in large chunks of input, process it, and write potentially large chunks of output. HDFS does not handle random access particularly well. For reliability, file data is simply mirrored to multiple storage nodes. This is referred to as replication in the Hadoop community. As

long as at least one replica of a data chunk is available, the consumer of that data will not know of storage server failures.

HDFS services are provided by two processes:

- NameNode handles management of the file system metadata, and provides management and control services.
- DataNode provides block storage and retrieval services. There will be one NameNode process in an HDFS file system, and this is a single point of failure. Hadoop Core provides recovery and automatic backup of the NameNode, but no hot failover services. There will be multiple DataNode processes within the cluster, with typically one DataNode process per storage node in a cluster.

A Kavitha et.al (2013): This white paper explains us Big Data in simple language. What do you mean by Big Data? What are the traits of big data? It gives brief up of MapReduce model and introductorily oracle database. They mentioned case of Patient health information system on cloud. The real tie application of Big Data can also be in patient health information (PHR). They mentioned three aspects of the system. This application using finger print or iris pattern or face pattern of patient It capable of storing image king of data and able to processing it. Traditional enterprise data includes the entire PHR right from his/her birth with the details of the doctors and prescription and all records. Social data can be used online consultation and medicine purchase. Albert Yu et.al (2012): We study the problem of assigning subscribers to brokers in a wide-area content-based publish/subscribe system. A good assignment should consider both subscriber interests in the event space and subscriber locations in the network space, and balance multiple performance criteria including bandwidth, delay, and load balance. The resulting optimization problem is NP-complete, so systems have turned to heuristics and/or simpler algorithms that ignore some performance criteria. Evaluating these approaches has been challenging because optimal solutions remain elusive for realistic problem sizes.

Arnab Nandiet.al (2012): Computing interesting measures for data cubes and subsequent mining of interesting cube groups over massive data sets are critical for many important analyses done in the real world. Previous studies have focused on algebraic measures such as SUM that are amenable to parallel computation and can easily benefit from the recent advancement of parallel computing infrastructure such as MapReduce. In this paper, we detail real-world challenges in cube materialization and mining tasks on

web-scale data sets. Specifically, we identify an important subset of holistic measures and introduce MR-Cube, a MapReduce-based framework for efficient cube computation and identification of interesting cube groups on holistic measures. We provide extensive experimental analyses over both real and synthetic data.

Bhandarkar, M. (2010, April): Summary form of only given: Apache Hadoop has become the platform of choice for developing large-scale data-intensive applications. In this tutorial, we will discuss design philosophy of Hadoop, describe how to design and develop Hadoop applications and higher-level application frameworks to crunch several terabytes of data, using anywhere from four to 4,000 computers. We will discuss solutions to common problems encountered in maximizing Hadoop application performance. We will also describe several frameworks and utilities developed using Hadoop that increase programmer-productivity and application- performance.

Borthakur et.al (2011): Apache Hadoop platform is the first user-facing application of Facebook which is built on Apache HBase is a database-like layer built on Hadoop designed to support billions of messages per day. This paper describes the reasons why Facebook chose Hadoop and HBase over other systems such as Apache Cassandra and Voldemort and discusses the application's requirements for consistency, availability, partition tolerance, data model and scalability.

Das, S. et.al (2010, June): Many modern enterprises are collecting data at the most detailed level possible, creating data repositories ranging from terabytes to petabytes in size. The ability to apply sophisticated statistical analysis methods to this data is becoming essential for marketplace competitiveness. This need to perform deep analysis over huge data repositories poses a significant challenge to existing statistical software and data management systems. On the one hand, statistical software provides rich functionality for data analysis and modeling, but can handle only limited amounts of data; e.g., popular packages like R and SPSS operate entirely in main memory. On the other hand, data management systems - such as MapReduce-based systems - can scale to petabytes of data, but provide insufficient analytical functionality.

Destercke et.al (2013): There are many available methods to integrate information source reliability in an uncertainty representation, but there are only a few works focusing on the problem of evaluating this reliability. However, data reliability and confidence are essential components of a data warehousing system, as they influence subsequent retrieval and analysis. In this paper, we propose a generic method to assess data reliability

from a set of criteria using the theory of belief functions. Customizable criteria and insightful decisions are provided. The chosen illustrative example comes from real-world data issued from the Sym'Previus predictive microbiology oriented data warehouse, - Sym'Previus is a software, a tool to support decisions making.

DR. D.J. PATIL (2012): This book is helpful for those who have special interest about data science. The formal definition of data science is given that “A data driven organization acquires processes, and leverages data in a timely fashion to create efficiencies, iterate on and develop new products and navigate the competitive landscape.” He has his opinions on business opportunities in big data are important to listen. Big data gives 360 degree angle to see our customer expectation from us. In industry data scientist jobs are acquiring positions in exponential.

Dr. Milind Bhandarkar (2013): This Article is works in three phases of technological changes. Author looks in detailing description of big data. He defined big data in detail. He focused on apache he pointed out efforts taken by green plum in SQL- based OLAP platforms such as green plum, with Hadoop. He discussed in Hadoop adoption and use cases, how IT key players such as TCS, Social Media site companies getting benefits up to 25 % increase in ROI. 3V's of big data playing prime focusing in new era of technology. In next few years one should expect best practices for data governance and associated. “Indeed, we are witnessing the third revolution, following the industrial revolution, and internet revolution.”

Elena Baralis et.al (2013): A promising approach to Bayesian classification is based on exploiting frequent patterns, i.e., patterns that frequently occur in the training data set, to estimate the Bayesian probability. Pattern-based Bayesian classification focuses on building and evaluating reliable probability approximations by exploiting a subset of frequent patterns tailored to a given test case. This paper proposes a novel and effective approach to estimate the Bayesian probability.

Eltabakhet.al (2011): Hadoop has become an attractive platform for large-scale data analytics. To overcome bottleneck of Hadoop, author introduce CoHadoop, a lightweight extension of Hadoop that allows applications to control where data are stored. Instead, applications give hints to CoHadoop that some set of files are related and may be processed jointly; CoHadoop then tries to co-locate these files for improved efficiency. Author approach is designed such that the strong fault tolerance properties of Hadoop are retained. Colocation can be used to improve the efficiency of many operations, including

indexing, grouping, aggregation, columnar storage, joins, and sessionization. Author has been conducted a detailed study of joins and sessionization in the context of log processing.

Francois Goasdoue & M-CRousset (2013): The current trend for building an ontology-based data management system (DMS) is to. In this paper, Author extend the existing definitions of modules and we introduce novel properties of robustness that provide means for checking easily that a robust module-based DMS evolves safely w.r.t. both the schema and the data of the reference DMS. We carry out our investigations in the setting of description logics which underlie modern ontology languages, like RDFS, OWL, and OWL2 from W3C. Gautam shroff et.al (2013): The objective with this article is the fusion of social and business intelligence is defining the next generation of business analytics applications a new Artificial Intelligence (AI) driven information management architecture that based upon big data technologies and social media datasets. This article is mostly focused upon supply chain management concept .To survive in market everybody deals with their own techniques. In the world of CRM leads management can be important. Competitors' analysis plays important part in business. Social intelligence is becoming increasingly important for enterprise business intelligence. This intelligence can be get through opinion analysis. H. Altay Guveniret.al (2012) :In recent years, the problem of learning a real-valued function that induces a ranking over an instance space has gained importance in machine learning literature. Here, we propose a supervised algorithm that learns a ranking function, called ranking instances by maximizing the area under the ROC curve (RIMARC).

ROC curve (AUC) is a widely accepted performance measure for evaluating the quality of ranking, the algorithm aims to maximize the AUC value directly. For a single categorical feature, we show the necessary and sufficient condition that any ranking function must satisfy to achieve the maximum AUC. We also sketch a method to discretize a continuous feature in a way to reach the maximum AUC as well. RIMARC uses a heuristic to extend this maximization to all features of a data set.

Huanget.al (2012): Caching valid regions of spatial queries at mobile clients is effective in reducing the number of queries submitted by mobile clients and query load on the server. However, mobile clients suffer from longer waiting time for the server to compute valid regions.

Husain, M. F.et.al (2009): Handling huge amount of scalable data is a matter of concern for a long time. Same is true for semantic web data. Current semantic web frameworks lack this ability. In this paper, we describe a framework that we built using Hadoop to store and retrieve large number of RDF triples. We describe our schema to store RDF data in Hadoop Distribute File System. We also present our algorithms to answer a SPARQL query. We make use of Hadoop's MapReduce framework to actually answer the queries. Our results reveal that we can store huge amount of semantic web data in Hadoop clusters built mostly by cheap commodity class hardware and still can answer queries fast enough. Ibrahim et.al (2009): MapReduce is emerging as an important programming model for large scale parallel application. Meanwhile, Hadoop is an open source implementation of MapReduce enjoying wide popularity for developing data intensive application in the cloud. As, in the cloud, the computing unit is virtual machine (VM) based; it is feasible to demonstrate the applicability of MapReduce on virtualized data center. Although the potential for poor performance and heavy load no doubt exists, virtual machines can instead be used to fully utilize the system resources, ease the management of such systems, improve the reliability, and save the power. Kashif Javed (2012): Data and knowledge management systems employ feature selection algorithms for removing irrelevant, redundant, and noisy information from the data. There are two well-known approaches to feature selection, feature ranking (FR) and feature subset selection (FSS). CDFE uses a fill trapper approach to select a final subset. For data sets having hundreds of thousands of features, feature selection with FR algorithms is simple and computationally efficient but redundant information may not be removed. On the other hand, FSS algorithms analyze the data for redundancies but may become computationally impractical on high-dimensional data sets.

Leverich & Kozyrakis (2010): Distributed processing frameworks, such as Yahoo!'s Hadoop and Google's MapReduce, have been successful at harnessing expansive datacenter resources for large-scale data analysis. However, their effect on datacenter energy efficiency has not been scrutinized. Moreover, the filesystem component of these frameworks effectively precludes scale-down of clusters deploying these frameworks (i.e. operating at reduced capacity). This paper presents our early work on modifying Hadoop to allow scale-down of operational clusters. We find that running Hadoop clusters in fractional configurations can save between 9% and 50% of energy consumption, and that there is a tradeoff between performance energy consumption. We also outline further research into the energy-efficiency of these frameworks.

Liuet.al (2009, August): Hadoop framework has been widely used in various clusters to build large scale, high performance systems. However, Hadoop distributed file system (HDFS) is designed to manage large files and suffers performance penalty while managing a large amount of small files. As a consequence, many web applications, like WebGIS, may not take benefits from Hadoop. In this paper, we propose an approach to optimize I/O performance of small files on HDFS. The basic idea is to combine small files into large ones to reduce the file number and build index for each file.

Luca Cagliero & Alessandro Fiori (2013): The increasing availability of user-generated content coming from online communities allows the analysis of common user behaviors and trends in social network usage. This paper presents the TweM Tweet Miner framework that entails the discovery of hidden and high level correlations, in the form of generalized association rules, among the content and the contextual features of posts published on Twitter i.e., the tweets. Experiments, performed on both real Twitter posts and synthetic datasets, show the effectiveness and the efficiency of the proposed TweM framework in supporting knowledge discovery from Twitter user-generated content.

Lushan Hanet.al (2013): Point wise mutual information (PMI) is a widely used word similarity measure, but it lacks a clear explanation of how it works. We explore how PMI differs from distributional similarity, and we introduce a novel metric, $(\text{PMI})_{\max}$, that augments PMI with information about a word's number of senses. The coefficients of $(\text{PMI})_{\max}$ are determined empirically by maximizing a utility function based on the performance of automatic thesaurus generation. We show that it outperforms traditional PMI in the application of automatic thesaurus generation and in two word similarity benchmark tasks: human similarity ratings and TOEFL synonym questions. $(\text{PMI})_{\max}$ achieves a correlation coefficient comparable to the best knowledge-based approaches on the Miller-Charles similarity rating data set.

Ref. no.	Author	Technique	Results	Limitations
[1]	Ke Ma	Distributed Storage System, NameNod	This paper points out the deficiencies in masses mall file access problems.	File sharing is not provisioned. It only deals with storing and processing big data.

[2]	Yifeng Luo Jia Shi Shuigeng Zhou	Just-in-time block prefetching.	Big data applications' memory cache resource demands while not to harm the execution efficiency of upper-level running jobs.	It only deals with storing and processing big data. File sharing is not provisioned.
[3]	Dan Huang Dezhi Han Jun Wang Jiangling Yin	Optimize parallel data read. Imbalance of parallel writes.	The data requests as a bipartite matching problem and propose novel matching based algorithms for optimizing parallel data reads	This method doesn't provide auto-scaling of meta data.
[4]	Hadeel Alghamdi Farhana Zulkernine	Cloud-based Analytics-as-a- Service.	This paper WINGS to support a distributed hybrid big data storage framework on the back end.	Access time for the files is huge,It only deals with storing and processing big data
[5]	LeiGuo, ZhaolongNin g WeigangHou BinHu Pengxing Guo.	Optical network-on-chip (ONoC)	Designed a novel heuristic scheme ACGA to determine the most appropriate accelerator architecture that had the optimal task-core mapping solution for the service- demand matching.	Efficiency of file read and write decreases as the number of nodes increases.
[6]	Jia-Yow Weng Chao- Tung Yang Chih- Hung Chang	Object Storage Daemon (OSD). Ceph File System	Ceph is a high-performance, scalable storage, safe and stable management platform	File sharing is not provisioned. It only deals with storing and

		(CephFS).	management mode with simple and low cost.	processing big data.
[7]	PierreMatri, PhilipCarns, RobertRoss	High-performance computing (HPC) platforms.	SLoG middleware, proposing a distributed shared log abstraction over a parallel file system.	Doesn't provide auto-scaling of meta data. File sharing is not provisioned.
[8]	Youssef ATIK Charif MAHMOUD I	Named Data Networking (NDN)	Presented a fully distributed approach of distributed file system and computation distribution based on NDN architecture applied to IoT.	Access time for the files is huge,File sharing is not provisioned..
[9]	Jung-Eun Park Young-Hoon Park	File share reconstruction scheme	Investigated the issues that arise owing to changes in the total number of participating devices n in the (k,n) -file sharing scheme.	Efficiency of file read and write decreases as the number of nodes increases.
[10]	Hsing-bung (HB) Chen Qiang Guan Song Fu	Universal Namespace (UNS).	UNS provides portability and mobility of namespace by bridging together local and global files systems.	File sharing is not provisioned. It only deals with storing and processing big data

[11]	WeijiaXu RuizhuHuan g Yige Wang	Cyber infrastructure. Reconfigurable web service.	Provides a the framework to enhance the accessibility of large cyber infrastructure to users from diverse domain fields.	Doesn't provide auto- scaling of meta data, File sharing is not provisioned.
[12]	EnginArslan Ahmed Alhussen	Fast Integrity VERification (FIVER) algorithm	FIVER and FIVER-Hybrid algorithms to minimize the cost of integrity verification.	Access time for the files is huge,It only deals with storing and processing big data.
[13]	Sen Pan LipengZhu Junfeng Qiao	data open sharing; power big data	this paper proposes an open sharing model suitable for massive power big data, container technology and multi- tenant technology	Efficiency of file read and write decreases as the number of nodes increases.
[14]	HanumanGo da ra M.C. Govil E.S. Pilli	Terascale Big data processing	They provide dynamic nature and adapt to varying working conditions	Not very efficient, may have chance to provide false results
[15]	Valerie Hayot- Sasson Shawn T Brown Tristan Glatard	Neuroimaging	Big Data performance optimization strategies help improve performance of typical neuroimaging applications.	Doesn't provide auto- scaling of meta data It only deals with storing and processing big data

From the surveillance of all the 16 papers, we can summarize that,

1. File sharing is not provisioned. It only deals with storing and processing big data.

2. Efficiency of file read and writes decreases as the number of nodes increases.
3. Doesn't provide auto-scaling of Meta data.
4. Some of the methods provided aren't reliable.
5. Access time for the files is huge.

Chapter 3

ANALYSIS

3.1 Problem Identification

Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. Analysts in the field of engineering look at requirements, structures, mechanisms, and systems dimensions. Analysis is an exploratory activity.

The Analysis Phase is where the project lifecycle begins. The Analysis Phase is where you break down the deliverables in the high-level Project Charter into the more detailed business requirements. The Analysis Phase is also the part of the project where you identify the overall direction that the project will take through the creation of the project strategy documents.

Gathering requirements is the main attraction of the Analysis Phase. The process of gathering requirements is usually more than simply asking the users what they need and writing their answers down. Depending on the complexity of the application, the process for gathering requirements has a clearly defined process of its own. This process consists of a group of repeatable processes that utilize certain techniques to capture, document, communicate, and manage requirements.

File sharing means sending and receiving of different types of file (audio, video, and picture) within the same network or different network. File sharing is done using techniques like file storage, distribution and transmission.

File sharing is the act of circulating or giving access to carefully put away data, for example, PC programs, media (sound, pictures and video), reports, or electronic books. It might be actualized through an assortment of ways. Earlier file sharing applications have a drawback, once the file sharing link is discoverable, the file can be accessed by unauthorized user.

The Hadoop system itself is for the most part written in the Java programming dialect, with some local code in C and charge line utilities composed as shell contents. Despite the fact that MapReduce Java code is normal, any programming dialect can be

utilized with "Hadoop Streaming" to actualize the "guide" and "lessen" parts of the client's program.

Hadoop is an open-source programming structure utilized for circulated capacity and handling of dataset of enormous information utilizing the MapReduce programming model. Managing big data is being a big problem to many organizations. Due to this reason, file sharing faces many problems. Every day 2.5 Quintillion bytes are created.

Hadoop can easily handle by Hadoop File Distributed System (HDFS). This proposed system has an advanced security i.e. OTP service. The sender will send the required file to receiver. While opening the file, the receiver has to enter an OTP number, which he will receive from the sender. This provides the system more security as the OTP is accessible to authorized user.

3.1.1 Volume, Velocity, Variety

The “3V’s”, how Doug Laney calls them in his article 3-D Data Management: Controlling Data Volume, Velocity and Variety, published in 2001, represent key elements that are considered vital regarding the characteristics of Big Data systems.

The first characteristic of Big Data, which is “Volume”, refers to the quantity of data that is being manipulated and analyzed in order to obtain the desired results. It represents a challenge because in order to manipulate and analyze a big volume of data requires a lot of resources that will eventually materialize in displaying the requested results. For example a computer system is limited by current technology regarding the speed of processing operations. The size of the data that is being processed can be unlimited, but the speed of processing operations is constant. To achieve higher processing speeds more computer power is needed and so, the infrastructure must be developed, but at higher costs. By trying to compress huge volumes of data and then analyze it, is a tedious process which will ultimately prove more ineffective. To compress data it takes time, almost the same amount of time to decompress it in order to analyze it so it can be displayed, by doing this, displaying the results will be highly delayed. One of the methods of mining through large amount of data is with OLAP solutions (Online Analytical Processing) Data warehouse -> OLAP). An OLAP solution consists of tools and multidimensional databases that allow users to easily navigate and extract data from different points of view. Therefore, it identifies relations between elements in the

database so it can be reached in a more intuitive way. An example of how OLAP systems are rearranging the data imported from a data warehouse is below.

For obtaining results various OLAP tools are used in order for the data to be mined and analyzed “Velocity” is all about the speed that data travels from point A, which can be an end user interface or a server, to point B, which can have the same characteristics as point A is described. This is a key issue as well due to high requests that end users have for streamed data over numerous devices (laptops, mobile phones, tablets etc.). For companies this is a challenge that most of them can’t keep up to. Usually data transfer is done at less than the capacity of the systems. Transfer rates are limited but requests are unlimited, so streaming data in real-time or close to real-time are a big challenge. The only solution at this point is to shrink the data that is being sent. A good example is Twitter. Interaction on Twitter consists of text, which can be easily compressed at high rates. But, as in the case of “Volume” challenge, this operation is still time-consuming and there will still be delay in sending-receiving data.

The only solution to this right now is to invest in infrastructure. “Variety” is the third characteristic of Big Data. It represents the type of data that is stored, analyzed and used. The type of data stored and analyzed varies and it can consist of location coordinates, video files, data sent from browsers, simulations etc. The challenge is how to sort all this data so it can be “readable” by all users that access it and does not create ambiguous results. The mechanics of sorting has two key variables at the beginning: the system that transmits data and the system that receives it and interprets it so that can be later displayed.

3.1.2 Data privacy and Data security.

This has many implications and it concerns individuals and companies as well. Individuals have the right, according to International Telecommunications Union, to control the information that may be disclosed regarding them. Information posted by users on their online profiles is likely to be used in creating a “users profile” so that can be further used by companies to develop their marketing strategies and to extend their services. Individual’s privacy is still a delicate problem that can only be solved with drastic solutions. Allowing persons to choose whether they post or not information about them is a more secure way to achieve privacy, but will also cause software to “malfunction”. For example in a social network, if a person is allowed to choose whether

he/she wants to complete the fields regarding personal information and, in the same time, allow them to choose if the social network can store information about their IP address, location etc., this could be a possible threat to everyone else that is using the same social network. For companies the privacy issue is more related to the sensitive data that they work with. Whether is financial data, clients list, perspective projects, all represents valuable data that may or may not be disclosed. Companies have multiple choices regarding where to store their information. They can either store it on cloud systems.

3.2 Objectives

The proposed system has an advanced security i.e. OTP service. The sender will send the required file to receiver. While opening the file, the receiver must enter an OTP number, which he will receive from the sender. This provides the system more security as the OTP is accessible to authorized user. The data owner will be capable of revoking the access to the users at any point of time. The system will also take care that the shared user will have no rights to further share the data with others. Also, the system is implemented for all the types of data, i.e, text, audio, video, images etc

The objectives are as follows,

- Develop a standalone Hadoop application to provide the Data I/O operations to and from server using Java
- Provide an ability for the customers to share the file across different users with different access rights
- Implement the OTP service for providing security for the shared file
- Implement the Access management component
- Ability for the customer to revoke the access at any instance of time

3.3 Methodology

The main concept behind the Hadoop file system is the MapReduce. MapReduce is a concept which was developed by Google to sort the large data which is in petabytes by forming clusters of this data. MapReduce mainly has two parts i.e.-Map and Reduce.

Map is a transformation step in which the singular records are handled in parallel. Reduce is a summarization step in which all the related records are handled in a single entity. The concept of MapReduce is that initially all the data is divided into small parts called as chunks. These chunks are then individually processed by a map task. Later these chunks are physically partitioned and sorted accordingly.

Each sorted chunk is send to a reduce task. The figure below shows the MapReduce concept in detail. Each record will be split into several parts (chunks). A map task can run on any node in the computer and there maybe multiple map tasks running simultaneously. The map task converts each record into a key/value pair.

This key value pair will be partitioned and sorted. Each partition will then have a reduce task. Each partition's keys and values will have separate reduce tasks. There may be multiple reduce task running at a given time Any developer needs to provide to the Hadoop Framework 4 items i.e. a class which will read the input record and transform it into key/value pair, a map class, a reduce class and a class which will transform the key/value pair into output record. MapReduce requires a shared file system. Shared file system does not mean any file level system but it needs a distributed system with a plug in available to the framework. When HDFS will be used as a shared file system, Hadoop has an advantage that it will recognize which node has a physical copy of the input data and will check to read the data that is on the machine.

If the user doesn't want the reduce task, the user need not specify the reduce class. The framework will partition the input and schedule the map task. If needed the output of the map task will be sorted and given to the reduce task and the final output will be given to the user. The framework has two processes that handle the management of the MapReduce, they are TaskTracker and JobTracker. The TaskTracker provides the individual execution of map and reduce tasks. The JobTracker provides job submission, monitoring and control. The Hadoop File Distributed System: Is the one which is designed for the MapReduce concept. This HDFS system uses the MapReduce concept to sort large chunks of data, sort them and provide them as large chunks of output files.

The HDFS services are provided using two main processes i.e. NameNode and DataNode. The NameNode manages the file system data by providing the management and control services. The DataNode provides data storage and retrieval services.

Algorithm

Step 1: Open the windows application

Step 2: If new user then registers else login with user id and password

Step 3: Select the files to be uploaded

Step 4: Select the file to be shared

Step 5: Enter the user to who file will be sent

Step 6: When the user has to access the shared file, the OTP will be sent to the User's mobile number and Email ID.

Step 7: The receiver will have to enter the OTP to open the file

Step 8: Repeat the steps 4 to 6 any number of time.

Step 9: Stop

For the purpose of processing the large amount of data, the big data requires exceptional technologies. The various techniques and technologies have been introduced for manipulating, analyzing and visualizing the big data. There are many solutions to handle the Big Data, but the Hadoop is one of the most widely used technologies.

3.4 System Requirement Specification

3.4.1 Software Requirement Specification

A software requirements specification (SRS) – a requirements specification for a software system – is a complete description of the behavior of a system to be developed. In addition to a description of the software functions, the SRS also contains non-functional requirements. Software requirements are a sub-field of software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software.

Purpose

The purpose of this document is to provide Software Requirement Specification for “An Efficient and Secured OTP Enabled File Sharing service over Big Data Environment”.

Scope

The software product produced is an application by name “An Efficient and Secured OTP Enabled File Sharing service over Big Data Environment”. File sharing has been an essential part of this century. Using various applications, files can be shared to large number of users. For the purpose of storage, the Hadoop Distributed File System (HDFS) can be used. HDFS is mainly used for the unstructured data analysis. The HDFS handles large size of files in a single server. Common sharing methods like removable media, servers or computer network, World Wide Web-based hyperlink documents.

In the proposed project, the files are merged using MapReduce programming model on Hadoop. This process improves the performance of Hadoop by rejecting the files which are larger than the size of Hadoop and reduces the memory size required by the Name Node. The main concept behind the Hadoop file system is the MapReduce. MapReduce is a concept which was developed by Google to sort the large data which is in petabytes by forming clusters of this data. MapReduce mainly has two parts i.e.-Map and Reduce. Map is a transformation step in which the singular records are handled in parallel. Reduce is a summarization step in which all the related records are handled in a single entity.

The problem statement here in this project is to deal with the huge measure of little size records it requires greater investment (for name hub). The issue here is to enhance the execution of Hadoop in treatment of little records to accomplish the coveted yield by framework by utilizing productive consolidation.

Table 3.1: Software Requirements

Operating System	Linux or Cent OS
Programming Language – Backend	Core Java, Advanced Java, J2EE, Map Reduce Framework, MVC Framework
Programming language - Frontend	Bootstrap Framework, HTML, CSS, JavaScript, Ajax, JQuery
Development environment	Eclipse Oxygen IDE
Application Server	Apache Tomcat v9.0
Database	SQL

3.4.2 Hardware Requirement Specification

The Hardware Requirements Specification (SRS) captures the complete Hardware requirements for the system, or a portion of the system.

Table 3.2: Hardware Requirements

Processor	Intel Core i5 or AMD FX 8 core series with clock speed of 2.4 GHz or above
RAM	2GB or above
Hard disk	40 GB or above
Input device	Keyboard or mouse or compatible pointing devices
Display	XGA (1024*768 pixels) or higher resolution monitor with 32 bit color settings
Miscellaneous	USB Interface, Power adapter, etc

3.4.3 Functional Requirements

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification.

In software engineering and systems engineering, a Functional Requirement can range from the high-level abstract statement of the sender's necessity to detailed mathematical functional requirement specifications. Functional software requirements help you to capture the intended behaviour of the system.

Following are the functional requirements for the Hadoop File Sharing System,

- To address the data security issues in the Hadoop Distributed File System (HDFS)
- To integrate the existing file write operation with OTP enabled scheme
- To integrate the existing file read operation with OTP enabled scheme

3.4.4 Non-Functional Requirements

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the numbers of simultaneous users are > 10000 . Description of non-functional requirements is just as critical as a functional requirement.

Following are the non-functional requirements for the Hadoop File Sharing System,

- Should be easier to access it from the various browsers available.
- Response time of the applications should reflect the real time observations.
- The algorithm should never fail in any of the test cases.
- There shouldn't be any security concerns on the merged data.
- Each user's activity should be separated from the other user's activities

Chapter 4

SYSTEM DESIGN

4.1 System Architecture Diagram

The below figure shows a general block diagram describing the activities performed by this project.

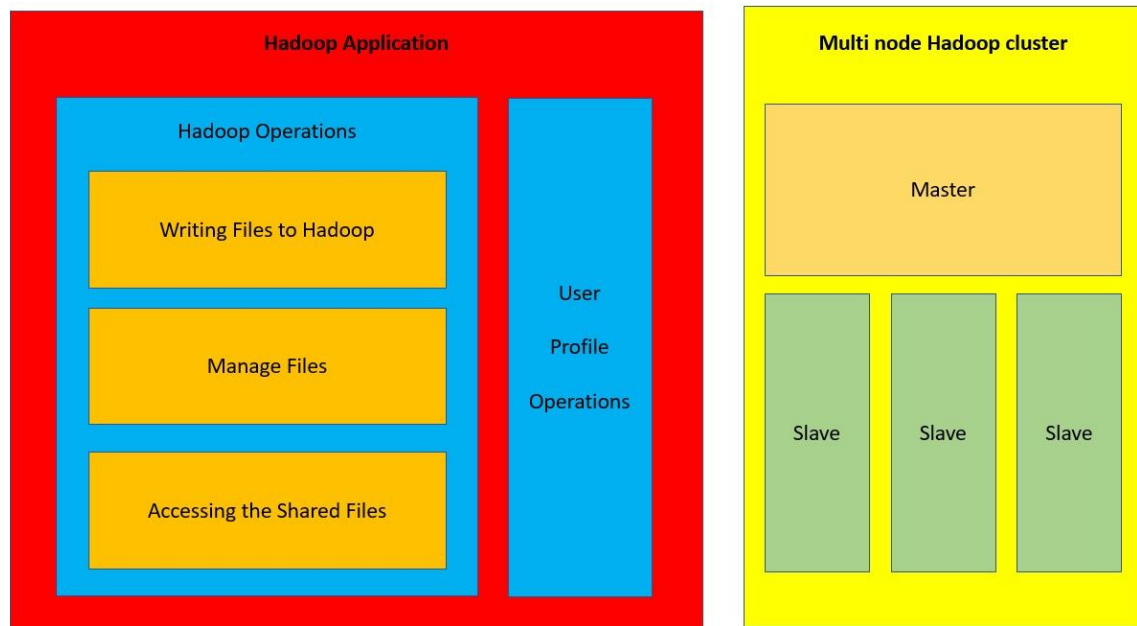


Fig 4.1 System Architecture Diagram

The major divisions of the project are:-

4.1.1 Data Access Layer

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs

4.1.2 Account Operations

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session
- Edit the existing Profile

- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

4.2 Detailed Design

4.2.1 High-level Design

The design of the system is perhaps the most critical factor affecting the quality of the software. The objective of the design phase is to produce overall design of the software. It aims to figure out the modules that should be in the system to fulfill all the system requirements in an efficient manner.

A) Module 1: Setting up multi node Hadoop cluster

Hadoop follows a master slave architecture design for data storage and distributed data processing using HDFS and MapReduce respectively. The master node for data storage is Hadoop HDFS is the NameNode and the master node for parallel processing of data using Hadoop MapReduce is the Job Tracker. The slave nodes in the Hadoop architecture are the other machines in the Hadoop cluster which store data and perform complex computations. Every slave node has a Task Tracker daemon and a DataNode that synchronizes the processes with the Job Tracker and NameNode respectively. In Hadoop architectural implementation the master or slave systems can be setup in the cloud or on-premise.

Hadoop Distributed File System (HDFS) stores the application data and file system metadata separately on dedicated servers. NameNode and DataNode are the two critical components of the Hadoop HDFS architecture. Application data is stored on servers referred to as DataNodes and file system metadata is stored on servers referred to as NameNode. HDFS replicates the file content on multiple DataNodes based on the replication factor to ensure reliability of data. The NameNode and DataNode communicate with each other using TCP based protocols.

For the Hadoop architecture to be performance efficient, HDFS must satisfy certain pre-requisites

- All the hard drives should have a high throughput.
- Good network speed to manage intermediate data transfer and block replications.

B) Module 2: User Profile Operations

This module implements the basic user profile operations on the prototype application. The user profile operations include creating a new account, logging in to the existing account, logging out, editing the profile, changing the password, and deleting the profile if not needed anymore.

This application is also deployed on the cloud server so that this can be accessed by anyone across the globe using the IP address of this cloud server. The implementation is done using the J2EE architecture and for the database needs we have used SQLITE3.

For communicating with the Database whenever the user performs any of his/her profile operations, we use the Data Access Operations (DAO) layer for communicating with the database.

C) Module 3: File Upload

In this module we provide the registered user with an HTML interface where he/she can upload the file into the Hadoop system. The user can do this using the provided browse button. The system currently supports any of the file MIME types.

The system also supports the file of any size. Upon submitting the file to the user interface, the request will be sent to the backend servlet running on Apache Tomcat which then reads the file from the client's machine on to the server in a streaming manner. This read will store the file into a temporary location on the server. The servlet will then make use of the helper classes to upload this file on to the Hadoop Distributed File System (HDFS) using the Out of the box (OOTB) commands provided by the Hadoop platform. Once the file upload to Hadoop is done, the servlet will remove this file from the temporary location on the server. The acknowledgement of the process will be sent back to the client's browser.

D) Module 4: Manage Files

In this module the registered users can manage their files on the Hadoop Distributed File System (HDFS). The user can perform the read, write, update or the delete operations on his/her files. All these operations will be made available to the client through a HTML interface. On the server, these requests will be fulfilled once again using the Out of the box (OOTB) commands provided by the Hadoop Platform. In addition to these operations, the users will also be provided with a provision to share his/her files with other registered users in the portal. This can be done by selecting the file which has to be shared and entering the Sharing component in this module.

The user will then be provided with a view listing all the registered users in the portal. The user can select any of the user from the list and grant access to that user on the selected file. The system allows the user to grant the access in two different levels namely

READ ONLY access and READ WRITE access. The READ ONLY access means the users with whom the file has been shared can only view the contents of the file but cannot perform any update or the delete operations on them. The READ WRITE access provides the users to perform read, write, and delete of the file shared with them.

4.2.2 Low-level Design

A) Module 1: Setting up multi node Hadoop cluster

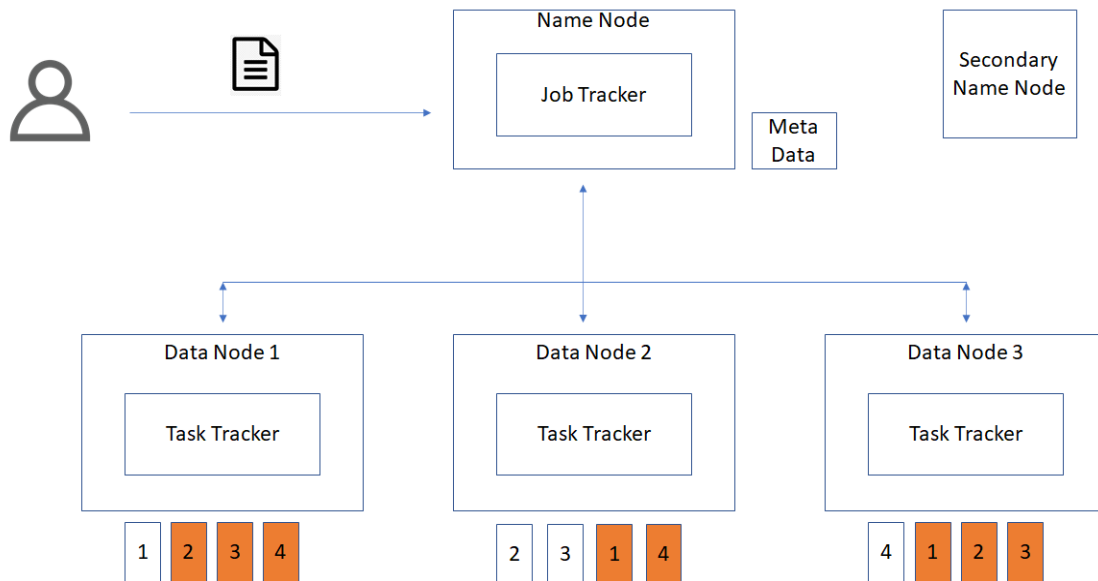


Fig 4.2 Architecture Design of Slave Node

Hadoop follows a master slave architecture design for data storage and distributed data processing using HDFS and MapReduce respectively. The master node for data storage in hadoop HDFS is the NameNode and the master node for parallel processing of data using Hadoop MapReduce is the Job Tracker. The slave nodes in the hadoop architecture are the other machines in the Hadoop cluster which store data and perform complex computations. Every slave node has a Task Tracker daemon and a DataNode. Hadoop Distributed File System (HDFS) stores the application data and file system metadata separately on dedicated servers. NameNode and DataNode are the two critical components of the Hadoop HDFS architecture. Application data is stored on servers referred to as DataNodes and file system metadata is stored on servers referred to as NameNode. HDFS replicates the file content on multiple DataNodes based on the replication factor to ensure reliability of data. The NameNode and DataNode communicate with each other using TCP based protocols. e that synchronizes the processes with the Job Tracker and NameNode respectively.

- **Namenode**

All the files and directories in the HDFS namespace are represented on the NameNode by Inodes that contain various attributes like permissions, modification timestamp, disk space quota, namespace quota and access times. NameNode maps the entire file system structure into memory. Two files fsimage and edits are used for persistence during restarts.

- Fimage file contains the Inodes and the list of blocks which define the metadata. It has a complete snapshot of the file systems metadata at any given point of time.
- The edits file contains any modifications that have been performed on the content of the fsimage file. Incremental changes like renaming or appending data to the file are stored in the edit log to ensure durability instead of creating a new fsimage snapshot everytime the namespace is being altered.

When the NameNode starts, fsimage file is loaded and then the contents of the edits file are applied to recover the latest state of the file system. The only problem with this is that over the time the edits file grows and consumes all the disk space resulting in slowing down the restart process. If the hadoop cluster has not been restarted for months together then there will be a huge downtime as the size of the edits file will be increase. This is when Secondary NameNode comes to the rescue. Secondary NameNode gets the fsimage and edits log from the primary NameNode at regular intervals and loads both the fsimage and edit logs file to the main memory by applying each operation from edits log file to fsimage. Secondary NameNode copies the new fsimage file to the primary NameNode and also will update the modified time of the fsimage file to fstime file to track when then fsimage file has been updated.

- **DataNode**

DataNode manages the state of an HDFS node and interacts with the blocks .A DataNode can perform CPU intensive jobs like semantic and language analysis, statistics and machine learning tasks, and I/O intensive jobs like clustering, data import, data export, search, decompression, and indexing. A DataNode needs lot of I/O for data processing and transfer. On startup every DataNode connects to the NameNode and performs a handshake to verify the namespace ID and the software version of the DataNode. If either of them does not match then the DataNode shuts down automatically. A DataNode verifies the block replicas in its ownership by sending a block report to the NameNode. As soon as the DataNode registers, the first block report is sent. DataNode sends heartbeat to the NameNode every

3 seconds to confirm that the DataNode is operating and the block replicas it hosts are available.

B) Module 2: User Profile Operations

This module implements the basic user profile operations on the prototype application. The user profile operations include creating a new account, logging in to the existing account, logging out, editing the profile, changing the password, and deleting the profile if not needed anymore.

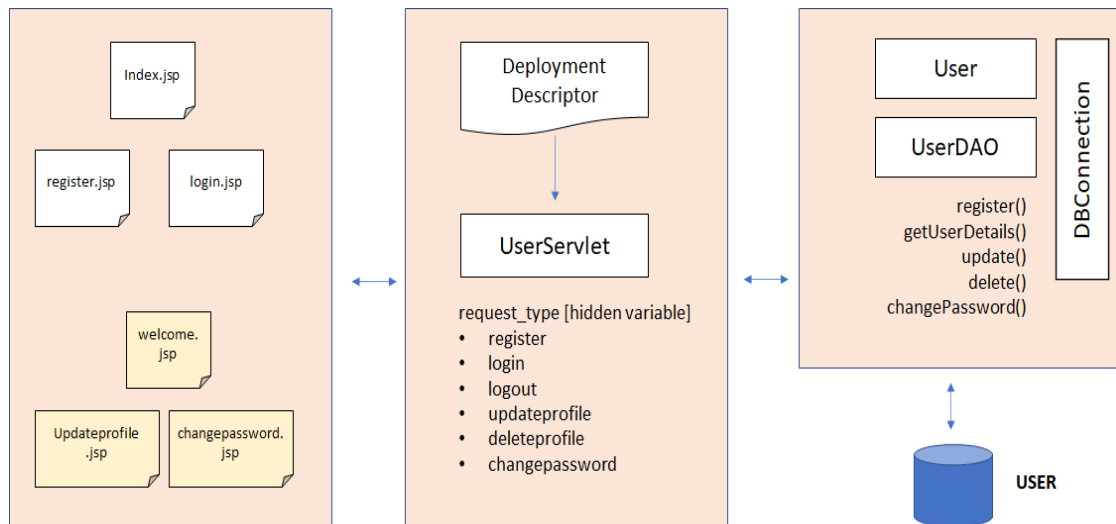


Fig 4.3 Profile Operation Module Interaction

This application is also deployed on the cloud server so that this can be accessed by anyone across the globe using the IP address of this cloud server. The implementation is done using the J2EE architecture and for the database needs we have used SQLITE3. For communicating with the Database whenever the user performs any of his/her profile operations, we use the Data Access Operations (DAO) layer for communicating with the database.

C) Module 3: File Upload

In this module we provide the registered user with an HTML interface where he/she can upload the file into the Hadoop system.

The user can do this using the provided browse button. The system currently supports any of the file MIME types.

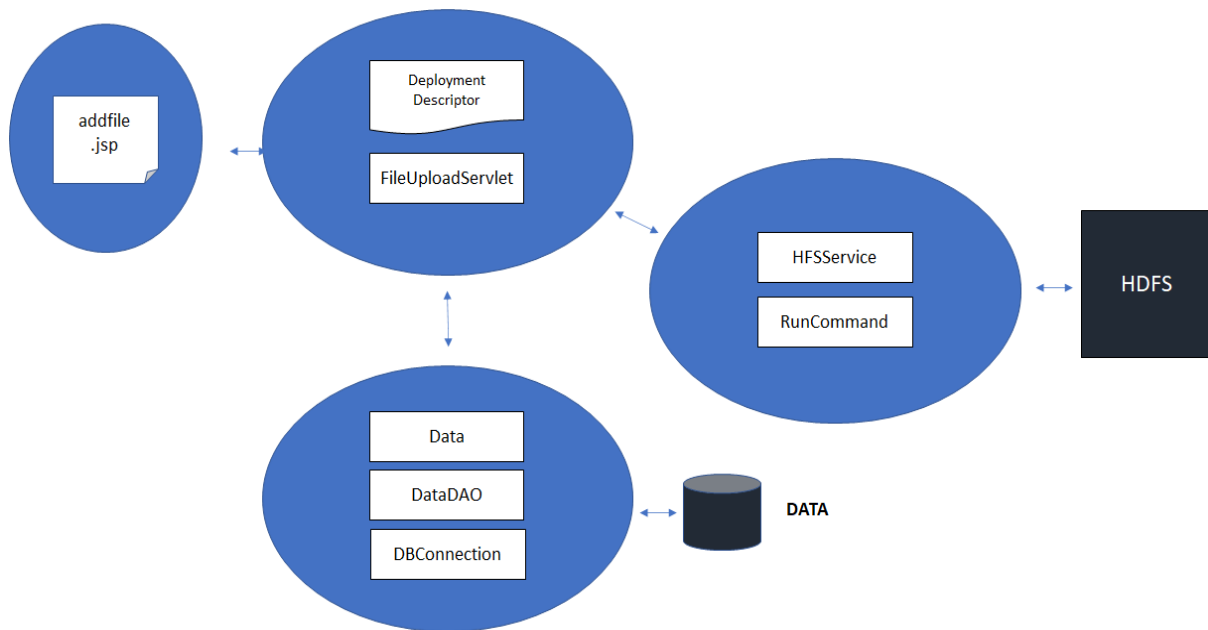


Fig 4.4 File Upload Process Description

The system also supports the file of any size. Upon submitting the file to the user interface, the request will be sent to the backend servlet running on Apache Tomcat which then reads the file from the client's machine on to the server in a streaming manner. This read will store the file into a temporary location on the server. The servlet will then make use of the helper classes to upload this file on to the Hadoop Distributed File System (HDFS) using the Out of the box (OOTB) commands provided by the Hadoop platform. Once the file upload to Hadoop is done, the servlet will remove this file from the temporary location on the server. The acknowledgement of the process will be sent back to the client's browser.

D) Module 4: Manage Files

In this module the registered users can manage their files on the Hadoop Distributed File System (HDFS). The user can perform the read, write, update or the delete operations on his/her files. All these operations will be made available to the client through a HTML interface. On the server, these requests will be fulfilled once again using the Out of the box (OOTB) commands provided by the Hadoop Platform. In addition to these operations, the users will also be provided with a provision to share his/her files with other registered users in the portal. This can be done by selecting the file which has to be shared and entering the Sharing component in this module.

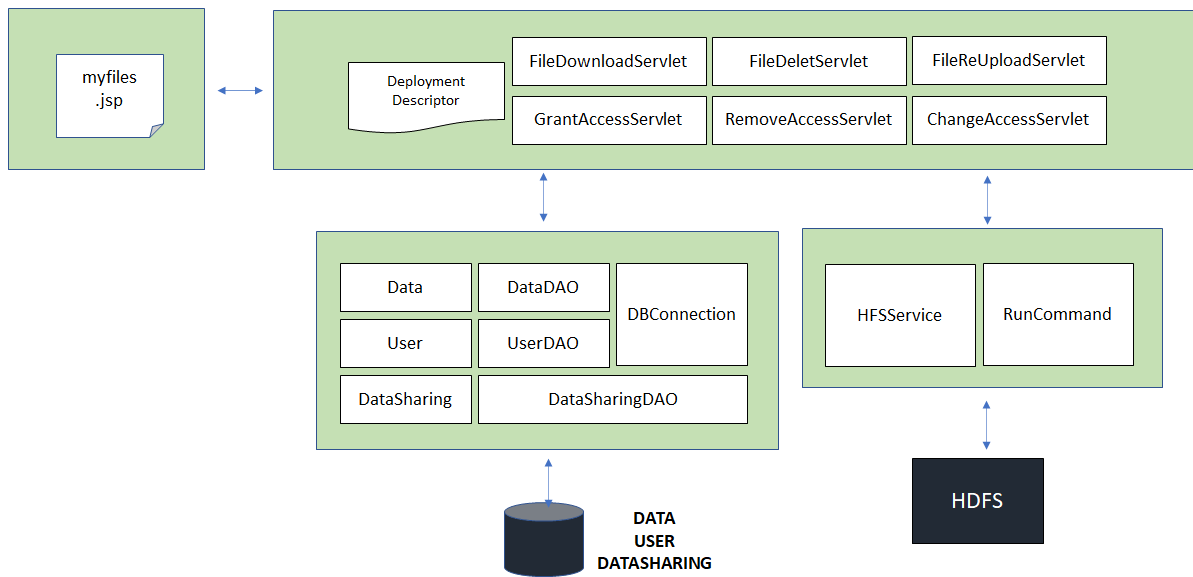


Fig 4.5 Managing File Servlet Interaction

The user will then be provided with a view listing all the registered users in the portal. The user can select any of the user from the list and grant access to that user on the selected file. The system allows the user to grant the access in two different levels namely **READ ONLY** access and **READ WRITE** access. The **READ ONLY** access means the users with whom the file has been shared can only view the contents of the file but cannot perform any update or the delete operations on them. The **READ WRITE** access provides the users to perform read, write, and delete of the file shared with them. The data owners have the full rights of revoking the access at any point of time.

E) Module 5: Shared Files

In this module, the user after logging in into the system can perform the file access operations on all the files which have been shared with him/her by other registered users. To get the access to the files the user will have to prove his/her identity again by entering the OTP. When the user requests to perform the access operations on the shared files, the system will send an OTP to his/her registered mobile number and also to his/her registered email ID.

The user will then have to enter the correct OTP before the system can grant access to him. The user will either be granted with **READ ONLY** access or the **READ WRITE** access on the files by the data owners. The system allows the users to access them in accordance with these access rules.

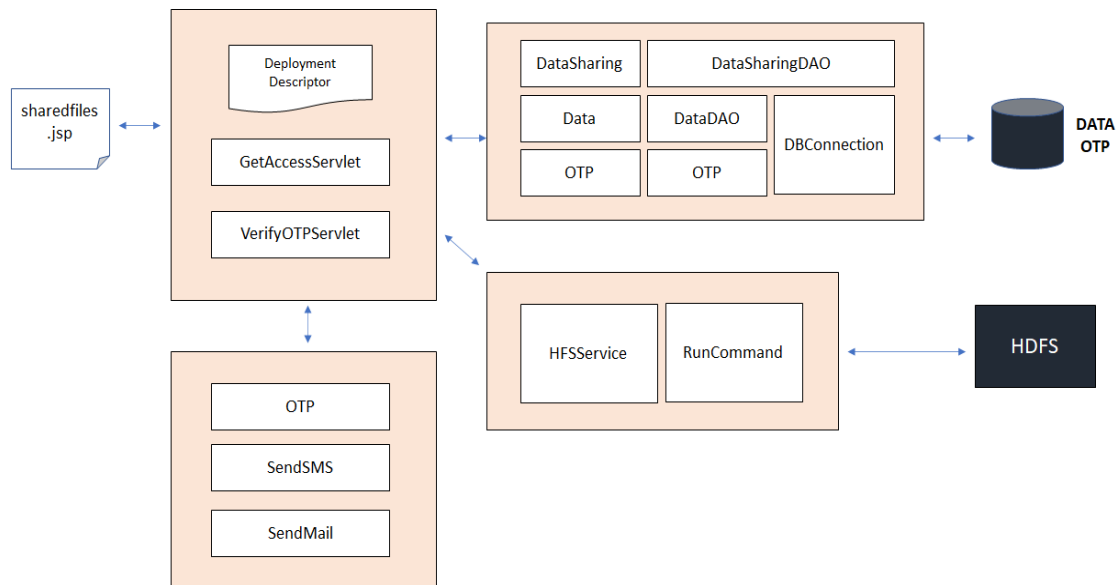


Fig 4.6 Flow of access on shared files

4.3 Data flow diagram

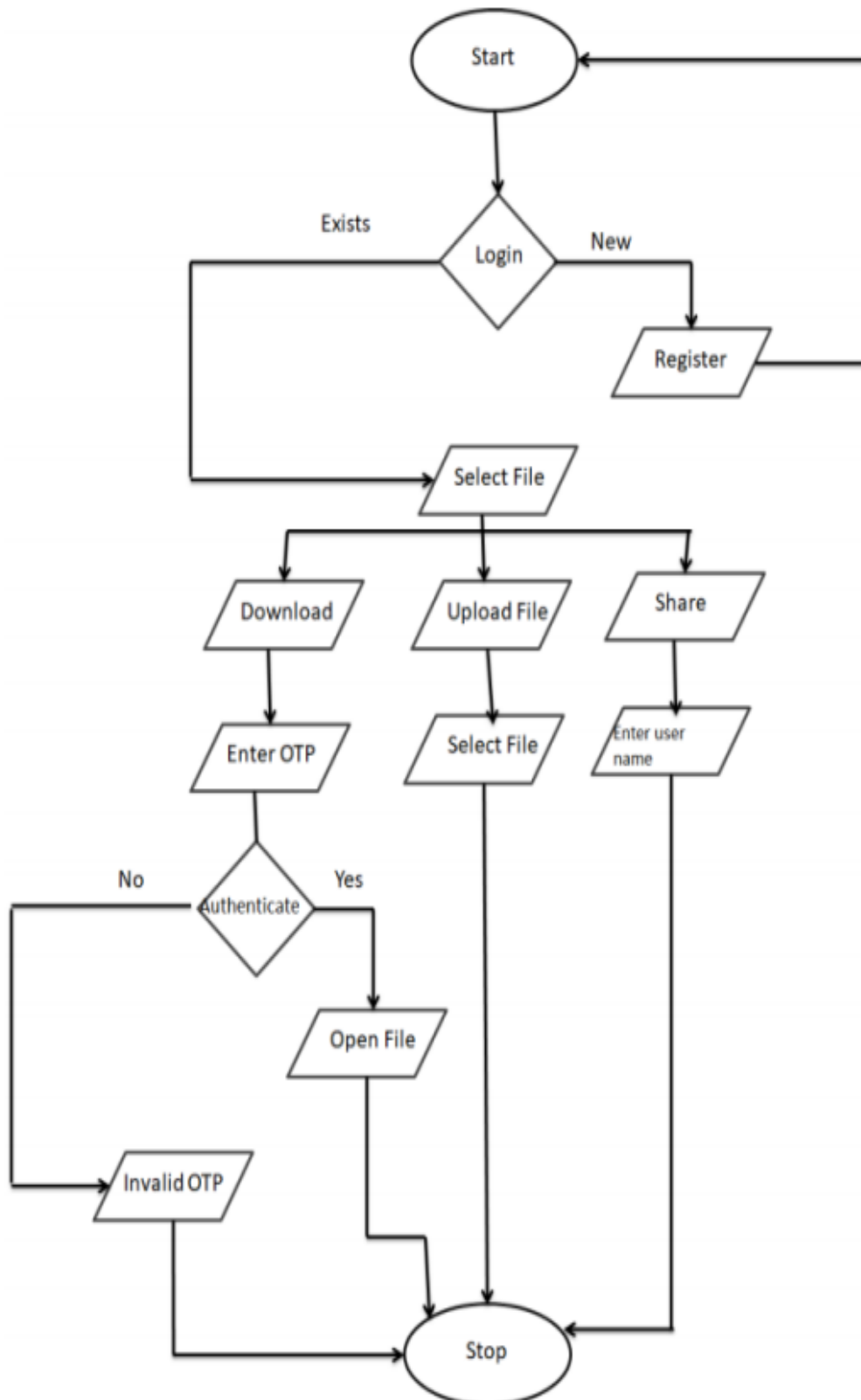
A data flow diagram is the graphical representation of the flow of data through an information system. DFD is very useful in understanding a system and can be efficiently used during analysis.

A DFD shows the flow of data through a system. It view a system as a function that transforms the inputs into desired outputs. Any complex systems will not perform this transformation in a single step and a data will typically undergo a series of transformations before it becomes the output.

With a data flow diagram, users are able to visualize how the system will operate that the system will accomplish and how the system will be implemented, old system data flow diagrams can be drawn up and compared with a new systems data flow diagram to draw comparisons to implement a more efficient system.

Data flow diagrams can be used to provide the end user with a physical idea of where the data they input, ultimately as an effect upon the structure of the whole system.

Dataflow Diagram is as shown below:

**Fig 4.7 Dataflow Diagram**

4.4 Use Case Diagram

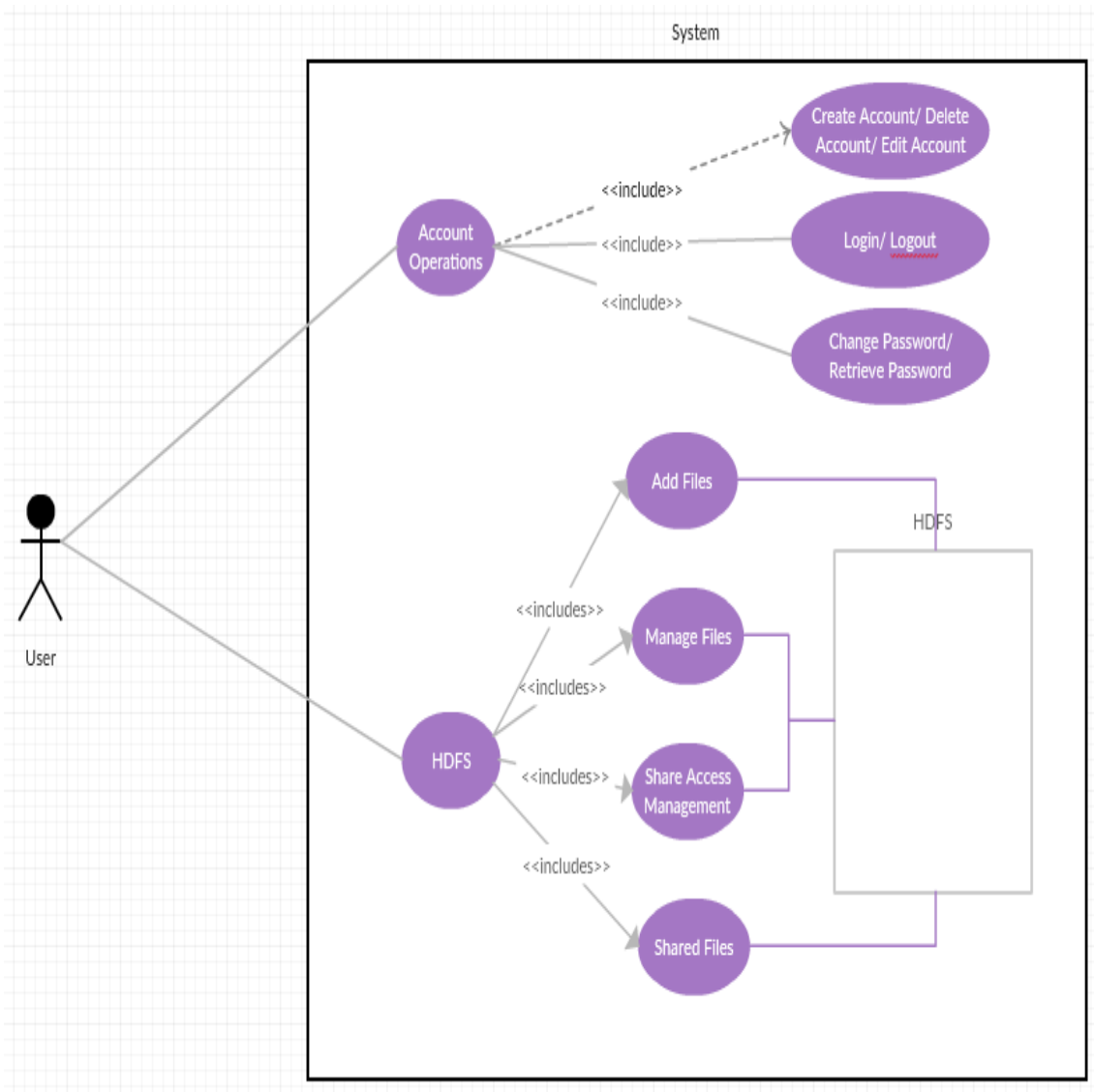


Fig 4.8 Use case diagram of the system

The external objects that interact directly with the system are called **actors**. Actors include humans, external devices and other software systems. The important thing about actors is that they are not under control of the application. In this project, user of the system is the actor.

To find use cases, for each actor, list the fundamentally different ways in which the actor uses the system. Each of these ways is a use case.

4.5 Sequence diagram

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It shows the participants in an interaction and the sequence of messages among them; each participant is assigned a column in a table.

4.5.1 Sequence Diagram 1

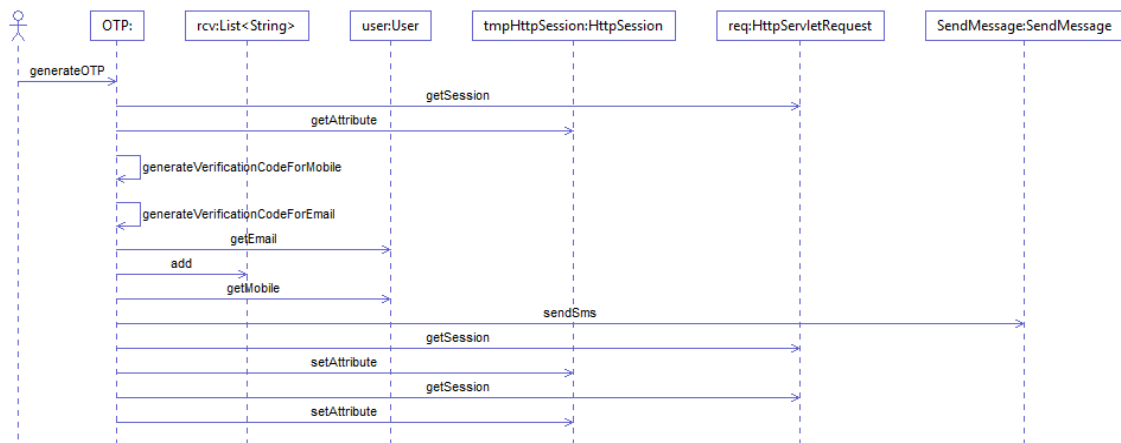


Fig 4.9 Sequence Diagram for OTP Generation

4.5.2 Sequence Diagram 2

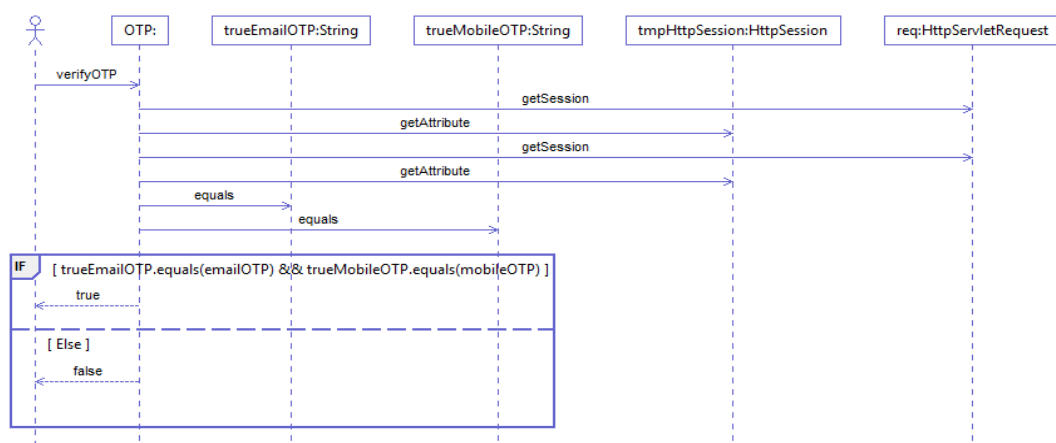


Fig 4.10 Sequence Diagram for OTP Verification

4.6 Class Diagrams

A class diagram in the Unified Modelling 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.

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modelling translating the models into programming code.

4.6.1 Class diagram 1

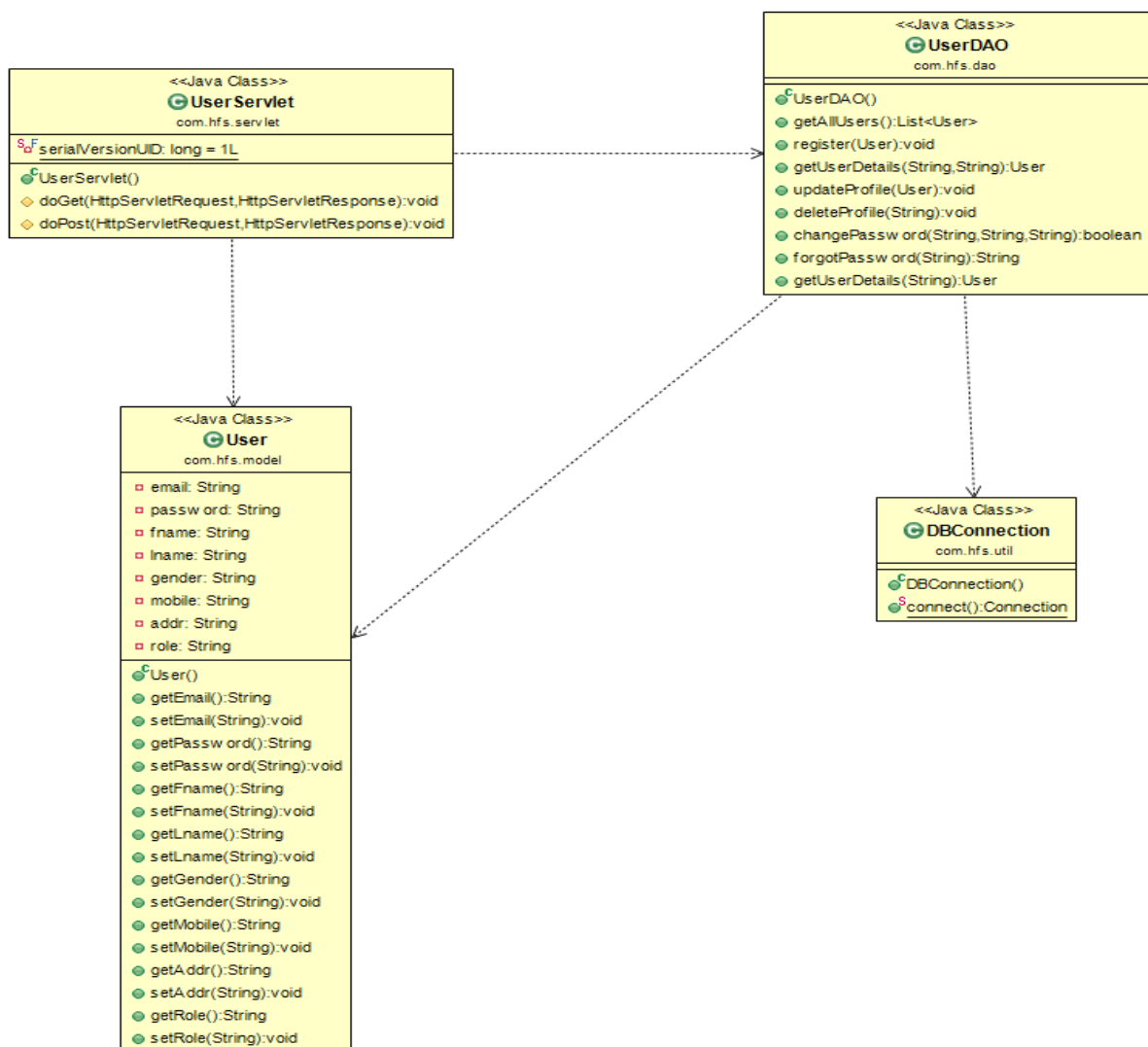


Fig 4.11 Class Diagram – 1(User Dao)

4.6.2 Class diagram 2

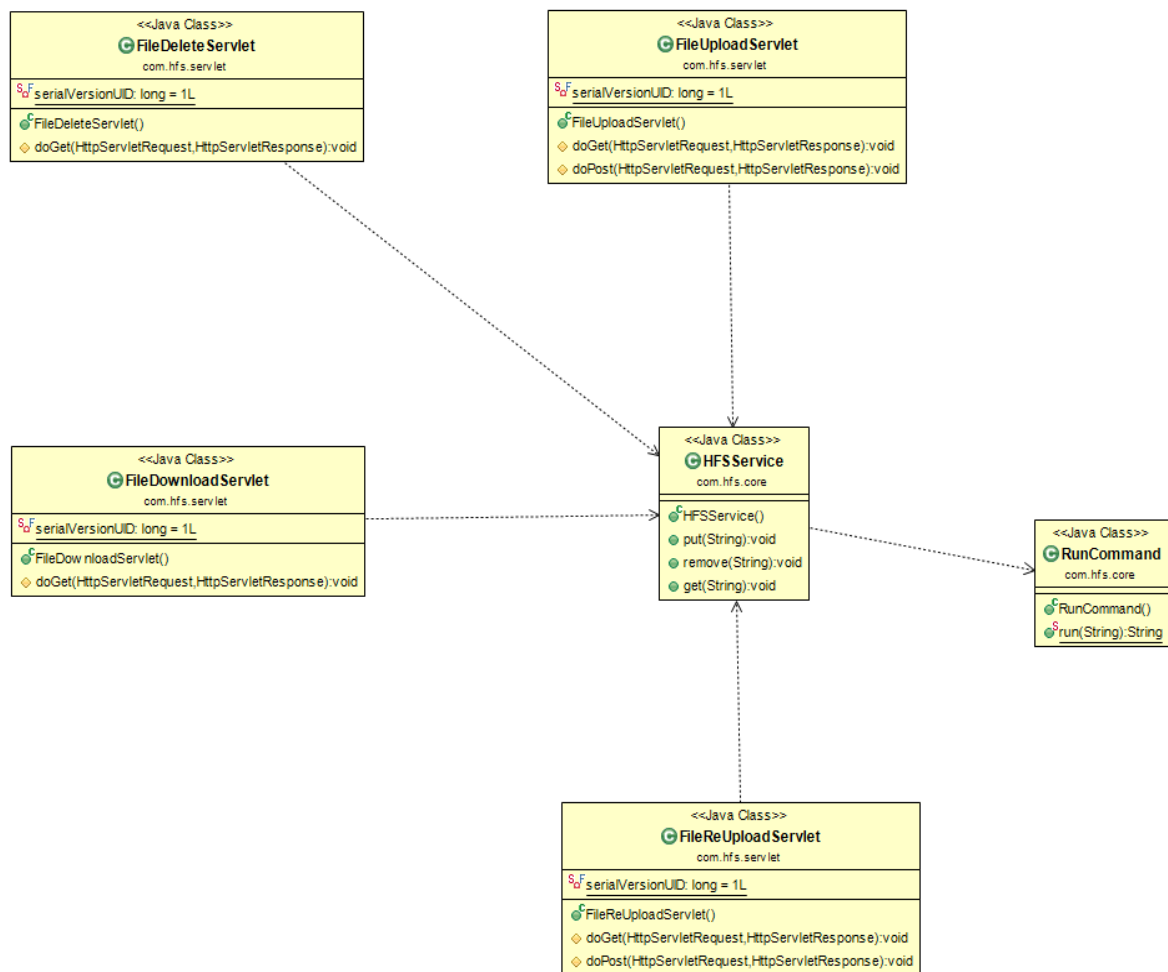


Fig 4.12 Class-Diagram 2(File upload-download)

4.6.3 Class Diagram 3

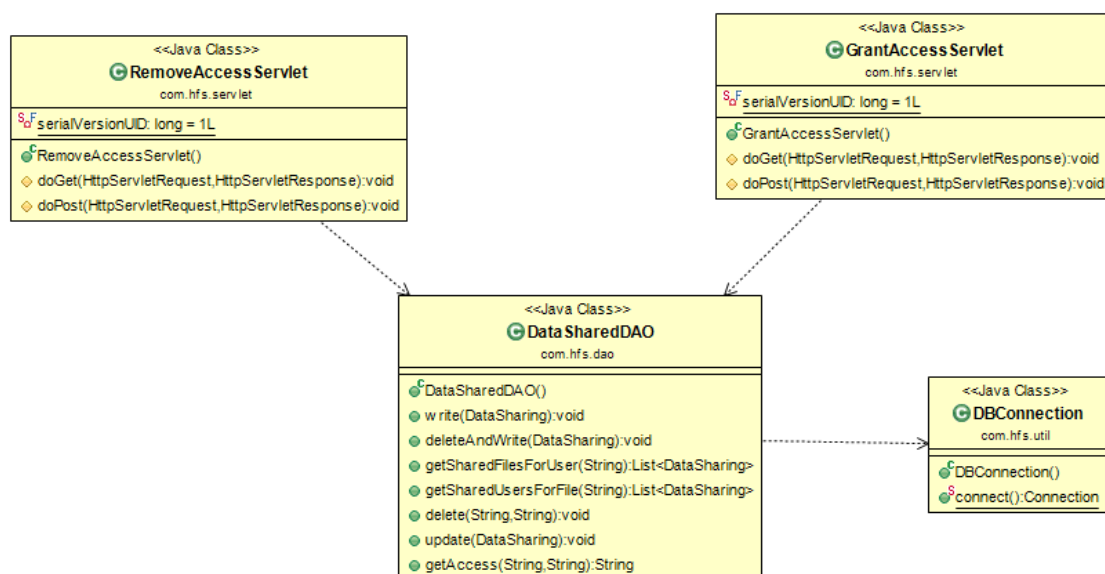


Fig 4.13 Class-Diagram 3(Access Granting)

4.6.4 Class Diagram 4

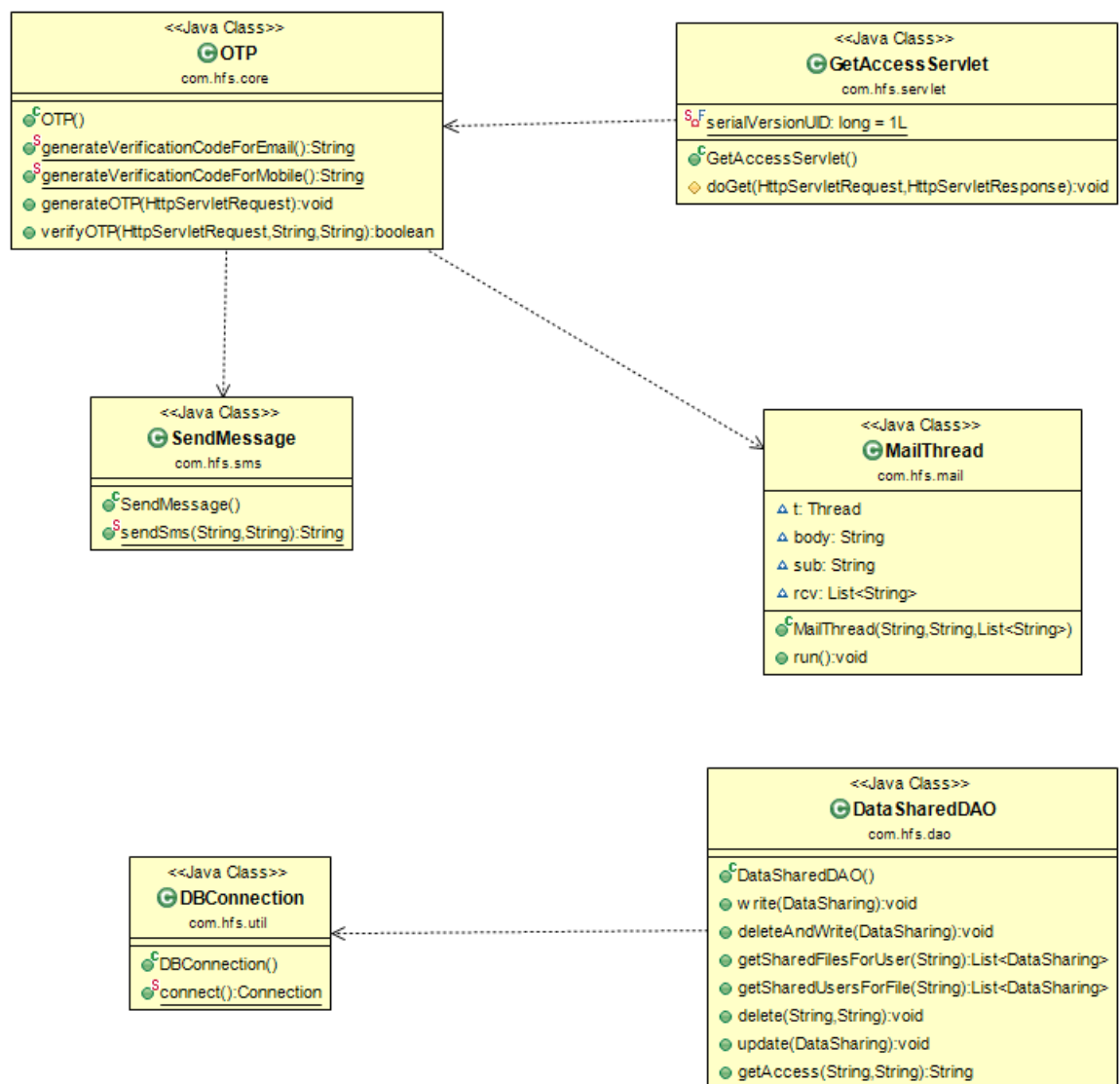


Fig 4.14 Class-Diagram 4(OTP Authentication)

Chapter 5

IMPLEMENTAON

5.1 Overview of System Implementation

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. In other words, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. Many implementations may exist for a given specification or standard.

Implementation is one of the most important phases of the Software Development Life Cycle (SDLC). It encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. Specifically, it involves coding the system using a particular programming language and transferring the design into an actual working system.

This phase of the system is conducted with the idea that whatever is designed should be implemented; keeping in mind that it fulfills user requirements, objective and scope of the system. The implementation phase produces the solution to the user problem.

This project is implemented considering the following aspects:

1. Usability Aspect.
2. Technical Aspect.

5.1.1 UsabilityAspect

a) The project is implemented as a Java application

- Firstly, Java provides a wonderful library which simplifies the implementation part of it.
- Secondly, JAVA is platform independent, meaning the project can run on

literally any platform which has JVM installed within it.

- Thirdly, Oracle Corporation claims more than 70 billion devices run on JAVA which makes the end users used to it.
- Lastly, it can be readily portable to any devices like mobile phones, iPad, PDA, and any hand held devices that are capable of running JAVA.

b) The user-friendly interface using Java's view architecture

The interface provided by this application is very user friendly and is developed using Java Swings.

A) Servers

1) Apache Tomcat to develop the product

Apache Tomcat (or simply Tomcat, formerly also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run. Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files

2) JBOSS Application server to host the product

WildFly, formerly known as JavaBeans Open Source Software Application Server (JBoss AS, or simply JBoss) is an application server that implements the Java Platform, Enterprise Edition (Java EE).

JBoss is written in Java and as such is cross-platform: usable on any operating system that supports Java. JBoss was developed by JBoss, now a division of Red Hat. Licensed under the terms of the GNU Lesser General Public License, JBoss is free and open source software. The renaming to WildFly was done to reduce confusion.

The renaming only affects the JBoss Application Server project. The JBoss Community or the Red Hat JBoss product line (with JBoss Enterprise Application Platform) all retain their names.

3) Database

MySQL officially, but also called /maɪ ˈsiːkwəl/ "My Sequel") is (as of 2008) the world's most widely used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. It is named after co-founder Michael Widenius' daughter, My. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr, Nokia.com, and YouTube.

5.2 Pseudocode

Pseudo code is an informal high-level description of the operating principle of a computer program or other algorithm. The purpose of using pseudo code is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

No standard for pseudo code syntax exists, as a program in pseudo code is not an executable program. Pseudo code resembles, but should not be confused with skeleton

programs, including dummy code, which can be compiled without errors. This contains several folders that make up to an entire plugin. Each folder has a specific function that helps in the functioning of the plugin.

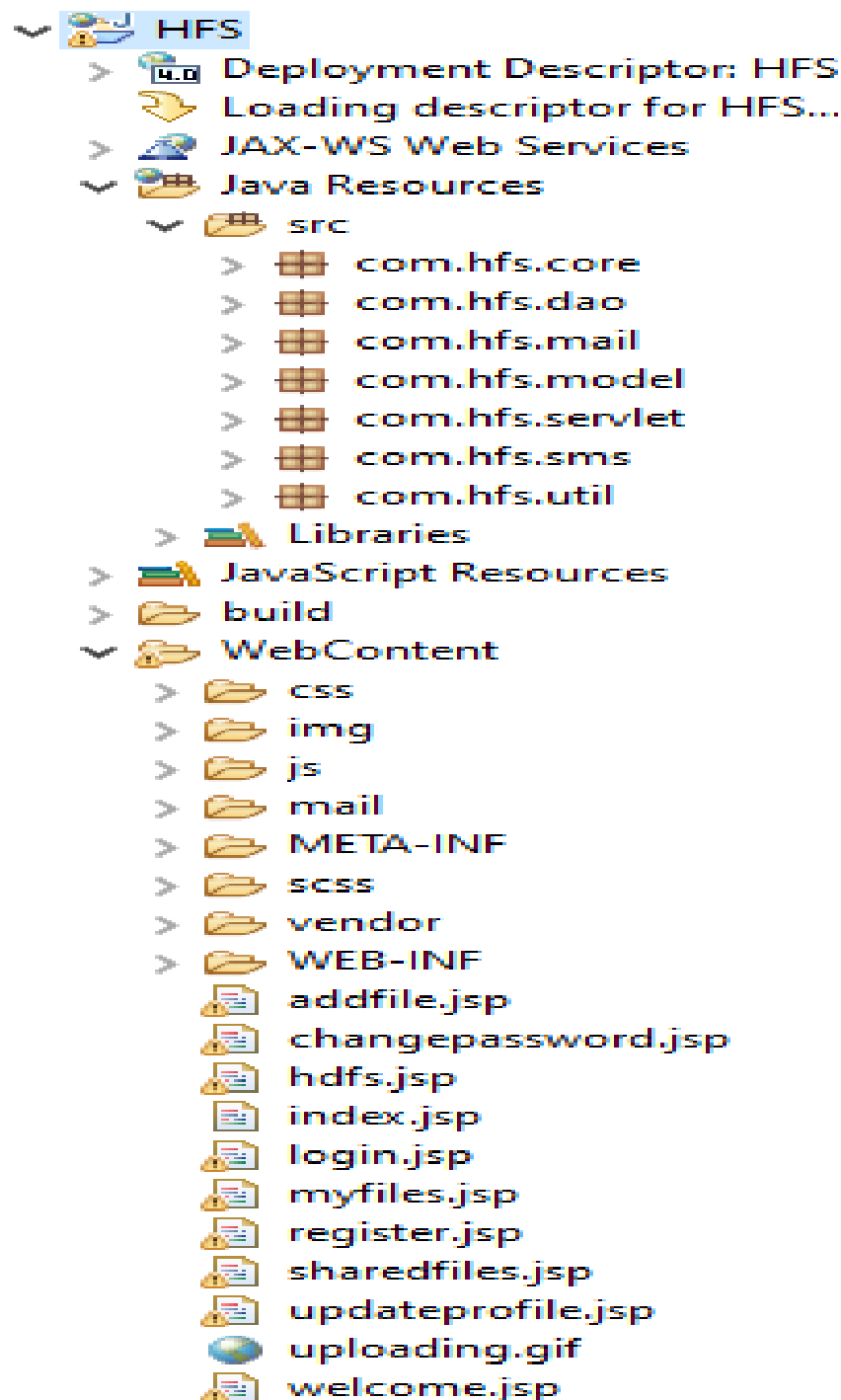


Fig 5.1 Directory Structure

The files written under the folder named 'Java Resources' constitutes the back end logic. The files written under the folder named 'WebContent' constitutes the front end logic. The mapping from front end request to the backend servlet will be written inside

web.xml.

5.2.1 HFSService.java

```
public class HFSService
{
    public void put(String filename) throws Exception
    {
        String command = Constants.HADOOP_INSTALLATION_PATH +
        "/hadoop fs -put " + Constants.TEMP_FILE_UPLOAD +
        File.separator + filename + " " + Constants.DATA_PATH;
        RunCommand.run(command);
    }

    public void remove(String filename) throws Exception
    {
        String command = Constants.HADOOP_INSTALLATION_PATH +
        "/hadoop fs -rm " + Constants.DATA_PATH + File.separator +
        filename;
        RunCommand.run(command);
    }

    public void get(String filename) throws Exception
    {
        String command = Constants.HADOOP_INSTALLATION_PATH +
        "/hadoop fs -get " + Constants.DATA_PATH + File.separator +
        filename + " " + Constants.TEMP_FILE_DOWNLOAD;
        RunCommand.run(command);
    }
}
```

5.2.2 Otp.java

```
public class OTP
{
    public static String generateVerificationCodeForEmail()
    {
        String alphanumericString =
        "ABCDEFGHJKLMNOPQRSTUVWXYZ"
        + "0123456789"
        + "abcdefghijklmnopqrstuvwxyz";
        StringBuilder sb = new StringBuilder(10);
        for (int i = 0; i < 10; i++)
        {
            int index = (int) (alphanumericString.length() *
            Math.random());
            sb.append(alphanumericString.charAt(index));
        }
    }
}
```

```
        return sb.toString();
    }

    public static String generateVerificationCodeForMobile()
    {
        String numericString = "0123456789";
        StringBuilder sb = new StringBuilder(6);
        for (int i = 0; i < 6; i++)
        {
            int index = (int) (numericString.length() *
Math.random());
            sb.append(numericString.charAt(index));
        }
        return sb.toString();
    }

    public void generateOTP(HttpServletRequest req)
    {
        User user = (User)
req.getSession().getAttribute("user");
        String mobileOTP = generateVerificationCodeForMobile();
        String emailOTP = generateVerificationCodeForEmail();
        List<String> rcv = new ArrayList<String>();
        rcv.add(user.getEmail());
        new MailThread("Hello, <br/>Your Email OTP for
accessing File on HDFS is : <br/>" + emailOTP, "OTP for
Accessing File on HDFS", rcv);
        SendMessage.sendSms(user.getMobile(), "Your mobile OTP
for accessing File on HDFS is: " + mobileOTP);

        System.out.println("Email OTP: " + emailOTP);
        System.out.println("Mobile OTP: " + mobileOTP);

        req.getSession().setAttribute("mobileOTP", mobileOTP);
        req.getSession().setAttribute("emailOTP", emailOTP);
    }

    public boolean verifyOTP(HttpServletRequest req, String
mobileOTP, String emailOTP)
    {
        String trueEmailOTP = (String)
req.getSession().getAttribute("emailOTP");
        String trueMobileOTP = (String)
req.getSession().getAttribute("mobileOTP");
        if (trueEmailOTP.equals(emailOTP) &&
trueMobileOTP.equals(mobileOTP))
            return true;

        else

            return false;
    }
}
```

```

    }
}

```

5.2.3 RunCommand.java

```

public class RunCommand
{
    public static String run(String cmd)
    {
        String s;
        Process p;
        String result = "";
        try
        {
            p = Runtime.getRuntime().exec(cmd);
            BufferedReader br = new BufferedReader(new
InputStreamReader(p.getInputStream()));
            while ((s = br.readLine()) != null)
            {
                result += s;
                result += "@-@";
            }
            p.waitFor();
            p.destroy();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return result;
    }
}

```

5.2.4 FileUploadServlet

```

public class FileUploadServlet extends HttpServlet
{
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException,
IOException
    {
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req,
HttpServletResponse resp) throws ServletException,

```



```

IOException
{
    User uModel = (User)
req.getSession().getAttribute("user");
    new MultipartRequest(req, Constants.TEMP_FILE_UPLOAD);
    File folder = new File(Constants.TEMP_FILE_UPLOAD);
    File file = folder.listFiles()[0];
    String prefix =
String.valueOf(System.currentTimeMillis());
    File newFile = new File(Constants.TEMP_FILE_UPLOAD +
File.separator + prefix + "_" + file.getName());
    file.renameTo(newFile);

    HFSService hfsService = new HFSService();
    try
    {
        hfsService.put(newFile.getName());
    }
    catch (Exception e1)
    {
        e1.printStackTrace();
        resp.sendRedirect("addfile.jsp?msg=Error: " +
e1.getMessage());
    }

    newFile.delete();
    file.delete();

    DataDAO dDao = new DataDAO();
    Data dModel = new Data();
    dModel.setEntrytime(new
Timestamp(System.currentTimeMillis()));
    dModel.setFilename(newFile.getName());
    dModel.setId(prefix);

    dModel.setMimetype(newFile.getName().substring(newFile.getNam
e().lastIndexOf(".") + 1));
    dModel.setUsername(uModel.getEmail());
    dModel.setVersion("1");
    try
    {
        dDao.write(dModel);
        resp.sendRedirect("addfile.jsp?msg=File Upload to
HDFS successful");
    }
    catch (Exception e2)
    {
        e2.printStackTrace();
        resp.sendRedirect("addfile.jsp?msg=Error: " +
e2.getMessage());
    }
}

```

```

    }

}

```

5.2.5 FileDownloadServlet.java

```

public class FileDownloadServlet extends HttpServlet
{

    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException,
        IOException
    {
        String filename = req.getParameter("filename");

        HFSService hfsService = new HFSService();
        try
        {
            hfsService.get(filename);
        }
        catch (Exception e)
        {
            e.printStackTrace();
            resp.sendRedirect("myfiles.jsp?msg=Error: " +
                e.getMessage());
        }

        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        String filepath = Constants.TEMP_FILE_DOWNLOAD +
            File.separator;
        resp.setContentType("APPLICATION/OCTET-STREAM");
        resp.setHeader("Content-Disposition", "attachment;
            filename=\"" + filename + "\"");

        FileInputStream fileInputStream = new
            FileInputStream(filepath + filename);

        int i;
        while ((i = fileInputStream.read()) != -1)
        {

        }
        fileInputStream.close();
        out.close();

        new
            File(Constants.TEMP_FILE_DOWNLOAD+File.separator+filename).de

```

```

lete();

    resp.sendRedirect("myfiles.jsp");
}

}

```

5.2.6 FileReuploadServlet.java

```

public class FileReUploadServlet extends HttpServlet
{
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException,
        IOException
    {
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException,
        IOException
    {
        User userModel = (User)
req.getSession().getAttribute("user");
        new MultipartRequest(req, Constants.TEMP_FILE_UPLOAD);
        String filename = req.getParameter("filename");

        File folder = new File(Constants.TEMP_FILE_UPLOAD);
        File file = folder.listFiles()[0];
        File newFile = new File(Constants.TEMP_FILE_UPLOAD +
File.separator + filename);
        file.renameTo(newFile);

        HFSService hfsService = new HFSService();
        try
        {
            hfsService.remove(newFile.getName());
            hfsService.put(newFile.getName());
        }
        catch (Exception e1)
        {
            e1.printStackTrace();
            resp.sendRedirect("myfiles.jsp?msg=Error: " +
e1.getMessage());
        }

        newFile.delete();
    }
}

```

```

        file.delete();

        DataDAO dDao = new DataDAO();
        Data dModel = new Data();
        dModel.setEntrytime(new
Timestamp(System.currentTimeMillis()));
        dModel.setFilename(newFile.getName());
        dModel.setId(filename.substring(0,
filename.indexOf("_")));
        dModel.setMimetype(newFile.getName().substring(newFile.getNam
e().lastIndexOf(".") + 1));

        dModel.setUsername(uModel.getEmail());
        String from = req.getParameter("from");

        try
        {
            dDao.reupload(dModel);
            if (from != null && from.equals("shared"))
            {
                resp.sendRedirect("sharedfiles.jsp?msg=New
Version of " + filename.substring(filename.indexOf("_") + 1)
+ " uploaded successfully");
            }
            else
            {
                resp.sendRedirect("myfiles.jsp?msg=New Version of
" + filename.substring(filename.indexOf("_") + 1) + "
uploaded successfully");
            }
        }
        catch (Exception e2)
        {
            e2.printStackTrace();
            if (from != null && from.equals("shared"))
            {
                resp.sendRedirect("sharedfiles.jsp?msg=Error:
"+e2.getMessage());
            }
            else
            {
                resp.sendRedirect("myfiles.jsp?msg=Error:
"+e2.getMessage());
            }
        }
    }
}

```

5.2.7 FileDeleteServlet.java

```
public class FileDeleteServlet extends HttpServlet
{

    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException,
        IOException
    {
        String from = req.getParameter("from");
        try
        {
            String filename = req.getParameter("filename");
            HFSService hfsService = new HFSService();
            hfsService.remove(filename);
            DataDAO dDao = new DataDAO();

            dDao.delete(filename.substring(0,
filename.indexOf("_")));
            if (from != null && from.equals("shared"))
            {
                resp.sendRedirect("sharedfiles.jsp?msg=" +
filename.substring(filename.indexOf("_") + 1) + "File
Deleted");
            }
            else
            {
                resp.sendRedirect("myfiles.jsp?msg=" +
filename.substring(filename.indexOf("_") + 1) + "File
Deleted");
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
            if (from != null && from.equals("shared"))
            {
                resp.sendRedirect("sharedfiles.jsp?msg=Error: " +
e.getMessage());
            }
            else
            {
                resp.sendRedirect("myfiles.jsp?msg=Error: " +
e.getMessage());
            }
        }
    }
}
```

```
}
```

5.3 Implementation Support

5.3.1 Installation of Eclipse

The following steps should be followed to install eclipse:

- *Installation of JVM:* Regardless of the operating system, some Java virtual machine (JVM) has to be installed. A Java Runtime Environment (JRE), or a Java Development Kit (JDK) can be installed depending on what is to be done with Eclipse. If Eclipse is intended for Java development, then a JDK (the JDK includes--among other useful things--the source code for the standard Java libraries)has to be installed.
- Download Eclipse from the Eclipse Downloads Page.
- The download will be delivered as a compressed (i.e. a ".zip", or ".tar.gz") file. Decompress this file into the directory of your choice (e.g. "c:\Program Files\Eclipse Indigo" on Windows). You can optionally create a shortcut of the executable file ("eclipse.exe" on Windows, or "eclipse" on Linux).

5.3.2 Installation of Apache Tomcat Server

The following steps should be followed to install Apache Tomcat in Eclipse:

- If Apache Tomcat is not present on the machine, it has to be downloaded and unzipped.
- Start the Eclipse WTP workbench.
- Open Window -> Preferences -> Server -> Installed Runtime to create a Tomcat installed runtime.
- Click on Add to open the New Server Runtime dialog, then select your runtime under Apache (Apache Tomcat v7.0 in this project).
- Ensure the selected JRE is a full JDK and is of a version that will satisfy Apache Tomcat (this scenario was written using SUN JDK 1.6.029). If necessary, you can click on Installed JREs to add JDKs to Eclipse.
- Click Finish. Click Next, and fill in your Tomcat installation directory

Chapter 6

TESTING

6.1 Introduction

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. The system has been verified and validated by running the test data and live data.

6.2 Levels of Testing

6.2.1 Unit Testing

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In object-oriented programming a unit is often an entire interface, such as a class, but could be an individual method.

For unit testing first we adopted the code testing strategy, which examined the logic of program. During the development process itself all the syntax errors etc. got rooted out. For this developed test case that result in executing every instruction in the program or module i.e. every path through program was tested. Test cases are data chosen at random to check every possible branch after all the loops.

Test cases

The unit test cases for the project File Sharing over Big Data Environment are as shown in Table 6.1:

Table 6.1: Test cases for the project

Steps	Test Action	Results
Step 1	Enter the URL http://<HOST>:8080/HFS/index.jsp	Index page loaded successfully
Step 2	Click on Register	Register page loaded successfully
Step 3	Create a new account	Account Creation Successful
Step 4	Click on login	Login page loaded
Step 5	Login to an existing account	Login Successful
Step 6	Click on HDFS	HDFS page loaded successfully
Step 7	Click on Add Files	Add Files Page loaded successfully
Step 8	Select any file on the file system and click on upload	The upload process started
Step 9	Verify if the upload process is completed	The upload process is completed successfully
Step 10	Login to Hadoop administration console and check if the file is uploaded to Hadoop system	Yes it is upladed
Step 11	Click on browse button again and add another file	The file added to Hadoop successfully
Step 12	Click on My Files	My files page loaded
Step 13	Verify the contents	Contents in the My files page is correct

Step 14	Click on any files under drop down click on Download	File download completed
Step 15	Verify the contents of downloaded file	Downloaded file content is correct
Step 16	Click on file menu and click on sharing	Sharing screen loaded
Step 17	Share the file with any other user granting readonly access	Share successful
Step 18	Share the file with any other user granting read write access	Share successful
Step 19	Click on file menu and click on upload new version	New version upload screen shown
Step 20	Upload the new version of the file	New version of the file upload done
Step 21	Check if the version number is increased	Yes it is
Step 22	Check if the file delete is working	Yes it is working
Step 23	Login as the user who have the read only access to the shared file	Login successful
Step 24	Click on Shared Files under My Files	All the shared files are shown correctly
Step 25	Under the file, click on Get Access	OTP received to mobile and email
Step 26	Enter the OTP and verify	OTP verified successfully and read only access granted
Step 27	Follow the similar steps for the user having read write access	Read write access is working fine

6.2.1.1 User Input

User will be inputting all the data from using a web browser.

6.2.1.2 Error Handling

In this system, we have tried to handle all the errors that occurred while running the application. the common errors we saw were reading a tuple with an attribute set to null and database connection getting lost.

For Testing we used Top-Down design a decomposition process which focuses as the flow of control, at latter strategies concern itself with code production. The first step is to study the overall aspects of the tasks at hand and break it into a number of independent modules. The second step is to break one of these modules further into independent sub modules. One of the important features is that each level the details at lower levels are hidden. So unit testing was performed first and then system testing.

6.2.2 Integration Testing

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined may not produce the desired functions. Integrated testing is the systematic testing to uncover the errors with an interface. This testing is done with simple data and developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance.

Steps to perform integration testing:

Step 1: Create a Test Plan

Step 2: Create Test Cases and Test Data

Step 3: Once the components have been integrated execute the test cases

Step 4: Fix the bugs if any and re test the code

Step 5: Repeat the test cycle until the components have been successfully integrated

Table 6.2: Test cases for integration testing

Name of the Test	Integration testing
Test plan	To check whether the system works properly when all the modules are integrated.
Test Data	Sample files

6.2.3 System testing

Ultimately, software is included with other system components and the set of system validation and integration tests are performed. System testing is a series of different tests whose main aim is to fully exercise the computer-based system. Although each test has a different role all work should verify that all system elements are properly integrated and formed allocated functions.

Table 6.3: Test cases for Input-Output

Name of the Test	System Testing
Item being tested	Over all functioning of GUI with all functions properly linked.
Sample Input	Sample files
Expected Output	All the modules like control center, monitor center, etc working as expected
Actual Output	Application reacts to user inputs in expected manner.
Remarks	Successful

6.2.4 Validation Testing

At the culmination of black box testing, software is completely assembled is as a package. Interfacing errors have been uncovered and the correct and final series of tests, i.e., validation tests begins. Validation test is defined with a simple definition that validation succeeds when the software function in a manner that can be reasonably accepted by the customer.

6.2.5 Output Testing

After performing validation testing, the next step is output testing of the proposed system. Since the system cannot be useful if it does not produce the required output. Asking the user about the format in which the system is required tests the output displayed or generated by the system is required tests the output displayed or generated by the system under consideration. The output format is considered in two ways, one is on screen format and the other is printed format. The output format on the screen is found to be corrected as

the format was designated in the system has according to the user needs. As for the hard copy the output comes according to the specification requested by the user. The output testing does not result in any correction in the system.

6.2.6 Test data and Output:

Taking various kind soft data plays a vital role in system testing. After preparing the test data system under study is tested using the test data. While testing, errors are again uncovered and corrected by using the above steps and corrections are also noted for future use.

6.2.7 User acceptance Testing:

User acceptance testing of the system is the key factor for the success of the system. A system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system at the time of development and making change whenever required. This is done with regard to the input screen design and output screen design.

6.2.8 GUI Testing:

GUI testing is use to ensure the visual clarity of the system, flexibility of the system, user friendliness of the system. The various components which are to be tested are:

- Relative layout
- Various Links and Buttons

Chapter 7

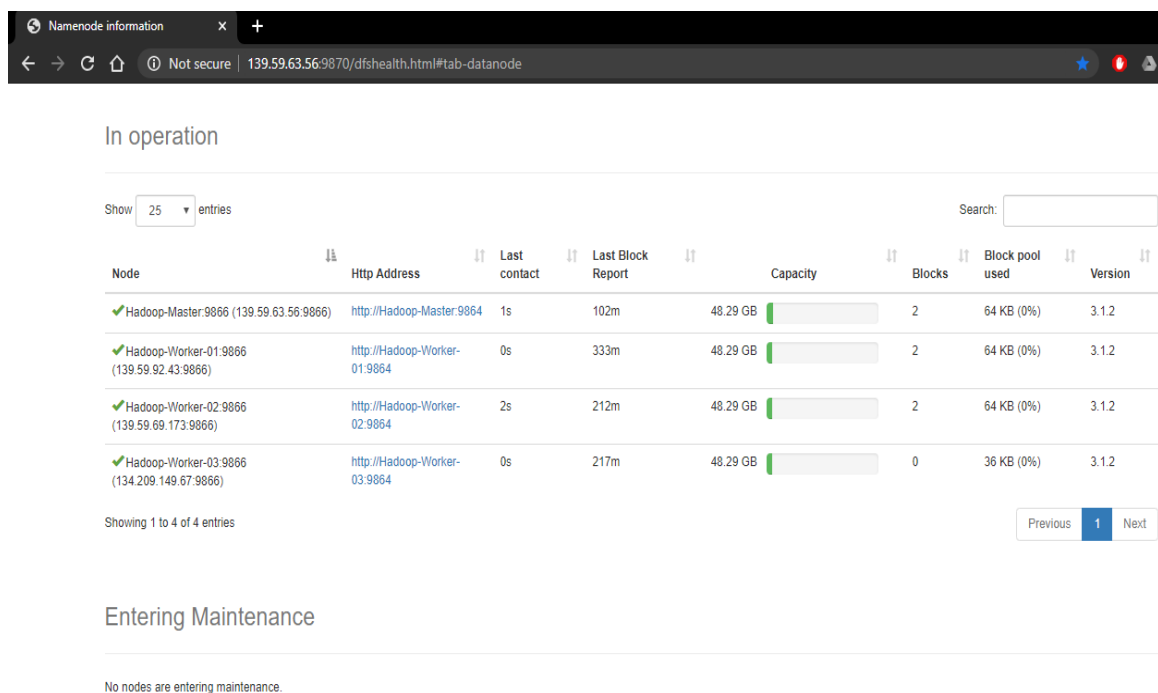
Discussion of Results

7.1 Snapshots

The outcomes of test results for a variety of user interactions with the application are discussed in the following sections of the chapter.

7.1.1 Namenode Information

The fig 7.1 shows that all the hadoop node that are create and in full operation. There are is one master node and three slave node along with the information like last contact, capacity and blocks used.



The screenshot displays the 'Namenode information' page in a web browser. The address bar shows the URL '139.59.63.56:9870/dfshealth.html#tab-datanode'. The page title is 'In operation'. Below the title, there is a search bar and a 'Show 25 entries' dropdown. The main content is a table with columns: Node, Http Address, Last contact, Last Block Report, Capacity, Blocks, Block pool used, and Version. The table lists four nodes: Hadoop-Master-9866, Hadoop-Worker-01-9866, Hadoop-Worker-02-9866, and Hadoop-Worker-03-9866. All nodes are in a 'green' state, indicating they are in full operation. The 'Capacity' column shows 48.29 GB for all nodes. The 'Blocks' column shows 2 blocks for the master and worker nodes, and 0 blocks for the third worker node. The 'Block pool used' column shows 64 KB (0%) for the master and worker nodes, and 36 KB (0%) for the third worker node. The 'Version' column shows 3.1.2 for all nodes. At the bottom of the table, it says 'Showing 1 to 4 of 4 entries' and 'No nodes are entering maintenance.'

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ Hadoop-Master-9866 (139.59.63.56:9866)	http://Hadoop-Master-9866	1s	102m	48.29 GB	2	64 KB (0%)	3.1.2
✓ Hadoop-Worker-01-9866 (139.59.92.43:9866)	http://Hadoop-Worker-01-9866	0s	333m	48.29 GB	2	64 KB (0%)	3.1.2
✓ Hadoop-Worker-02-9866 (139.59.69.173:9866)	http://Hadoop-Worker-02-9866	2s	212m	48.29 GB	2	64 KB (0%)	3.1.2
✓ Hadoop-Worker-03-9866 (134.209.149.67:9866)	http://Hadoop-Worker-03-9866	0s	217m	48.29 GB	0	36 KB (0%)	3.1.2

Fig 7.1 Namenode Information

7.1.2 Index Page

The fig 7.2 shows the intial welcome page of the application which is shown to the user when user types in the url of the application form this page the user can register, login or go back to the home page.

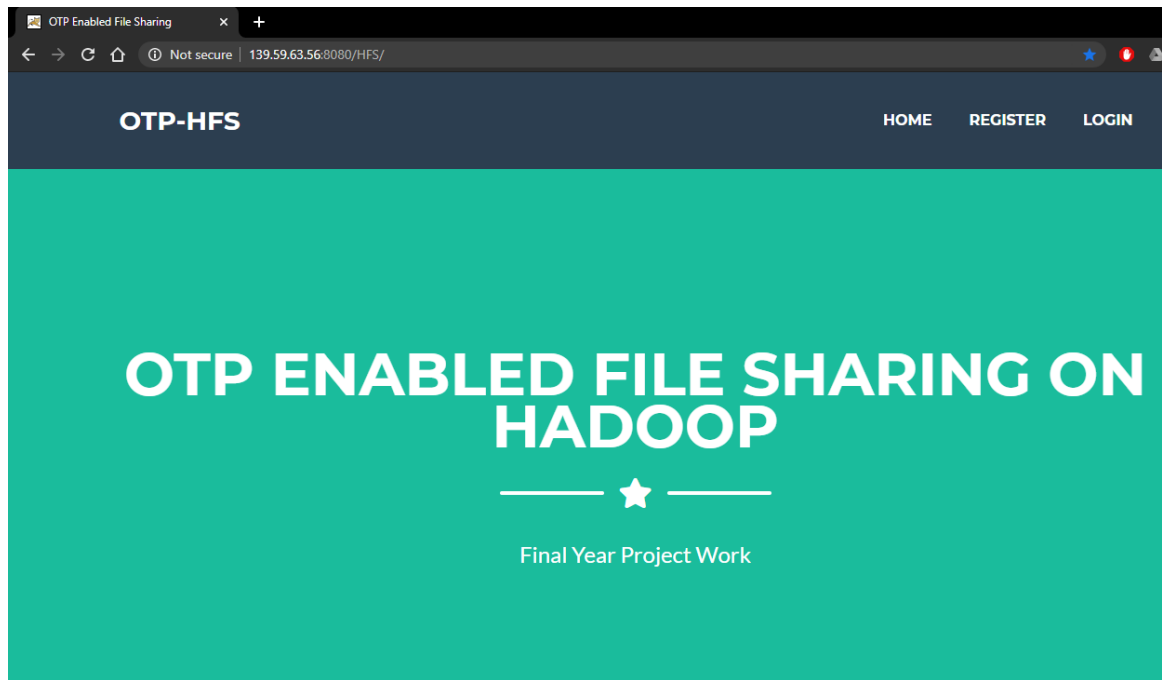


Fig 7.2 Index Page

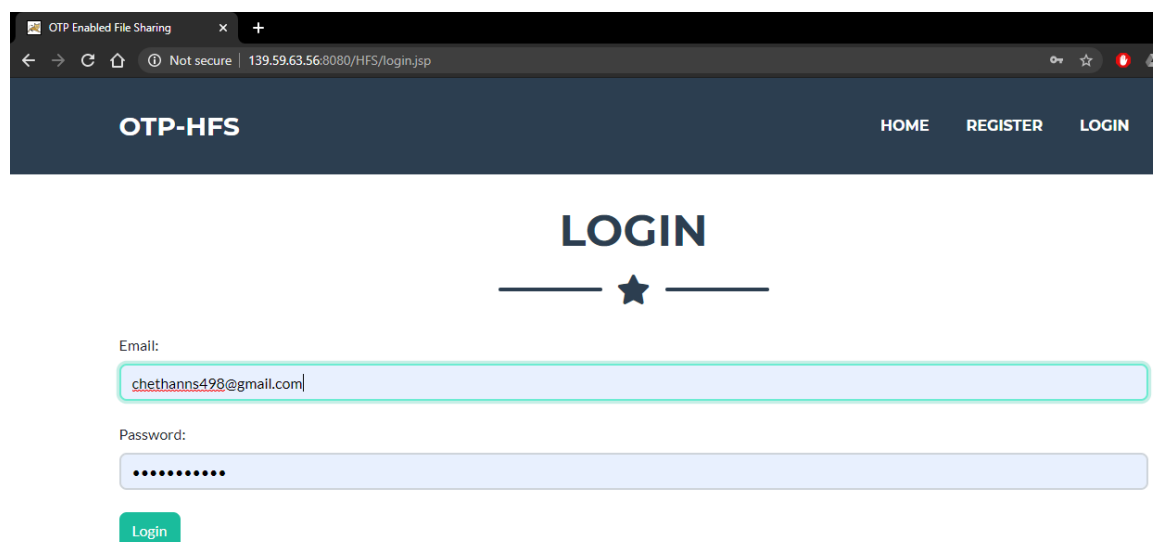
7.1.3 Register Page

The fig 7.3 shows the register page where the user can register to the application. The user has to enter a valid email Id, password, first and last name, gender, mobile number and address.

Fig 7.3 Register Page

7.1.4 Login Page

The fig 7.4 shows the login page, The user can login to the application if he/she is already registered. The user has to enter his email id and password to login.

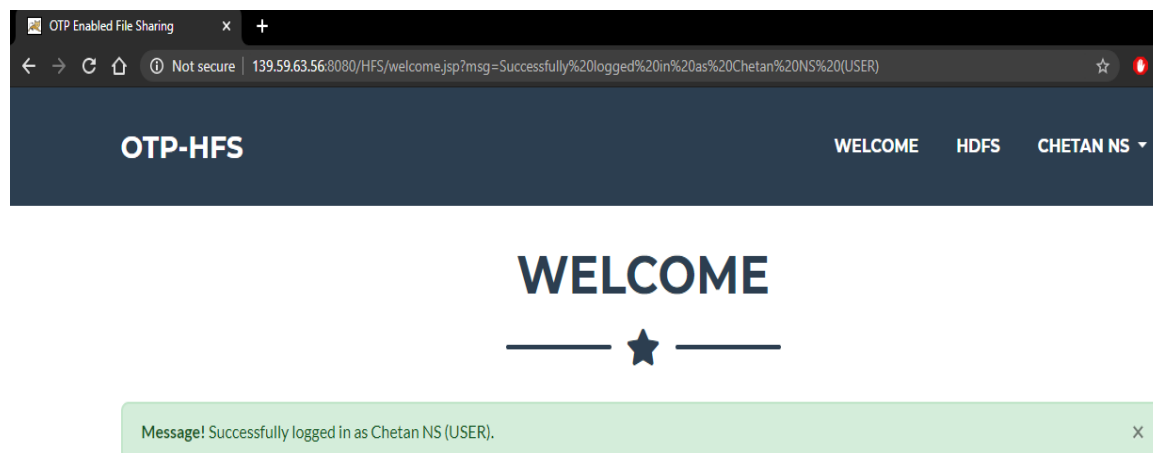


The screenshot shows a web browser window with the address bar displaying "139.59.63.56:8080/HFS/login.jsp". The page has a dark blue header with "OTP-HFS" on the left and "HOME", "REGISTER", and "LOGIN" on the right. The main content area features the word "LOGIN" in large, bold, dark blue letters, centered below a horizontal line with a star in the middle. Below this, there are two input fields: "Email:" with the value "chethanns498@gmail.com" and "Password:" with masked characters ".....". A green "Login" button is positioned below the password field.

Fig 7.4 Login Page

7.1.5 Welcome Page

The fig 7.5 shows the welcome Page, this page is shown when the user has successfully logged in to the application form this page user can perform Hdfs operation and user profile operation.



The screenshot shows a web browser window with the address bar displaying "139.59.63.56:8080/HFS/welcome.jsp?msg=Successfully%20logged%20in%20as%20Chetan%20NS%20(USER)". The page has a dark blue header with "OTP-HFS" on the left and "WELCOME", "HDFS", and "CHETAN NS" on the right. The main content area features the word "WELCOME" in large, bold, dark blue letters, centered below a horizontal line with a star in the middle. Below this, there is a green message box with the text "Message! Successfully logged in as Chetan NS (USER)." and a close button (X).

Fig 7.5 Welcome Page

7.1.6 User Profile Operation

The fig 7.6 show the option to perform user Profile Operation, the Operation that can be performed are Edit Profile, Change Password, delete profile and logout.

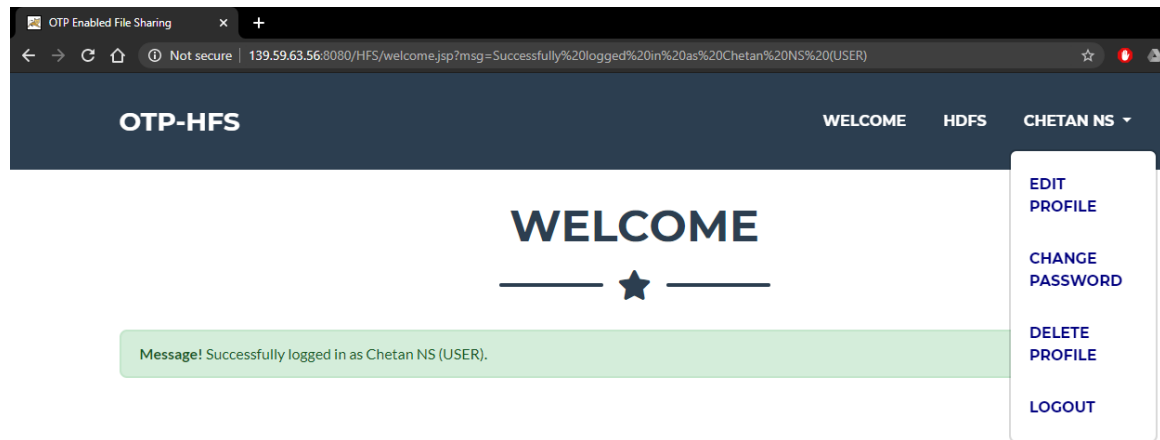


Fig 7.6 User Profile Operation

7.1.7 Update Profile Page

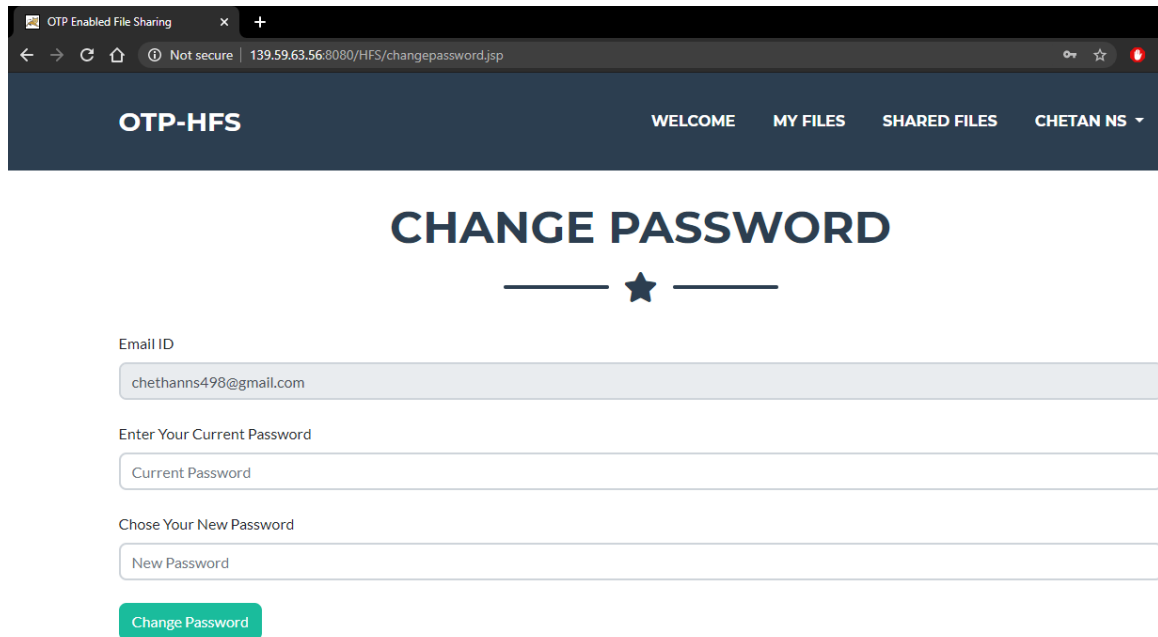
The fig 7.7 shows update profile page, the user can update first name, last name, gender, mobile number, address but user cannot update his email id.

A screenshot of the 'UPDATE PROFILE' page in the OTP-HFS application. The browser's address bar shows the URL '139.59.63.56:8080/HFS/updateprofile.jsp'. The application's header includes the logo 'OTP-HFS' and navigation links 'WELCOME', 'MY FILES', 'SHARED FILES', and 'CHETAN NS'. The main content area features a large 'UPDATE PROFILE' heading with a star icon below it. Below the heading, there are four input fields: 'Enter your Email ID' (containing 'chethanns498@gmail.com'), 'Enter your First name' (containing 'Chetan'), 'Enter your Last name' (containing 'NS'), and 'Select your Gender' (with radio buttons for 'Male' and 'Female').

Fig 7.7 Update Profile Page

7.1.8 Change Password Page

The fig 7.8 shows change password page, the user has to enter the old password in order to change password.

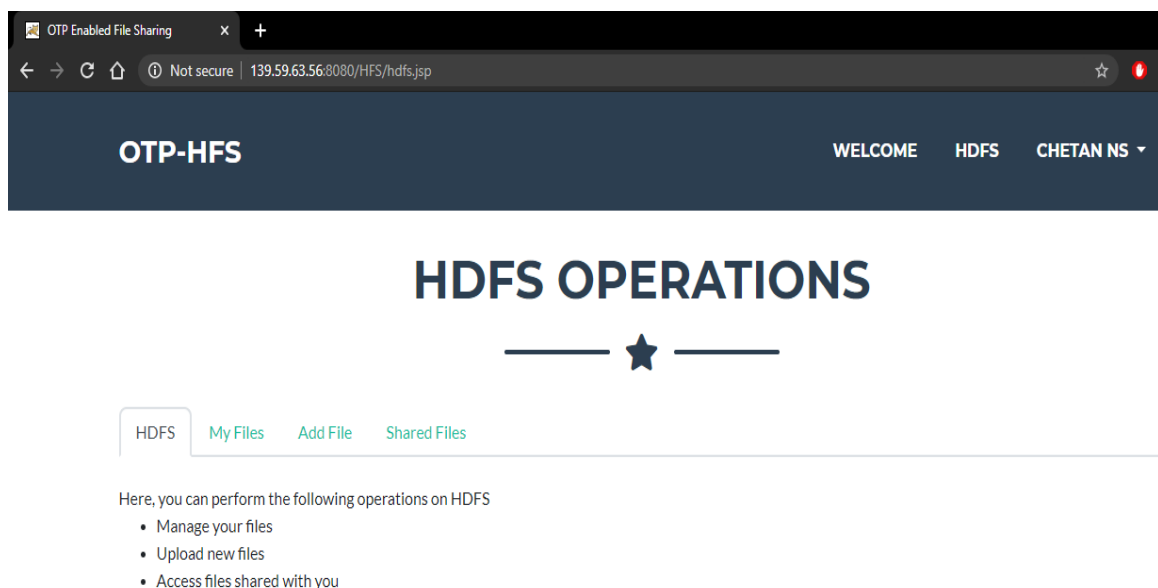


The screenshot shows a web browser window with the address bar displaying "139.59.63.56:8080/HFS/changepassword.jsp". The page has a dark blue header with the text "OTP-HFS" on the left and navigation links "WELCOME", "MY FILES", "SHARED FILES", and "CHETAN NS" on the right. The main content area is white and features the title "CHANGE PASSWORD" in large, bold, dark blue letters, centered and flanked by horizontal lines and a star icon. Below the title, there are three input fields: "Email ID" (containing "chethans498@gmail.com"), "Enter Your Current Password" (containing "Current Password"), and "Chose Your New Password" (containing "New Password"). A green "Change Password" button is located at the bottom left of the form.

Fig 7.8 Change Password Page

7.1.9 HDFS Operations Page

The fig 7.9 shows Hdfs operation page, this is where the user gets to know what operation he/she can perform like manage your files, upload new file, access file shared with user.



The screenshot shows a web browser window with the address bar displaying "139.59.63.56:8080/HFS/hdfs.jsp". The page has a dark blue header with the text "OTP-HFS" on the left and navigation links "WELCOME", "HDFS", and "CHETAN NS" on the right. The main content area is white and features the title "HDFS OPERATIONS" in large, bold, dark blue letters, centered and flanked by horizontal lines and a star icon. Below the title, there is a horizontal menu with four items: "HDFS" (selected), "My Files", "Add File", and "Shared Files". Below the menu, there is a paragraph of text: "Here, you can perform the following operations on HDFS". This is followed by a bulleted list of three items: "Manage your files", "Upload new files", and "Access files shared with you".

Fig 7.9 HDFS Operations Page

7.1.10 My files Page

The fig 7.10 shows myfile page, where the user can view the file the user has uploaded. The files uploaded has file name, file type, version, last modified time.

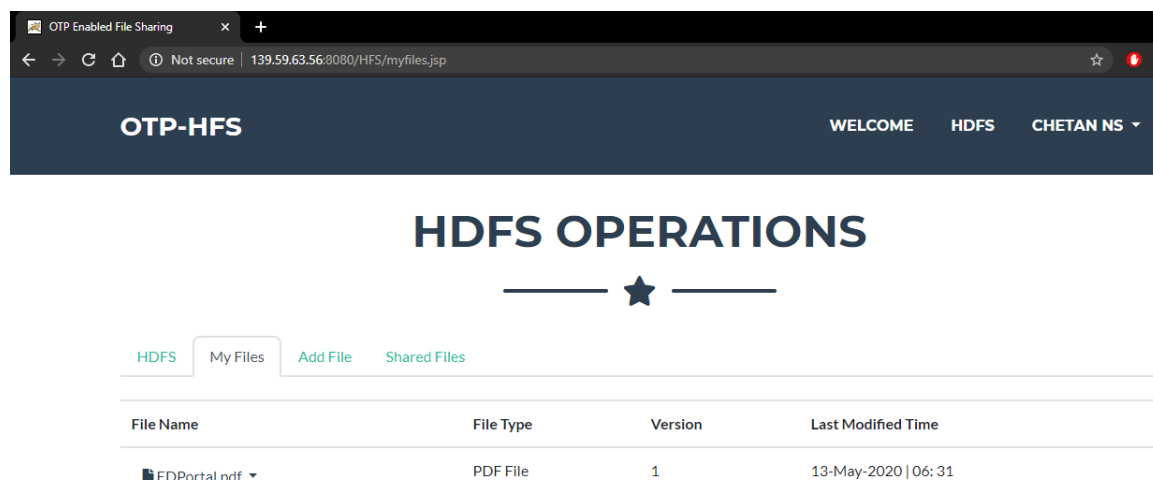


Fig 7.10 Myfile Page.

7.1.11 Add File Page

The fig 7.11 shows add file page, where the registered user is provided with an HTML interface where he/she can upload the file into the Hadoop system. The user can do this using the provided browse button.

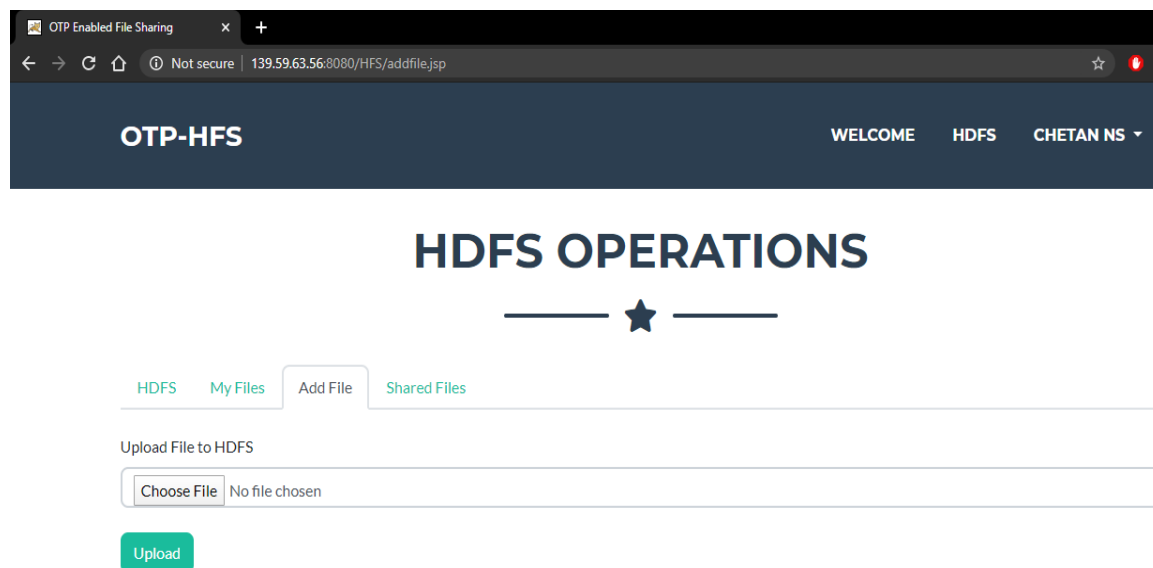


Fig 7.11 Add File Page

7.1.12 Manage file

The fig 7.12 shows manage file, the user can perform operation like downloading the uploaded file, sharing the file, uploading the new version of the file and also deleting the file.

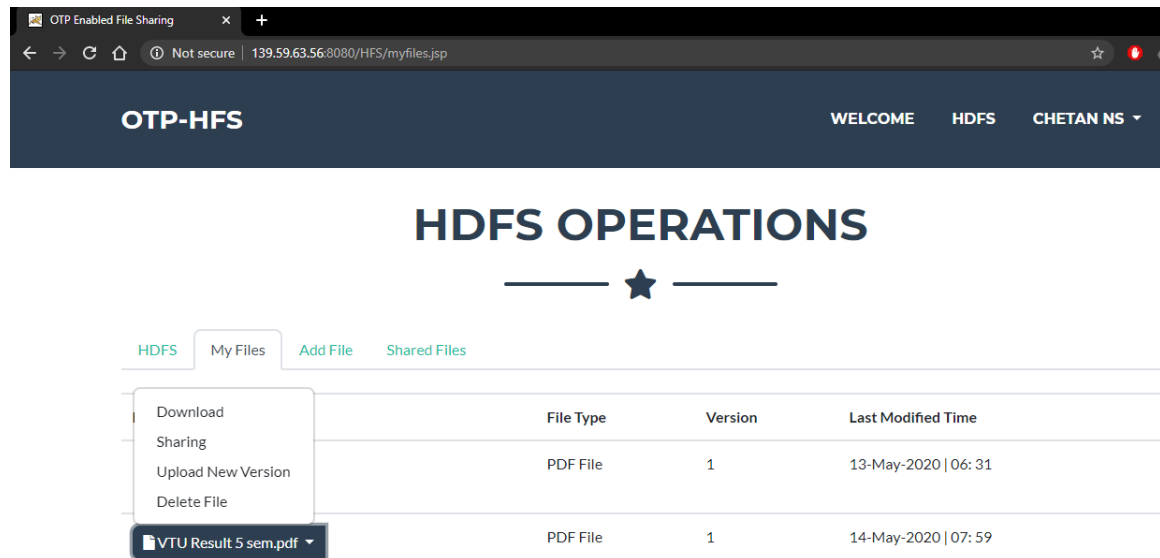


Fig 7.12 Manage file

7.1.13 Delete File

The fig 7.13 shows the delete file, the user can delete the file uploaded when the file is deleted the the user is given a message that the file is deleted.

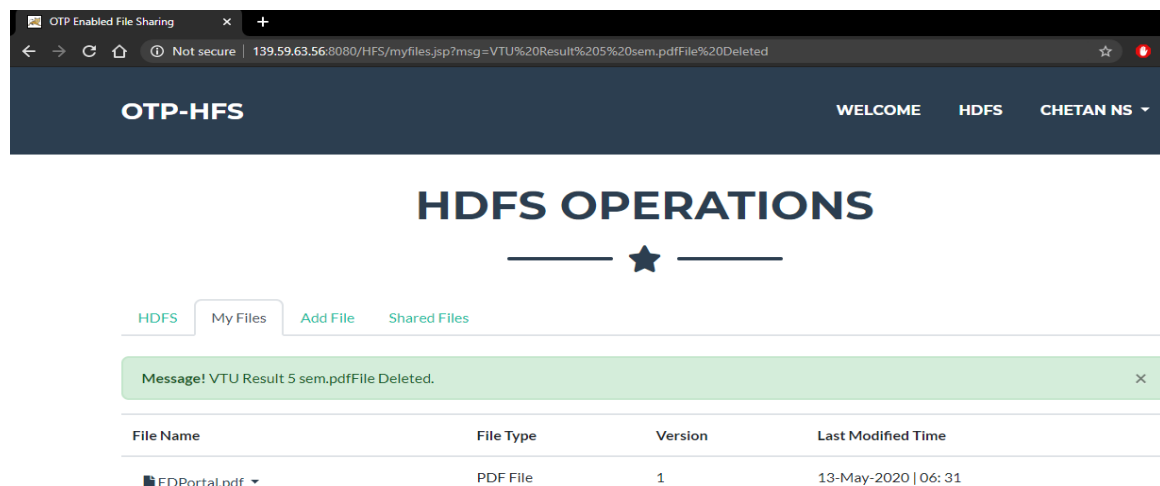


Fig 7.13 Delete File

7.1.14 File sharing

The fig 7.14 shows file sharing compont, the users will also be provided with a provision to share his/her files with other registered users in the portal. This can be done by selecting the file which has to be shared and entering the Sharing component in this module. The user will then be provided with a view listing all the registered users in the portal. The user can select any of the user from the list and grant access to that user on the selected file.

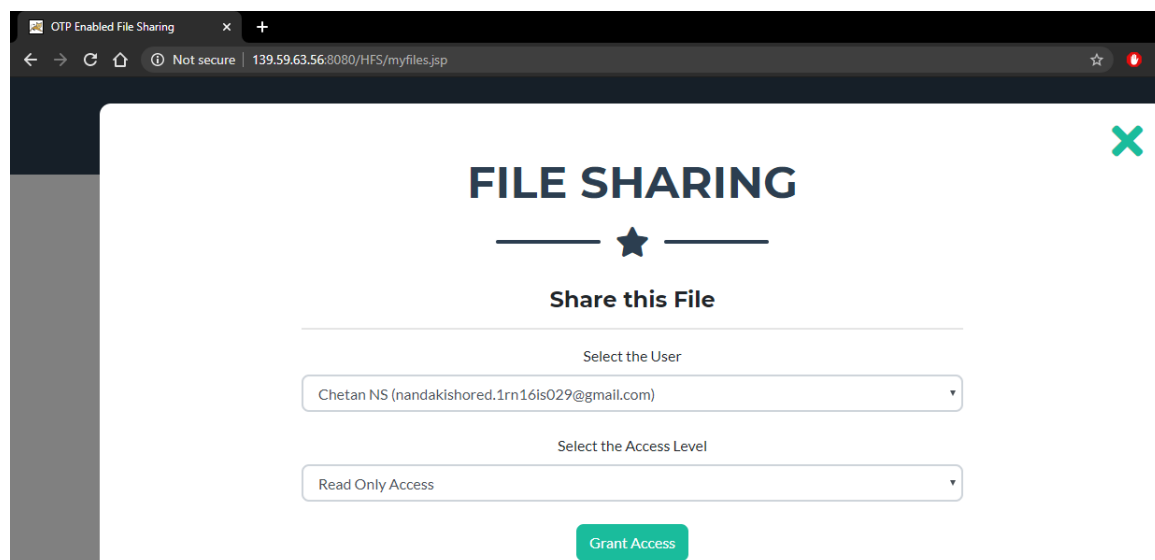


Fig 7.14 File sharing

7.1.15 Change or Revoke access

The fig 7.15 shows that The data owners have the full rights of revoking the access at any point of time or also can change the access rights to it.

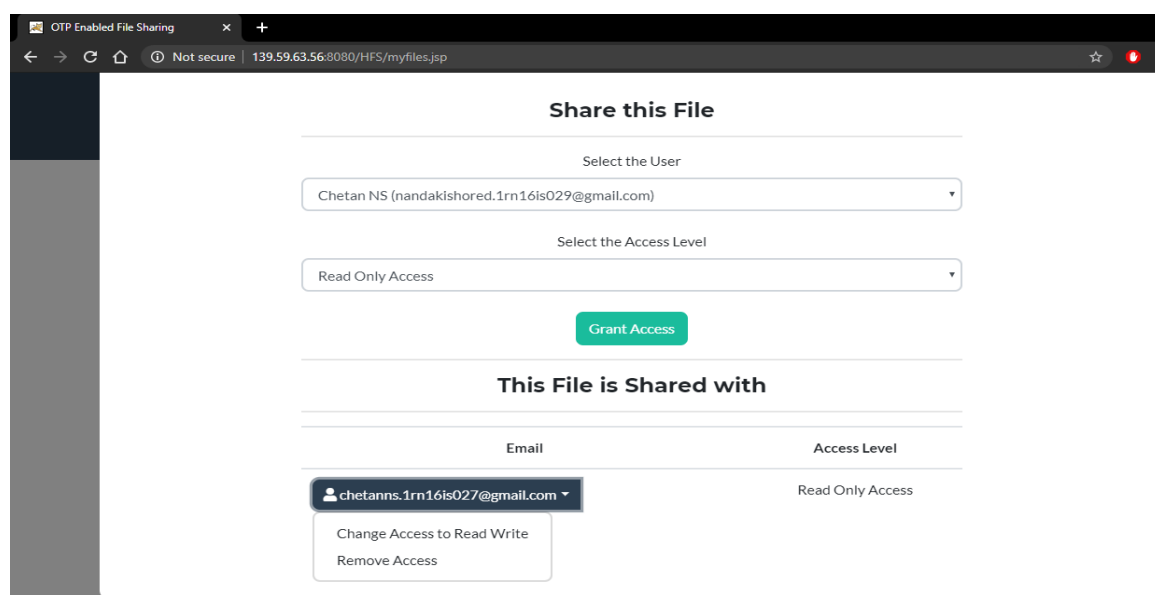


Fig 7.15 Change or Revoke access

7.1.16 Shared file page

The fig 7.16 shows , the user after logging in into the system can perform the file access operations on all the files which have been shared with him/her by other registered user.

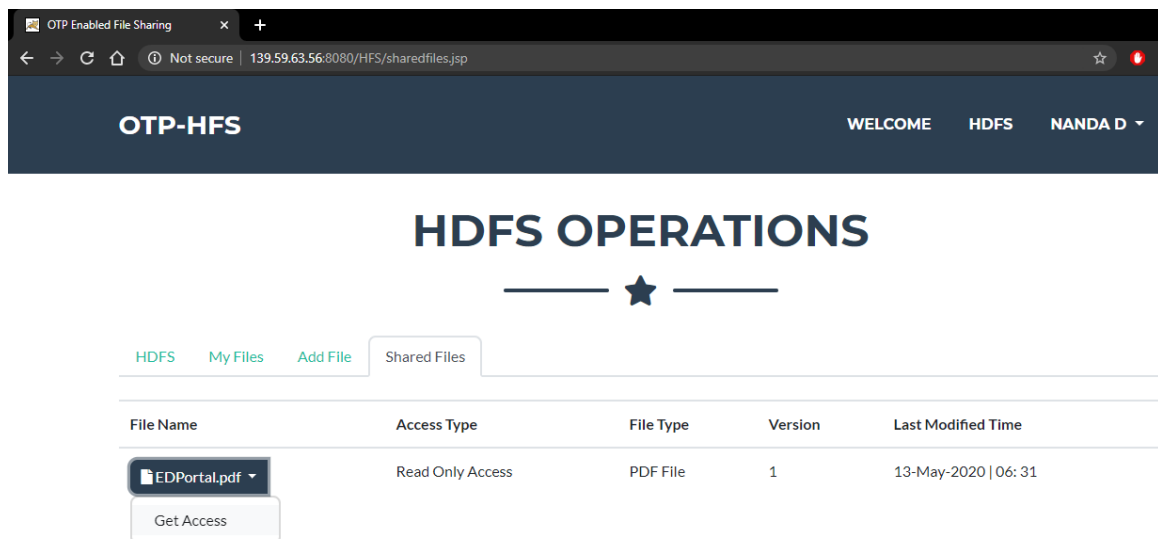


Fig 7.16 Shared file page

7.1.17 Getting Access

The fig 7.17 shows To get the access to the files the user will have to prove his/her identity again by entering the OTP. When the user requests to perform the access operations on the shared files, the system will send an OTP to his/her registered mobile number and also to his/her registered email ID.

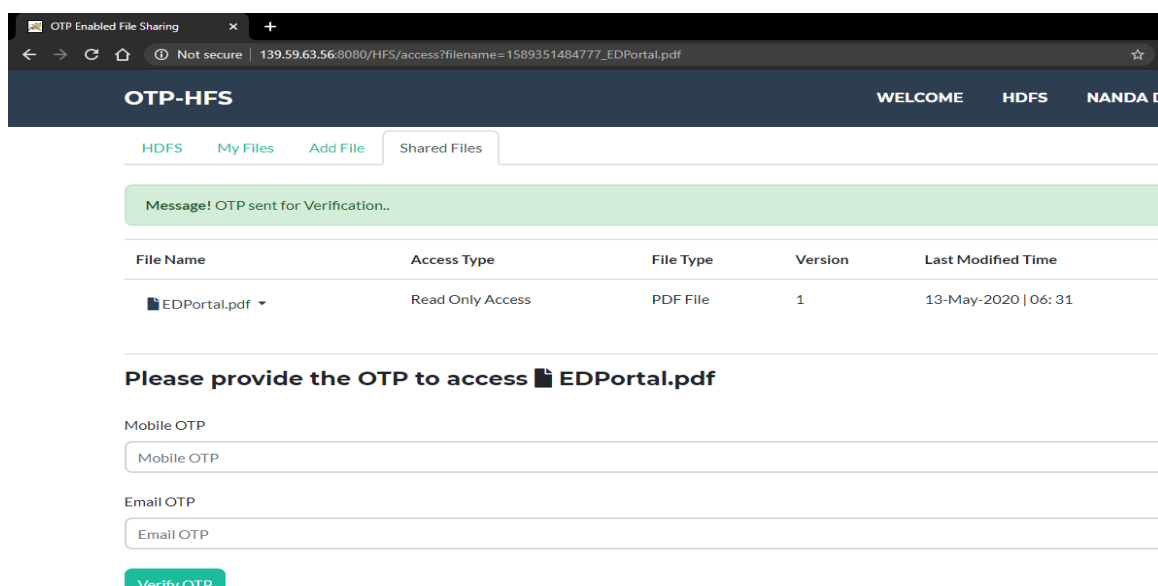


Fig 7.17 Getting Access

7.1.18 Verified OTP

The fig 7.18 shows that The user will have to enter the correct OTP before the system can grant access to him. The user will either be granted with READ ONLY access or the READ WRITE access on the files by the data owners. The system allows the users to access them in accordance with these access rules.

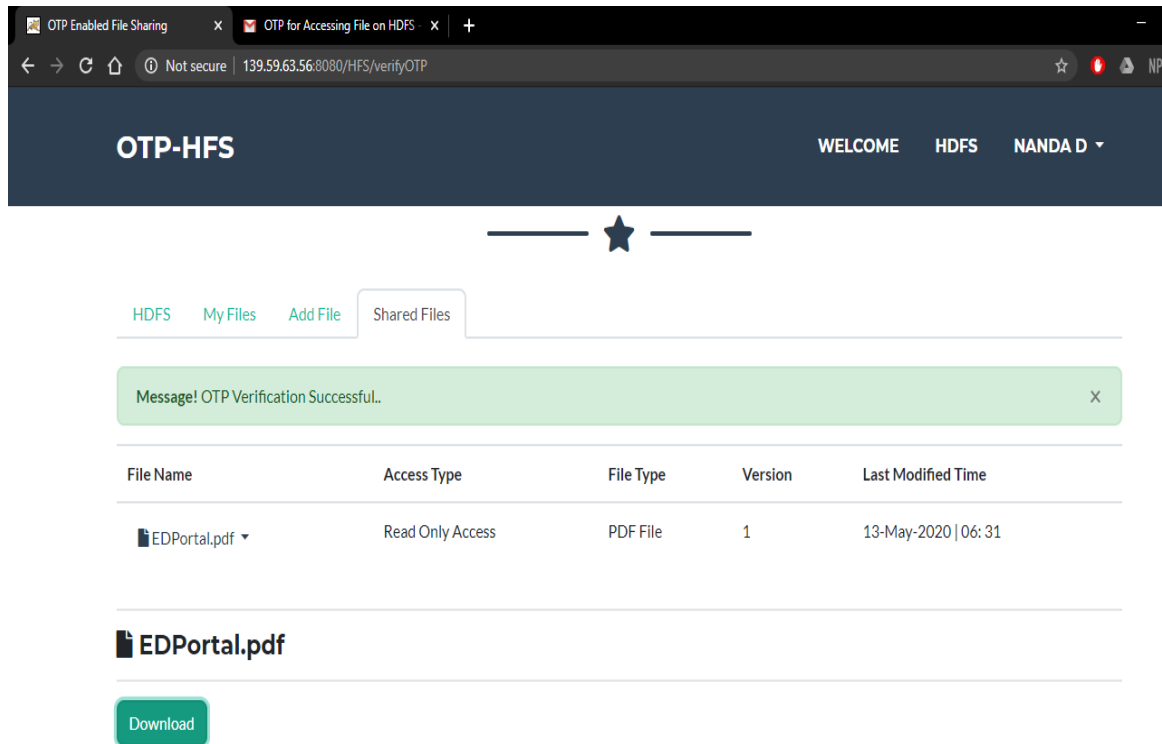


Fig 7.18 Verified OTP

7.1.19 Email OTP

The fig 7.19 shows the email otp sent to the user email Id.

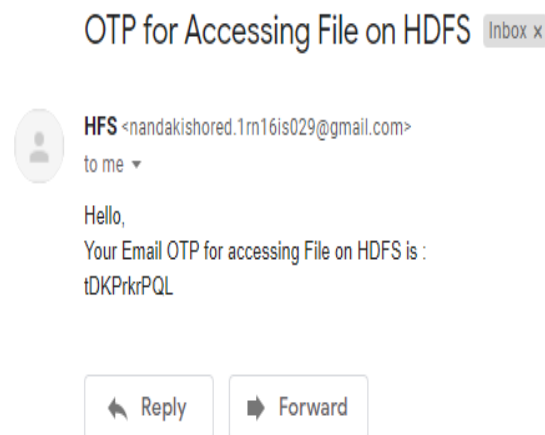


Fig 7.19 Email OTP

7.1.20 Mobile OTP

The fig 7.20 shows the mobile otp sent to the user mobile.

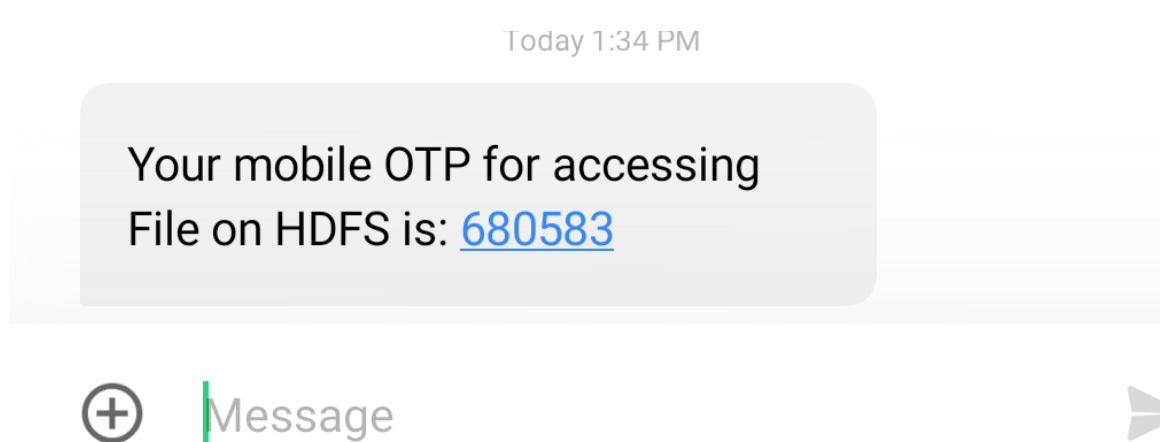


Fig 7.20 Mobile OTP

Chapter 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

While sharing a file the main drawbacks include storage problem and file security. These drawbacks make an application weak and ineffective over the long run. These drawbacks are solved in this project. Hadoop File system is able to store large amounts of data. This data is secured as the concept of OTP is used while accessing the file. The OTP service is put forth in such manner that the receiver has to enter the OTP which is send to him from the sender after verifying the receiver. Once the OTP is entered the file is accessible else the file cannot be opened

8.2 Future work

In Future, the solution can be provided to complete Hadoop Eco system solution addressing the small files problems and other open problems in Hadoop. Another feature like Google o auth authentication can be provided in order to simplify the registration process. The application can be implemented in android and ios for mobile users.

REFERENCES

- [1] Ke Ma, Research and Implementation of Distributed Storage System Based on Big Data, IEEE, 2019
- [2] Yifeng Luo, Jia Shi, Shuigeng Zhou, JeCache: Just-Enough Data Caching for Just-in-Time Prefetching in Big Data Applications, IEEE, 2017
- [3] Amil Ahmad Ilham, Muhammad Niswar, Andi Muhammad Ryanto, Performance Analysis of Big Data Frameworks on Virtualized Clusters, IEEE, 2018
- [4] Valerie Hayot-Sasson, Shawn T Brown and Tristan Glatard, Performance Evaluation of Big Data Processing Strategies for Neuroimaging, IEEE, 2019
- [5] Hanuman Godara M.C. Govil E.S. Pilli, Performance Factor Analysis and Scope of Optimization for Big Data Processing on Cluster, IEEE, 2018
- [6] Sen Pan Lipeng Zhu Junfeng Qiao, An Open Sharing Pattern Design of Massive Power Big Data, IEEE, 2019
- [7] Engin Arslan, Ahmed Alhussen, A Low-Overhead Integrity Verification for Big Data Transfers, IEEE, 2018
- [8] Weijia Xu Ruizhu Huang Yige Wang, Enabling User Driven Big Data Application on Remote Computing Resources, IEEE, 2018
- [9] Hsing-bung (HB) Chen Qiang Guan Song Fu, UNS: A Portable, Mobile, and Exchangeable Namespace for Supporting Fetch-From-Anywhere Big Data Eco-Systems, IEEE, 2018
- [10] Jung-Eun Park, and Young-Hoon Park, Efficient File-Share Reconstruction Scheme for Device Addition/Removal in Personal Area Network, IEEE, 2018
- [11] Junior DONGO Youssef ATIK Charif MAHMOUDI, Distributed File System for NDN: an IoT Application, IEEE, 2018
- [12] Pierre Matri, Philip Carns, Robert Ross, Alexandru Costan, Mar'ia S. P'erez, Gabriel Antoni, SLoG: Large-Scale Logging Middleware for HPC and Big Data Convergence, IEEE, 2018

- [13] Jia-Yow Weng, Chao-Tung Yang Chih-Hung Chang, The Integration of Shared Storages with the CephFS and Rados Gateway for Big Data Accessing, IEEE, 2018
- [14] Lei Guo, Zhaolong Ning , Weigang Hou , Bin Hu, and Pengxing Guo, Quick Answer for Big Data in Sharing Economy: Innovative Computer Architecture Design Facilitating Optimal Service- Demand Matching, IEEE, 2018
- [15] Hadeel Alghamdi Farhana Zulkernine, Leveraging Distributed Big Data Storage Support in CLAAaaS for WINGS Workflow Management System, IEEE, 2017