**DEDUPLICATION IN CLOUD COMPUTING TO**

**IMPROVISE EFFICIENCY TOWARDS POTENTIAL PRACTICAL USAGE**

**A PROJECT REPORT**

***Submitted by***

|  |  |
| --- | --- |
| **VIGNESH S** | **(211417104299)** |
| **SHARAN R** | **(211417104255)** |
| **SHACHIN KARTHIK J** | **(211417104250)** |
|  |  |

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

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE, POONAMALLEE**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MARCH 2021**

**ANNA UNIVERSITY: CHENNAI 600 025**

## BONAFIDE CERTIFICATE

Certified that this project report **“DEDUPLICATION IN CLOUD COMPUTING TO IMPROVISE EFFICIENCY TOWARDS POTENTIAL PRACTICAL USAGE”** is the bonafide work of “**VIGNESH S (211417104299), SHARAN R (211417104255), SHACHIN KARTHIK J (211417104250),**” who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| **Dr. S. MURUGAVALLI, M.E., Ph.D.,**  **HEAD OF DEPARTMENT**      DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123. | **Mr. G. SENTHILKUMAR., M.C.A.,M.Phil.,M.B.A., M.E., (Ph.D),**  **SUPERVISOR**  ASSOCIATE PROFESSOR  DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123 |

Certified that the above candidate were examined in the university project viva-voice held on 26.03.2021

|  |  |
| --- | --- |
| **INTERNAL EXAMINER** | **EXTERNAL EXAMINER** |

**ACKNOWLEDGEMENT**

A project of this magnitude and nature requires kind co-operation and support from many for its successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.**, for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Thiru.C.SAKTHIKUMAR,M.E.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.,**  who helped us in completing the project.

We thank the Head of the CSE Department, **Dr. S. MURUGAVALLI, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank our internal guide **Mrs. ANURADHA, M.E.,**

**Associate Professor** and all the faculty members of the Department of CSE for their valuable guidance and continuous support for the development of our project.

### STUDENT1 (REGISTER NO)

### STUDENT2(REGISTER NO)

### STUDENT 3 (REGISTER NO)

### ABSTRACT

Cloud storage as one of the most important services of cloud computing helps cloud users break the bottleneck of restricted resources and expand their storage without upgrading their devices. In order to guarantee the security and privacy of cloud users, data are always outsourced in an encrypted form. However, encrypted data could incur much waste of cloud storage and complicate data sharing among authorized users. We are still facing challenges on encrypted data storage and management with deduplication. Traditional deduplication schemes always focus on specific application scenarios, in which the deduplication is completely controlled by either data owners or cloud servers.

They cannot flexibly satisfy various demands of data owners according to the level of data sensitivity. In this paper, we propose a heterogeneous data storage management scheme, which flexibly offers both deduplication management and access control at the same time across multiple Cloud Service Providers (CSPs). We evaluate its performance with security analysis, comparison and implementation. The results show its security, effectiveness and efficiency towards potential practical usage.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** | iii |
|  | **LIST OF TABLES** | iv |
|  | **LIST OF FIGURES** | vi |
|  | **LIST OF SYMBOLS, ABBREVIATIONS** | viii |
| **1.** | **INTRODUCTION** | 01 |
|  | 1.1 OVERVIEW | 02 |
|  | 1.2 PROBLEM DEFINITION | 03 |
| **2.** | **LITERATURE SURVEY** | 04 |
| **3.** | **SYSTEM ANALYSIS** | 08 |
|  | 3.1 EXISTING SYSTEM | 09 |
|  | 3.2 PROPOSED SYSTEM | 09 |
|  | 3.3 REQUIREMENT ANALYSIS AND SPECIFICATION | 10 |
|  | 3.3.1 Hardware Requirements | 10 |
|  | 3.3.2 Software Requirements  3.3.3 Technology Used | 10  10 |
| **4.** | **SYSTEM DESIGN** | 11 |
|  | 4.1 UML DIAGRAMS | 12 |
|  | 4.1.1 Use Case Diagram | 12 |
|  | 4.1.2 Package Diagram | 13 |
|  | 4.1.3 Flow Diagram | 14 |
|  | 4.1.4 Sequence Diagram  4.1.5 Data Flow Diagram  4.1.6 Collaboration Diagram | 15  16  17 |
| **5.** | **SYSTEM ARCHITECTURE** | 18 |
|  | 5.1 ARCHITECTURE OVERVIEW | 19 |
|  | 5.2 MODULE DESIGN SPECIFICATION | 20 |
|  | 5.2.1 System Design | 20 |
|  | 5.3 MODULES | 21 |
|  | 5.3.1 Module Explanation | 21 |
| **6.** | **SYSTEM IMPLEMENTATION** | 33 |
|  | 6.1 CODING | 34 |
| **7.** | **SYSTEM TESTING** | 37 |
|  | 7.1 UNIT TESTING | 38 |
|  | 7.2 INTEGRATION TESTING | 38 |
|  | 7.3 TEST CASES & REPORTS / PERFORMANCE ANALYSIS | 39 |
| **8.** | **CONCLUSION** | 41 |
|  | 8.1 CONCLUSION AND FUTURE ENHANCEMENTS  8.1.1 Conclusion  8.1.2 Future Enhancements | 42  42  42 |
|  | **APPENDICES** | 43 |
|  | A.1 SAMPLE SCREENS | 44 |
|  | **REFERENCES** | 50 |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO** |
| 4.1.1 | Use case Diagram | 12 |
| 4.1.2 | Package Diagram | 13 |
| 4.1.3 | Flow Diagram | 14 |
| 4.1.4  4.1.5  4.1.6 | Sequence Diagram  Data Flow Diagram  Collaboration Diagram | 15  16  17 |
| 5.1.1 | Architecture Diagram | 19 |
| 5.1.2 | Block Diagram | 20 |
| 5.2.1 | Heart beat PCB type | 22 |
| 5.2.2 | Temperature Sensor-Lm35 | 24 |
| 5.2.3 | 16x2 LCD Module | 24 |
| 5.2.4 | 16X2 LCD | 25 |
| 5.2.5 | LCD Pin Configuration | 26 |
| 5.2.6 | Proteus 7 | 28 |
| 5.2.7 | MPLAB IDE | 30 |
| 5.3.8 | SIM 900 | 32 |
| 5.2.9 | Block Diagram for Power Supply | 40 |
| 5.2.10 | Circuit Diagram of Regulator Power Supply | 41 |
| 5.2.11 | PIC Controller | 47 |
| 5.2.12 | PIC microcontroller architecture | 48 |
| 7.3.1 | Transmitter Module Screenshot | 62 |
| 7.3.2 | Receiver Module Screenshots | 63 |
| 7.3.3 | Output Screenshot | 63 |
| 8.1.1 | Online Monitoring | 67 |
| 8.1.2 | Device Control | 67 |
|  |  |  |
|  |  |  |

**LIST OF ABBREVIATIONS**

**CSP Cloud Service Providers**

**POF Provable Ownership of File**

**HDFS Hadoop Distributed File System**

**CSV Comma Separated Value**

**ABE Attribute Based Encryption**

**API Application Programming Interface**

**HTTP Hyper Text Transfer Protocol**

**GUI Graphical User Interface**

**JVM Java Virtual Machine**

**JSP Java Server Pages**

**XML Extensible Markup Language**

**HTML Hyper Text Markup Language**

**CHAPTER 1**

**INTRODUCTION**

**CHAPTER 1**

**INTRODUCTION**

* 1. **OVERVIEW**

CLOUD computing allows centralized data storage and online access to computer services or resources. It offers a new way of Information Technology (IT) services by rearranging various resources and providing them to users based on their demands. Cloud computing has greatly pervasive services and become a promising service platform due to a number of desirable properties such as scalability, elasticity, fault-tolerance, and pay-per-use. Dataj8 storage service is one of the most widely consumed cloud services. Cloud users have greatly benefited from cloud storage since they can store huge volume of data without upgrading their devices and access them at any time and in any place. However, cloud data storage offered by Cloud Service Providers (CSPs) still incurs some problems. First of all, various data stored at the cloud may request different ways of protection due to different data sensitivity. The data stored at the cloud include sensitive personal information, publicly shared data, data shared within a group, and so on. Obviously, crucial data should be protected at the cloud to prevent from any access of unauthorized parties. Some unimportant data, however, have no such a requirement. As outsourced data could disclose personal or even sensitive information, data owners sometimes would like to control their data by themselves, while on some occasion, they prefer to delegate their control to a third party since they cannot be always online or have no idea how to perform such a control. How to make cloud data access control adapt to various scenarios and satisfy different user demands becomes a practically important issue. Access control on encrypted data has been widely studied in the literature. However, few of them can flexibly support various requirements on cloud data protection in a uniform way, especially with economic deduplication management. Second, flexible cloud data deduplication with data access control is still an open issue. Duplicated data could be stored at the cloud in an encrypted form by the same or different users, in the same or different CSPs. From the standpoint of compatibility, it is highly expected that data deduplication can cooperate well with data access control. That is the same data (either encrypted or not) are only stored once at the cloud, but can be accessed by different users based on the policies of data owners or data holders (i.e., the eligible data users who hold original data). Although cloud storage space is huge, duplicated data storage could greatly waste networking resources, consume plenty of power energy, increase operation costs, and make data management complicated. Economic storage will greatly benefit CSPs by decreasing their operation costs and reversely benefit cloud users with reduced service fees. Obviously, cloud data deduplication is particularly significant for big data storage and management. However, the literature still lacks studies on flexible cloud data deduplication across multiple CSPs. Existing work cannot offer a generic solution to support both deduplication and access control in a flexible and uniform way over the cloud

* 1. **PROBLEM DEFINITION**

Cloud storage as one of the most important services of cloud computing. Data ownership proof is an essential process of data deduplication, especially for encrypted data. But this scheme does not provide flexible deduplication control across multiple Cloud Service Providers (CSPs). In this paper, we propose a multiple cloud service provider (CSPs) in which the data owner will upload the file and the hash MD5 algorithm is used to check data duplication during data storage at the cloud. CSPs. It can achieve data deduplication and access control with different security requirements. And also they have proposed a scheme called Provable Ownership of the File (POF). The result it is security, effectiveness and efficiency towards data storage management.

**CHAPTER 2**

**LITERATURE SURVEY**

**CHAPTER 2**

**LITERATURE SURVEY**

**Paper 1: ”A cryptographically secure technology is more efficient than the one with partial data”**

The server based on actual possession of the whole data present in the storage system with a cryptographically secure technology is more efficient than the one with partial data. In this technique, the client is required to access any small chunk of original data with dynamical coefficient chosen. The File’s Provable Ownership could be generated by this method with the ability to maintain high probability in detection of the malfunctioning of any client.

**Paper 2: “Client-side deduplication using whole file hashing”**

Hashing process is performed at the client, and connects to any one of Deduplicators according to their loads at that time. The deduplicator then identifies the duplication by comparing with the existing hash values in Metadata Server. In traditional deduplication systems, if it is a new hash value, it will be recorded in metadata server, and the file will be uploaded to File Servers, its logical path will also recorded in metadata server. If it does exist, the number of references for the file will be increased..

**Paper 3: “Deduplication offers great savings to the providers of cloud storage services”**

A simple solution that addresses the security risks of deduplication is limiting the number of uploads a user is permitted per time window. This approach might damage the experience of regular users, who have limited network connectivity, and yet it doesn’t prevent malicious users from writing scripts that repeat their attacks between the specified time windows Encryption with a personal key might prevent deduplication but it’s also susceptible to offline dictionary attacks.

**Paper 4: “Data deduplication is not compatible with the conventional process of converting the data into a secret form”**

Convergent key encryption is considered as the best way to ensure secure data deduplication. But it has been observed that convergent key encryption has various drawbacks. Hence convergent key encryption mechanism should not be used to protect data privacy; the better mechanism is needed to ensure secure and efficient data storage. A novel approach is proposed by assuming that the data requires varying levels of protection based on the popularity of the data item, i.e. when a data item belongs to more than a fixed number (k) of users, we assume that the data item is popular. The data is considered unpopular if this is owned by less than, k users. The popularity of the data item drives it for data deduplication. Data items which require lower levels of security i.e. owned by many people will be encrypted with the convergent key encryption, whereas highly sensitive data items, i.e. owned by very few people will be encrypted with conventional symmetric key encryption. This will introduce many questions like how we know the popularity of the data items. The beauty of this mechanism depends on how securely we will calculate the popularity of the data. Whenever the user wants to upload the data, first he will verify whether the data is popular or not. Because the popular files are encrypted with the convergent key, the user will check for the convergent key encrypted copy of the data stored in the cloud storage

**Paper 5: “A protected, lightweight, strong and effective plan for information interchange among the media clouds and mobile clients”**

The aim of this paper is to help real time formulating with power saving requirement. The purposed methodology is basically a mixture of PUK, SK and PRK and no requirement of consistent association between the clients. In this paper, high proficiency video coding related plan of encryption is discussed which is time effective, less unpredictable as far as calculations. The proposed plan encrypt the secret information in compressed space, not in encoding area and it comprises of different stages, which are information encryption, video encoding and decoding with or without interpreting. The analyzed plan attempts to maintain the visual nature of that video and keep the size of video stream same as before encryption process.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

In existing system, a heterogeneous data storage management scheme, which flexibly offers both deduplication management and access control at the same time across multiple Cloud Service Providers (CSPs). They evaluate its performance with security analysis, comparison and implementation. They use Attribute Based Encryption (ABE) to realize deduplication data access control managed by data owner. This scheme was to solve the problem of access control.

**EXISTING SYSTEM DISADVANTAGES**

* The security analysis and performance is not secure.
* They cannot overcome the issue of duplicated data storage in cloud computing.
* They cannot solve the problem of access control.

**3.2 PROPOSED SYSTEM**

In this work, we proposed to storage across multiple CSP’s and preserve data security by managing deduplication. we also introduced a scheme called Provable Ownership of the File (POF). They enhance user privacy and improve the performance of practical deployment. The random hash code challenge is applied to verify data ownership, which can guarantee that the data holder really have the original data rather than its hash code.

**Advantage:**

* They provide security, effectiveness and efficiency towards potential usage.
* They provide to save the cloud storage across multiple CSP’s and preserve data security in encrypted form.
* They specify a set of attributes to identify user and encrypts based on it.

**3.3 REQUIREMENT ANALYSIS AND SPECIFICATION**

**3.3.1 HARDWARE REQUIREMENTS**

* Hard Disk : 80GB and Above
* RAM : 4GB and Above
* Processor : P IV and Above

**3.3.2 SOFTWARE REQUIREMENTS**

* Windows 7 and above
* JDK 1.8
* J2EE
* Tomcat 7.0
* MySQL

**3.3.3 TECHNOLOGY USED**

* JAVA
* HTML
* CSS
* JAVA SCRIPT

**CHAPTER 4**

**SYSTEM DESIGN**

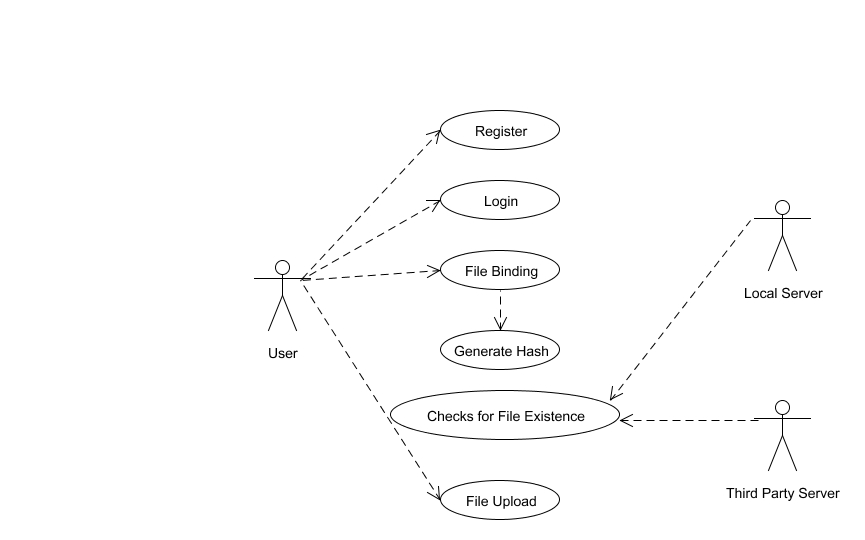
**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 UML DIAGRAMS**

**4.1.1 USE CASE DIAGRAM**

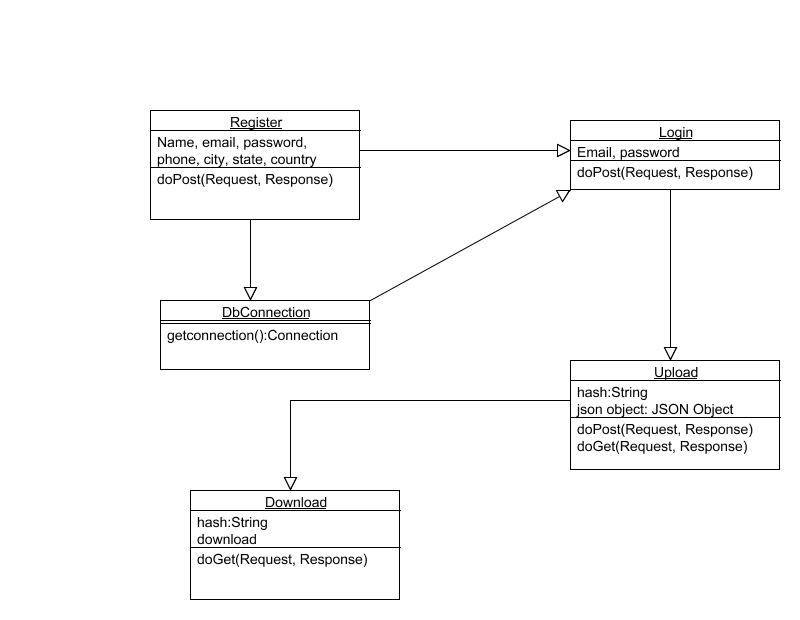
A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases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

****

**Fig 4.1.1. Use case diagram**

**4.1.2 PACKAGE DIAGRAM**

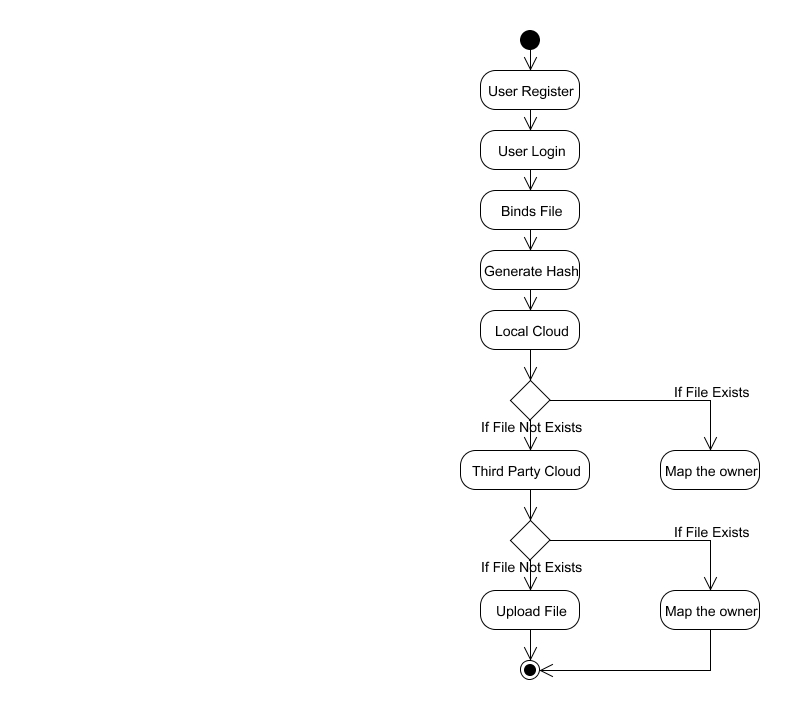
A package diagram, a kind of structural diagram, shows the arrangement and organization of model elements in middle to large scale projects. The package diagram can show both structure and dependencies between sub-systems or modules, showing different views of a system.

****

**Fig 4.1.2. Package diagram**

**4.1.3 FLOW DIAGRAM**

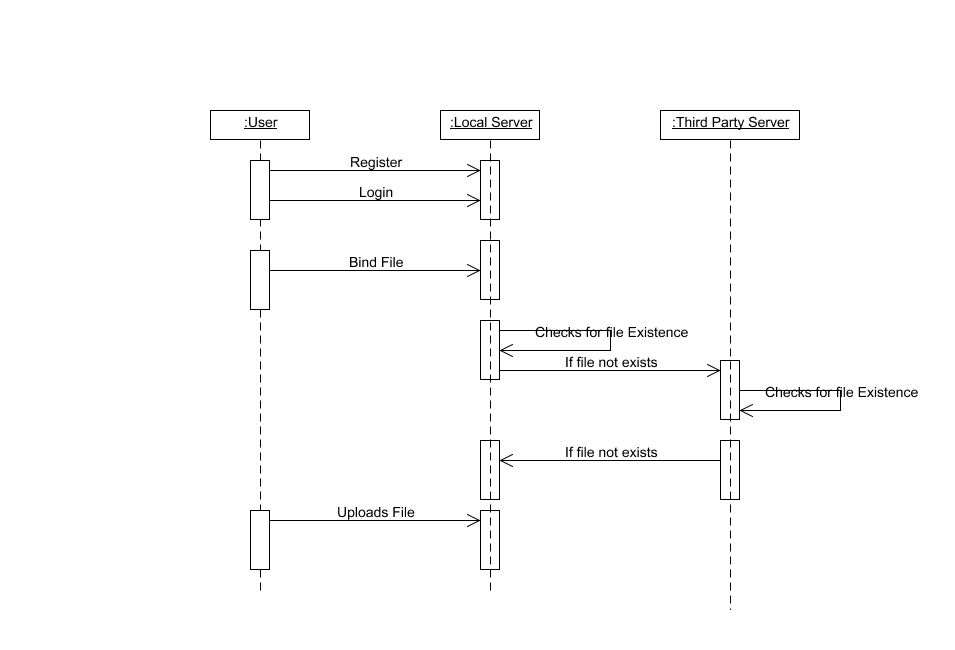
A diagram of the sequence of movements or actions of people or things involved in a complex system or activity.

****

**Fig 4.1.3. Flow diagram**

**4.1.4 SEQUENCE DIAGRAM**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the sequence diagram of the system under development.

****

**Fig 4.1.4. Sequence Diagram**

**4.1.5 DATA FLOW DIAGRAM**

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

**Level 0:**

User

Register

Admin

Verify User

**Level 1:**

Approved User

Login

Binds File

Generate Hash

**Level 2:**

Local Server

Third Party Server

User

File Binding

Generate Hash

Checks for File Existence

**Level 3:**

User

Binds File

If File already exists

Third Party Server

Local Server

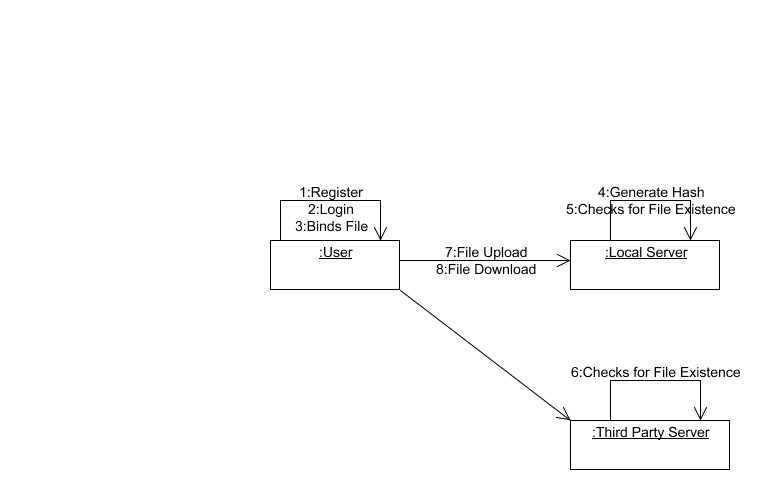
Map the owner

If File not exists

Upload the File

**4.1.6 COLLABORATION DIAGRAM**

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

****

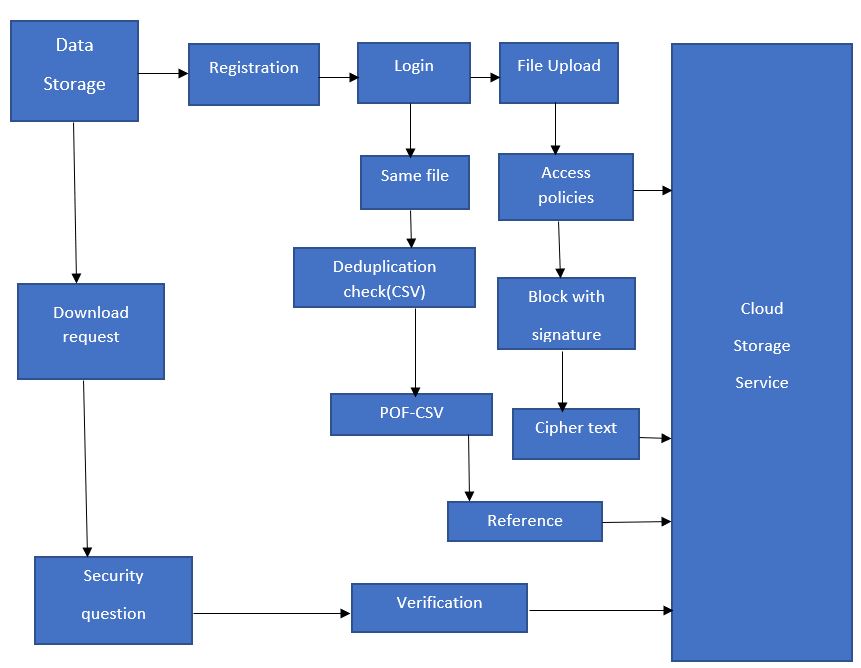
**CHAPTER 5**

**SYSTEM ARCHITECTURE**

**CHAPTER 5**

**SYSTEM ARCHITECTURE**

**5.1 ARCHITECTURE OVERVIEW**

****

**Fig 5.1.1 Architecture Diagram**

**5.2 MODULE DESIGN SPECIFICATION**

**5.2.1 SYSTEM DESIGN**

Owner has an initial level Registration Process in Cloud Service Provider (CSP). The users provide their own personal information for this process. The server in turn stores the information in its database. Then they have the Login process for the further access in cloud service Provider. The data Owner create their account under the Public cloud and upload the file in cloud storage. Here the Provable Ownership of the file (POF) scheme is proposed. While uploading the file by data owner, the hash key is generated based on MD5 algorithm. The hash key is unique for all the upload files. But if the same file is upload by the other data Owner it will not allow the file to upload rather then it will replace the reference id through Mapping of index. It also checks the file for physical present or not by both the data Owner. User will choose the file and uploads to Storage where the HDFS storage system. In the system will generate a signature in particular file and then split into multiple block. Each block will be generating signature with key. In the signature by using MD5 message-digest algorithm is cryptographic hash function producing a 128-bit hash value typically expressed in text format as 32 digit hex value so that files of same are de-duplicated. After that generate convergent keys for each block splitting to store CSV file .like filename, file path, blocks, username, password and block keys. The data owner will download the file from cloud service provider. If they do not find the file then they will request to download the file from different Cloud service provider and also check whether the file is present or not then it gives the response to data owner.

**5.3 MODULES**

* + - Cloud User Authentication.
    - File Upload and Comparison
    - Set Access Policy for File
    - File Download Request and Handling.

**5.3.1 MODULE EXPLANATION:**

**1.Cloud User Authentication**

Owner has an initial level Registration Process in Cloud Service Provider (CSP). The users provide their own personal information for this process. The server in turn stores the information in its database. Then they have the Login process for the further access in cloud service Provider.

**LOGIN**

**REGISTRATION**

**USER/CLIENT**

2. **File Upload and Comparison**

In this module, the data Owner create their account under the Public cloud and upload the file in cloud storage. Here the Provable Ownership of the file (POF) scheme is proposed. While uploading the file by data owner, the hash key is generated based on MD5 algorithm. The hash key is unique for all the upload files. But if the same file is upload by the other data Owner it will not allow the file to upload rather then it will replace the reference id through Mapping of index. It also check the file for physical present or not by both the data Owner.

**CLOUD**

**STORAGE**

**FILE UPLOAD**

**ACCESS POLICIES**

**BLOCK WITH SIGNATURE**

**CIPHER TEXT**

**3. Set Access Policy for File**

In this module User will chooses the file and uploads to Storage where the HDFS storage system. In the system will generate a signature in particular file and then split into multiple block. Each block will be generate signature with key . In the signature by using MD5 message-digest algorithm is cryptographic hash function producing a 128-bit hash value typically expressed in text format as 32 digit hex value so that files of same are de-duplicated. After that generate convergent keys for each blocks splitting to store CSV file .like filename, file path, blocks, username, password and block keys.

**CLOUD**

**STORAGE**

**LOGIN**

**SAME FILE**

**DEDUPLICATION CHECK (CSV)**

**REFERENCE**

**POF-CSV**

**4. File Download Request and Handling**

In this module, the data owner will download the file from cloud service provider. If they do not find the file then they will request to download the file from different Cloud service provider and also check whether the file is present or not then it gives the response to data owner.

**CLOUD**

**STORAGE**

**USER/**

**CLIENT**

**DOWNLOAD**

**REQUEST**

**VERIFICATION**

**SECURITY**

**QUESTION**

**SOFTWARE DESCRIPTION**

**JAVA**

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

**INTRODUCTION TO JAVA**

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (\*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name \*7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

**WORKING OF JAVA**

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of **main,** which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv [] of C), and is declared as a static method.

To output text from the program, we execute the **println** method of **System.out,** which is java’s output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user’s program.

Java consists of two things

* + Programming language
  + Platform

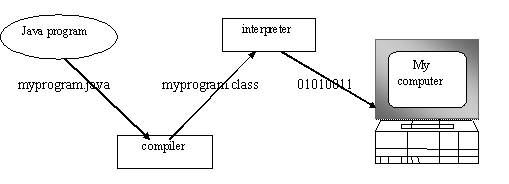
**THE JAVA PROGRAMMING LANGUAGE**

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

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

The code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both co implied and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works:

****

You can think of Java byte codes as the machine code instructions for the **Java Virtual Machine (JVM).** Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make “write once, run anywhere” possible.

You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can e run on Windows NT, Solaris and Macintos

Complier

Interpreter

Interpreter

Interpreter

**  **

**PC-Compatible Sun Ultra Solaris Power Macintosh**

**Windows NT System 8**

**THE JAVA PLATFORM**

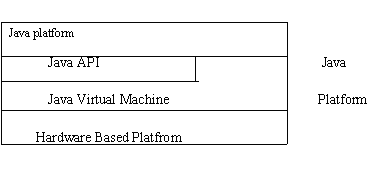
A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components :

* The Java Virtual Machine (JVM)
* The Java Application Programming Interface (Java API)

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

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries **(packages)** of related components. The following figure depicts a Java program, such as an application or applet, that’s running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

****

As a platform-independent environment, Java can be a bit slower than native code. However, smart compliers, weel-tuned interpreters, and just-in-time byte compilers can bring Java’s performance close to that of native code without threatening portability.

**APACHE TOMCAT SERVER**

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

**Environment**  
 Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

**Product features**

Tomcat 3.x (initial release)

* implements the Servlet 2.2 and JSP 1.1 specifications
* servlet reloading
* basic HTTP functionality Tomcat 4.x
* implements the Servlet 2.3 and JSP 1.2 specifications
* servlet container redesigned as Catalina
* Java Management Extensions (JMX), JSP and Struts-based administration
* Tomcat 5.x
* implements the Servlet 2.4 and JSP 2.0 specifications
* reduced garbage collection, improved performance and scalability
* native Windows and Unix wrappers for platform integration
* faster JSP paring

**History** Tomcat started off as a servlet specification implementation by James Duncan Davidson, a software architect at Sun. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation.

Davidson had initially hoped that the project would become open-sourced and, since most open-source projects had O'Reilly books associated with them featuring an animal on the cover, he wanted to name the project after an animal. He came up with Tomcat since he reasoned the animal represented something that could take care of and fend for itself. His wish to see an animal cover eventually came true when O'Reilly published their Tomcat book with a tomcat on the cover.

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6.1 CODING**

**MD5 (Message digest)**

import java.io.UnsupportedEncodingException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

public class MD5 {

public String send(String data, String data1) {

Mac sha512\_HMAC = null;

String result = null;

String key1 = data;

String key = data1;

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*>" + key);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*>" + key1);

try {

byte[] byteKey = key.getBytes("UTF-8");

final String HMAC\_SHA512 = "HmacSHA512";

sha512\_HMAC = Mac.getInstance(HMAC\_SHA512);

SecretKeySpec keySpec = new SecretKeySpec(byteKey, HMAC\_SHA512);

sha512\_HMAC.init(keySpec);

byte[] mac\_data = sha512\_HMAC.

doFinal(key1.getBytes("UTF-8"));

//result = Base64.encode(mac\_data);

result = bytesToHex(mac\_data);

System.out.println(result);

//CarController car=new CarController();

// car.send(result);

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (NoSuchAlgorithmException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (InvalidKeyException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

System.out.println("Done");

}

return result;

}

public static String bytesToHex(byte[] bytes) {

final char[] hexArray = "0123456789ABCDEF".toCharArray();

char[] hexChars = new char[bytes.length \* 2];

for (int j = 0; j < bytes.length; j++) {

int v = bytes[j] & 0xFF;

hexChars[j \* 2] = hexArray[v >>> 4];

hexChars[j \* 2 + 1] = hexArray[v & 0x0F];

}

return new String(hexChars);

}

}

**CHAPTER 7**

**SYSTEM TESTING**

**CHAPTER 7**

**SYSTEM TESTING**

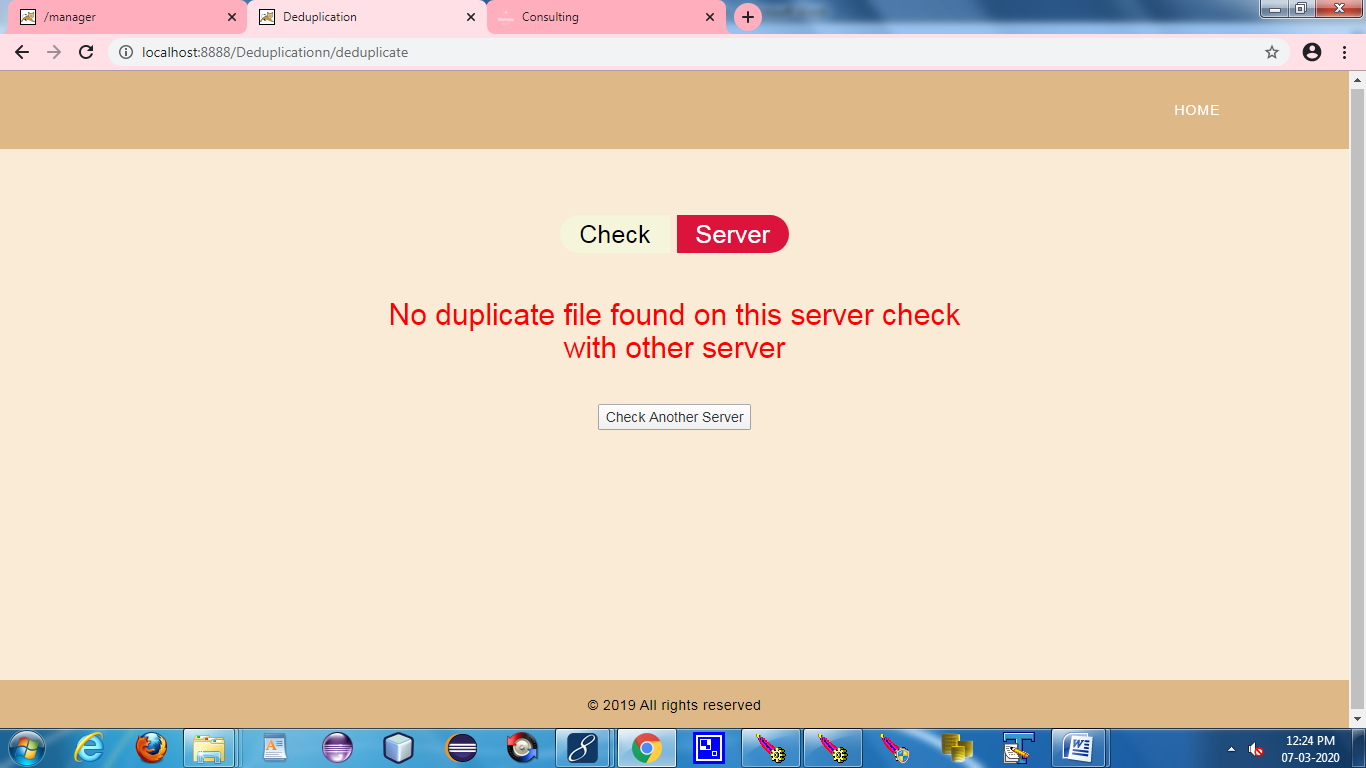
**7.1 UNIT TESTING**

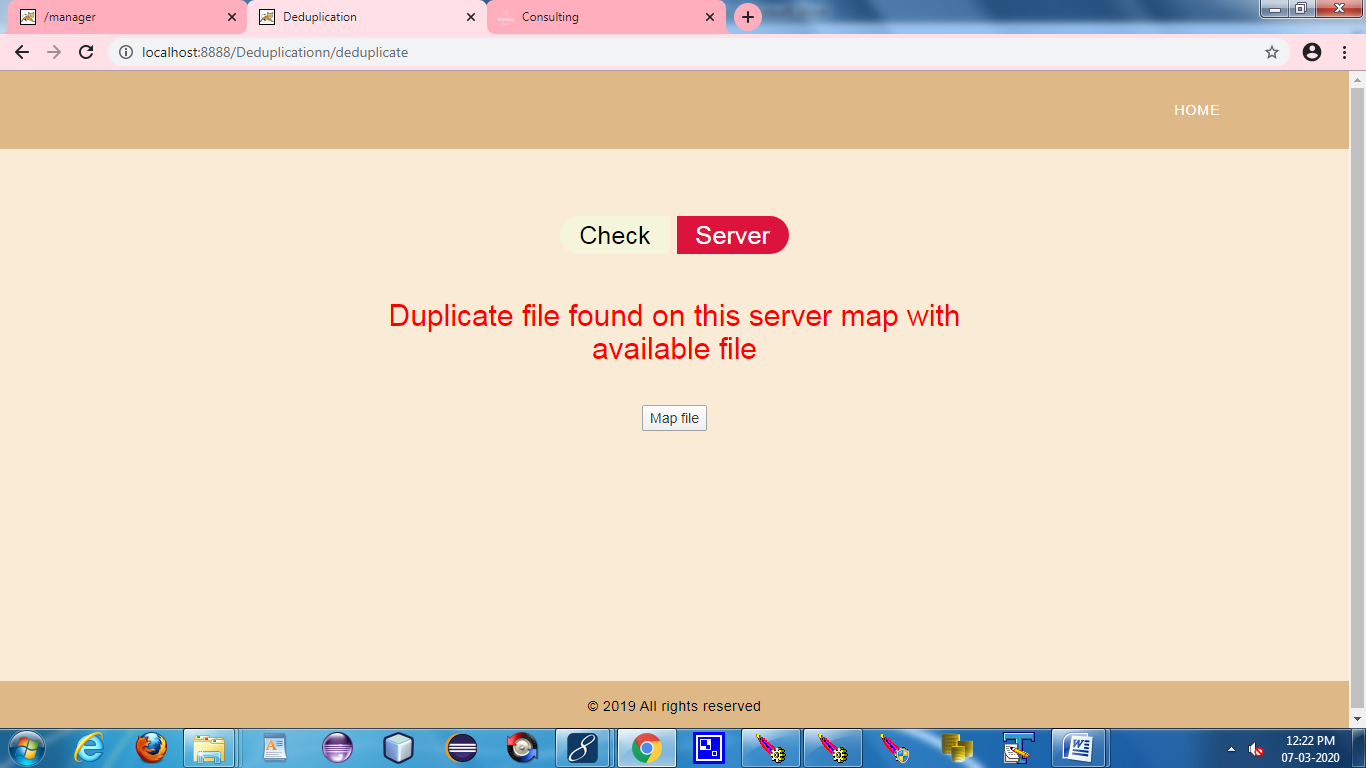
**Unit testing**is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object. In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a White Box testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing.

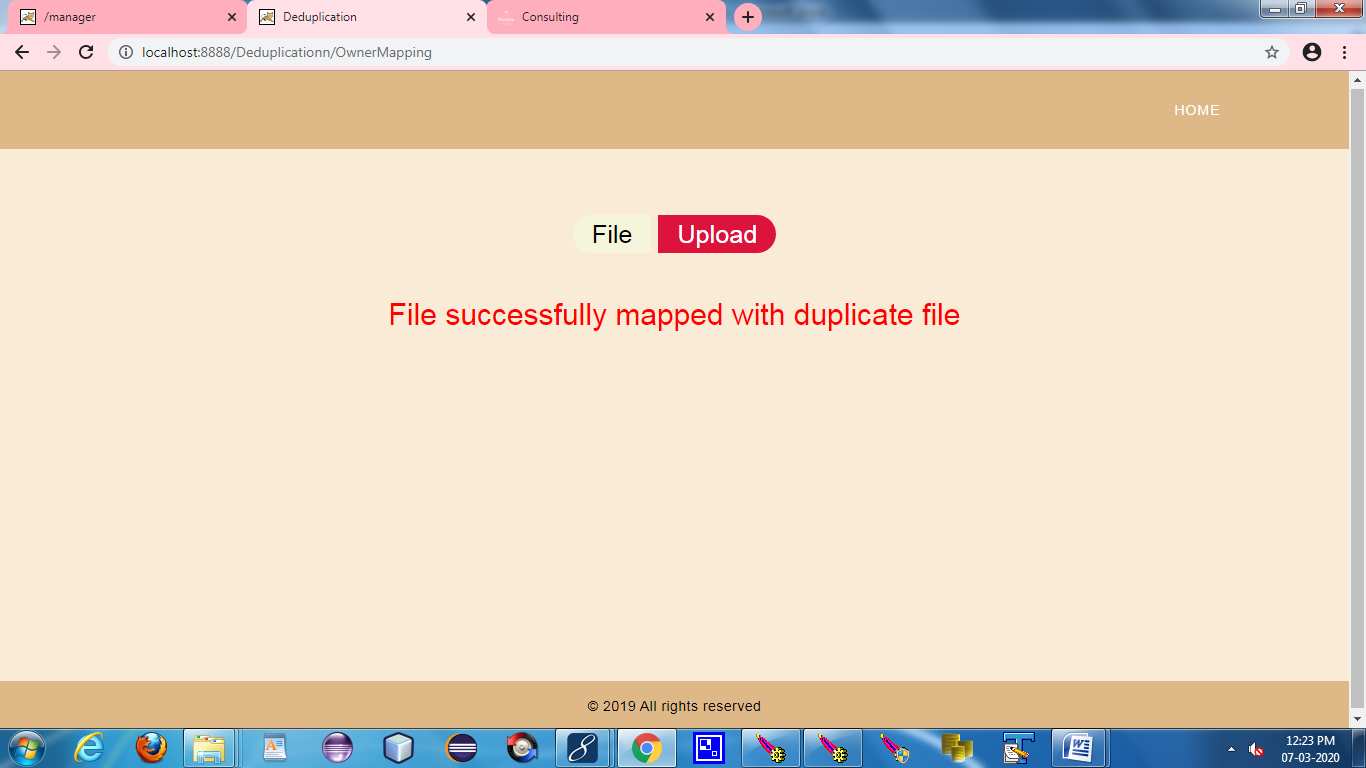
**7.2 INTEGRATION TESTING**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

**7.3 TEST CASES & REPORTS / PERFORMANCE ANALYSIS**







**CHAPTER 8**

**CONCLUSION**

**CHAPTER 8**

**CONCLUSION**

**8.1 CONCLUSION AND FUTURE ENHANCEMENTS**

**8.1.1 CONCLUSION**

The need to provide broadband wireless communications for underwater applications will be increasing in the forthcoming years. Underwater communication provides great potential to augment traditional acoustic communication due to its high data rates, low latency, less power consumption and smaller packaging. We conclude that Underwater communication are the robust and feasible carrier in today’s scenario but with rapid technological development and active ongoing research in UOWC, this technology will be more promising with game-changing potentials in the near future.

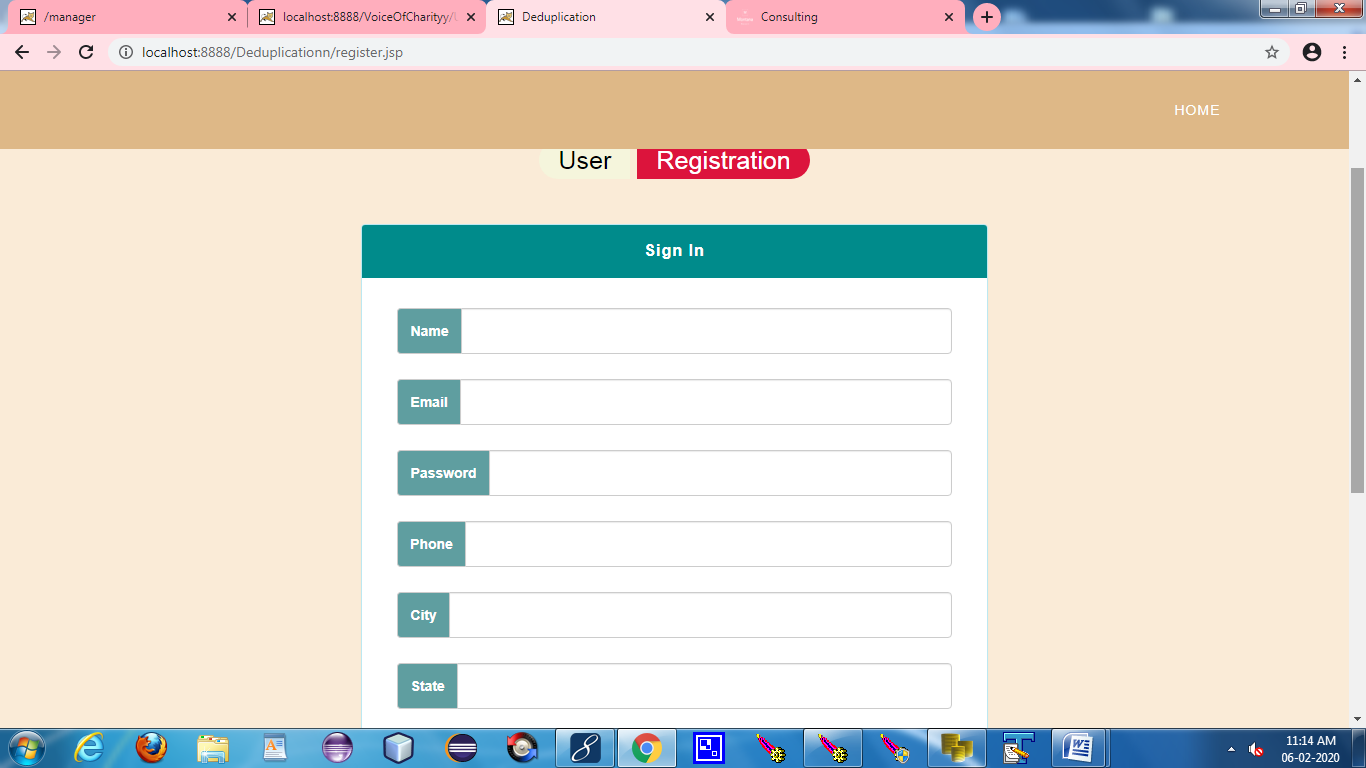
* + 1. **FUTURE ENHANCEMENTS**

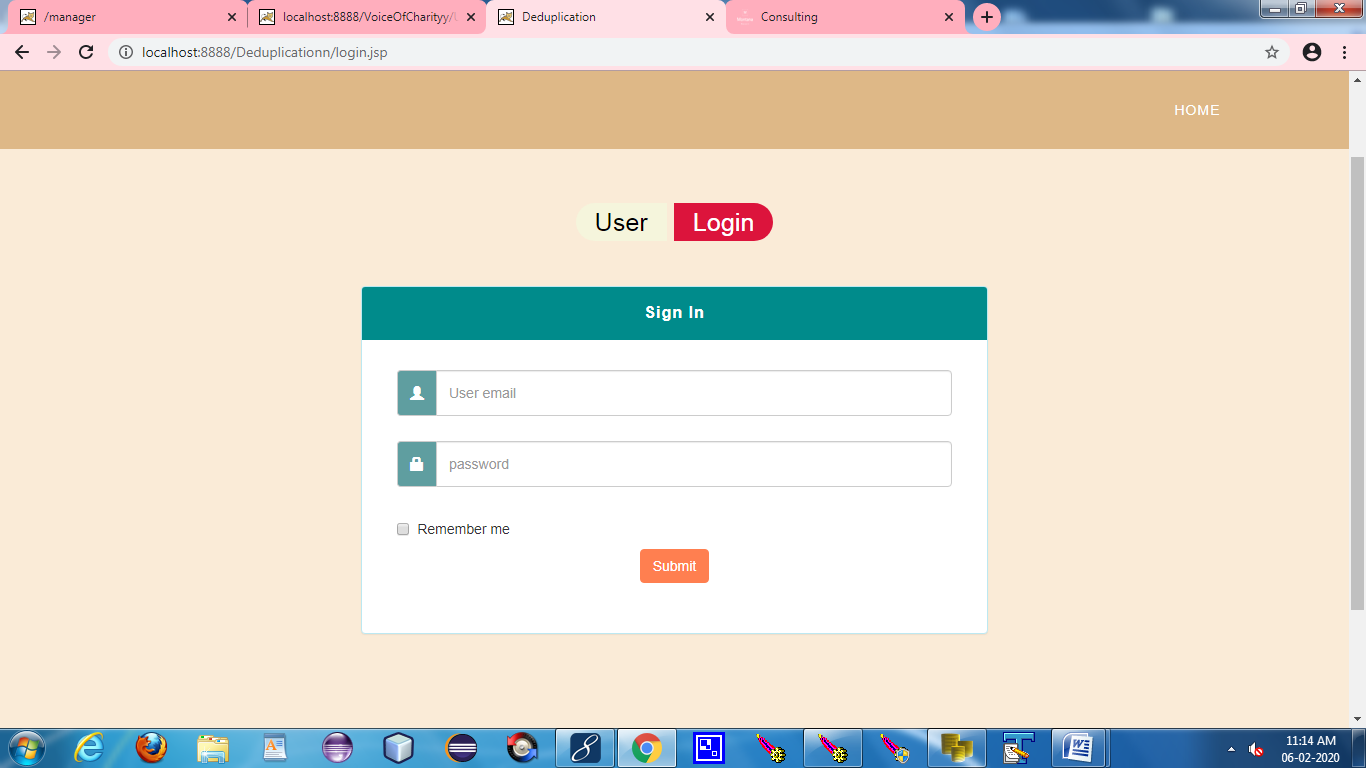
In addition, we will conduct game theoretical analysis to further prove the rationality and security of the proposed scheme

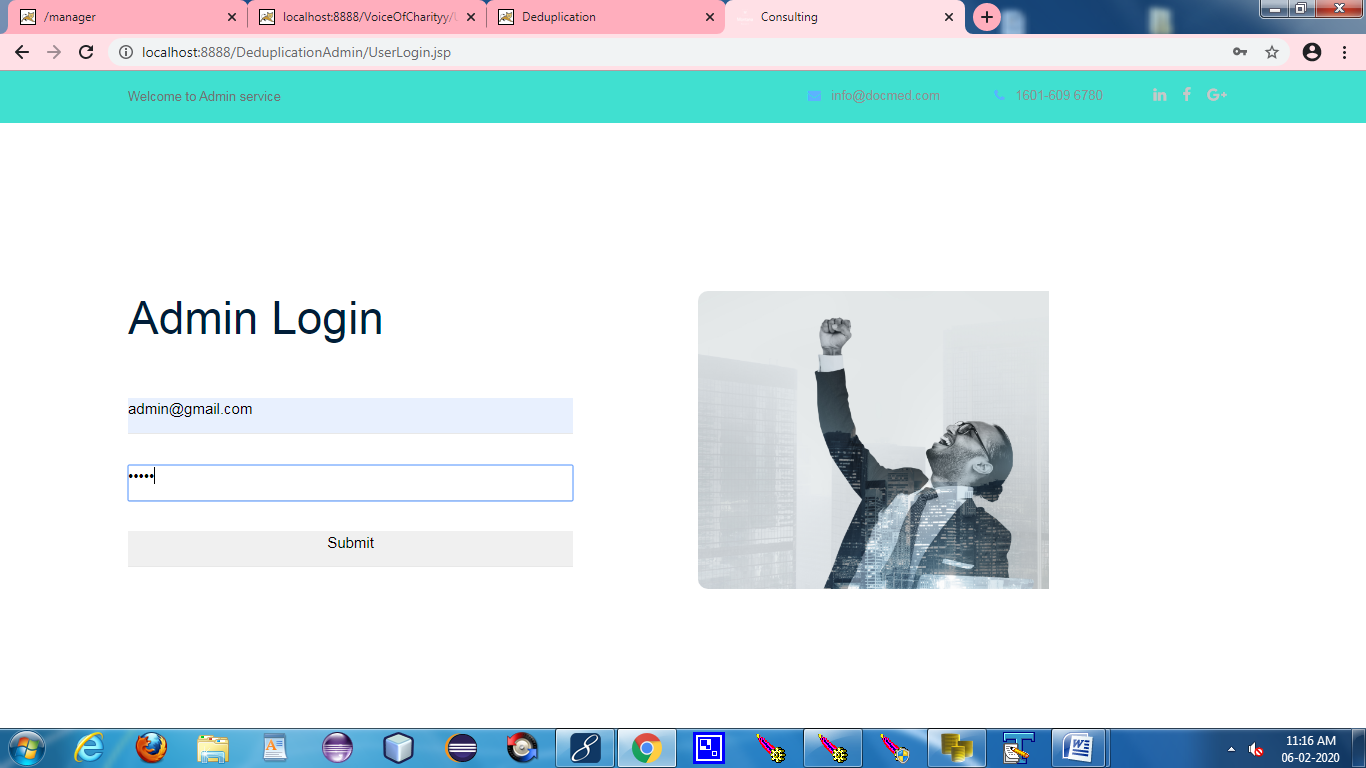
**APPENDICES**

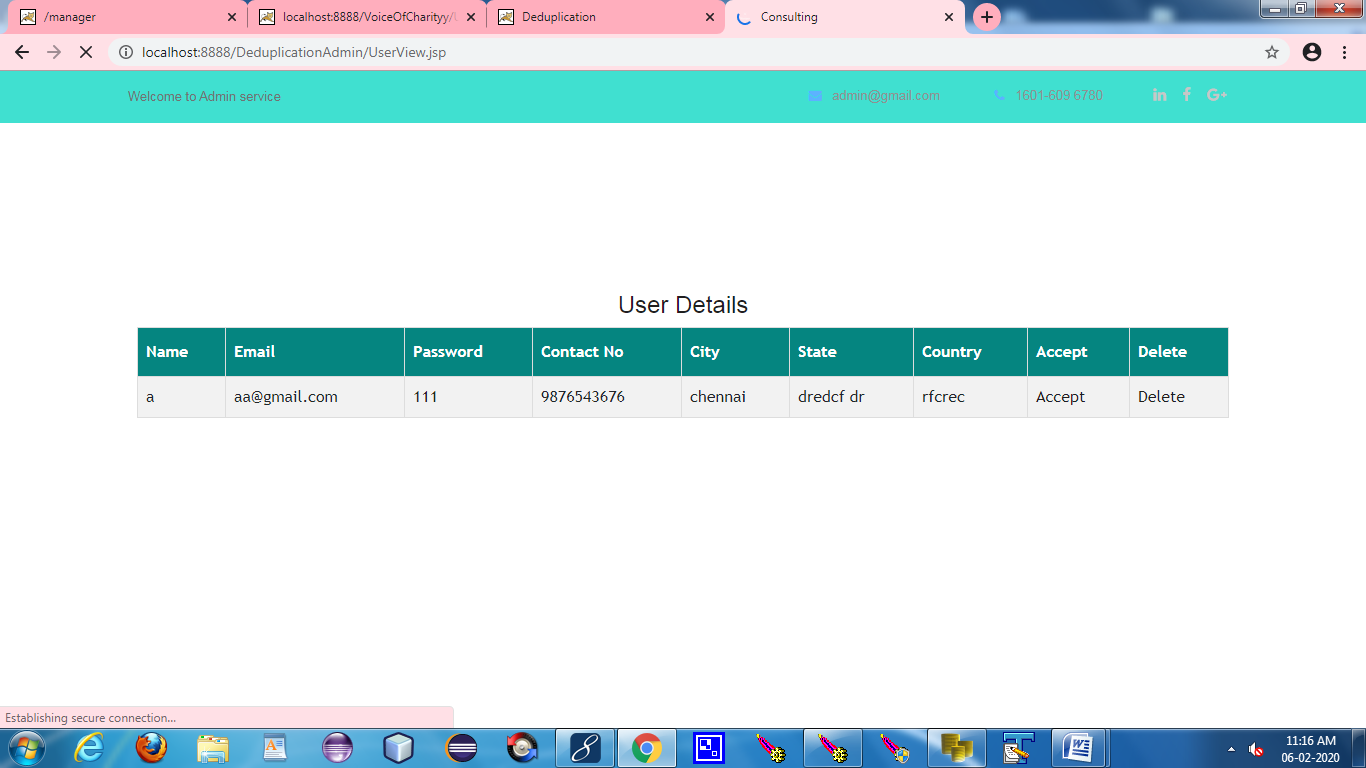
**APPENDICES**

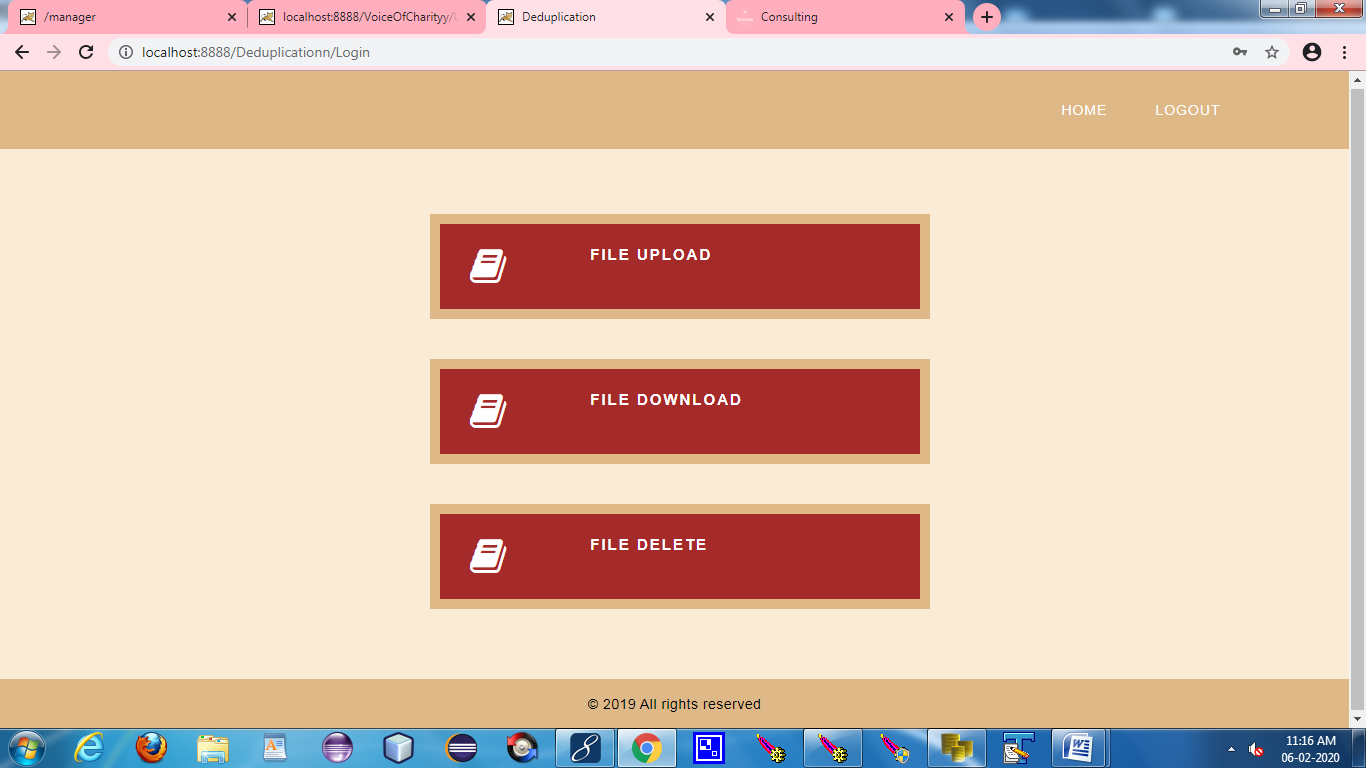
**A.1 SAMPLE SCREENS**

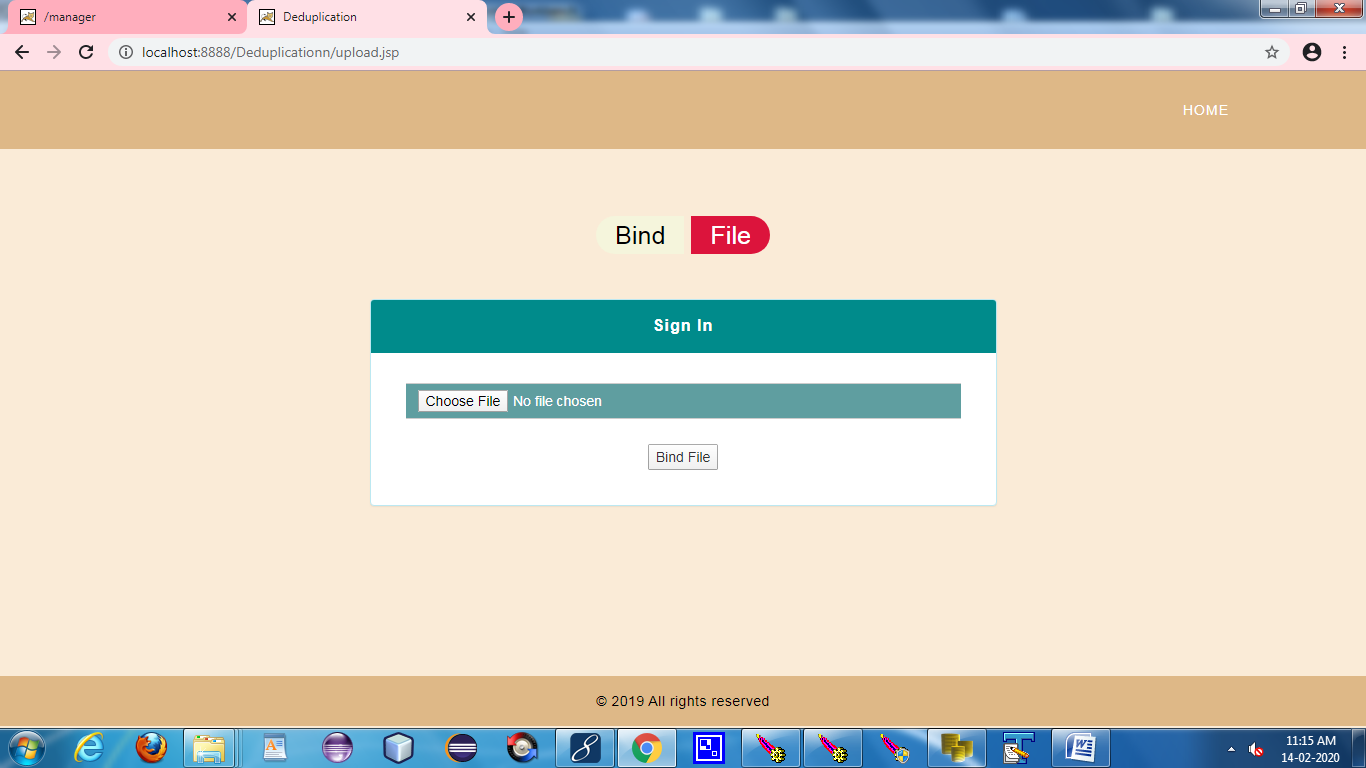


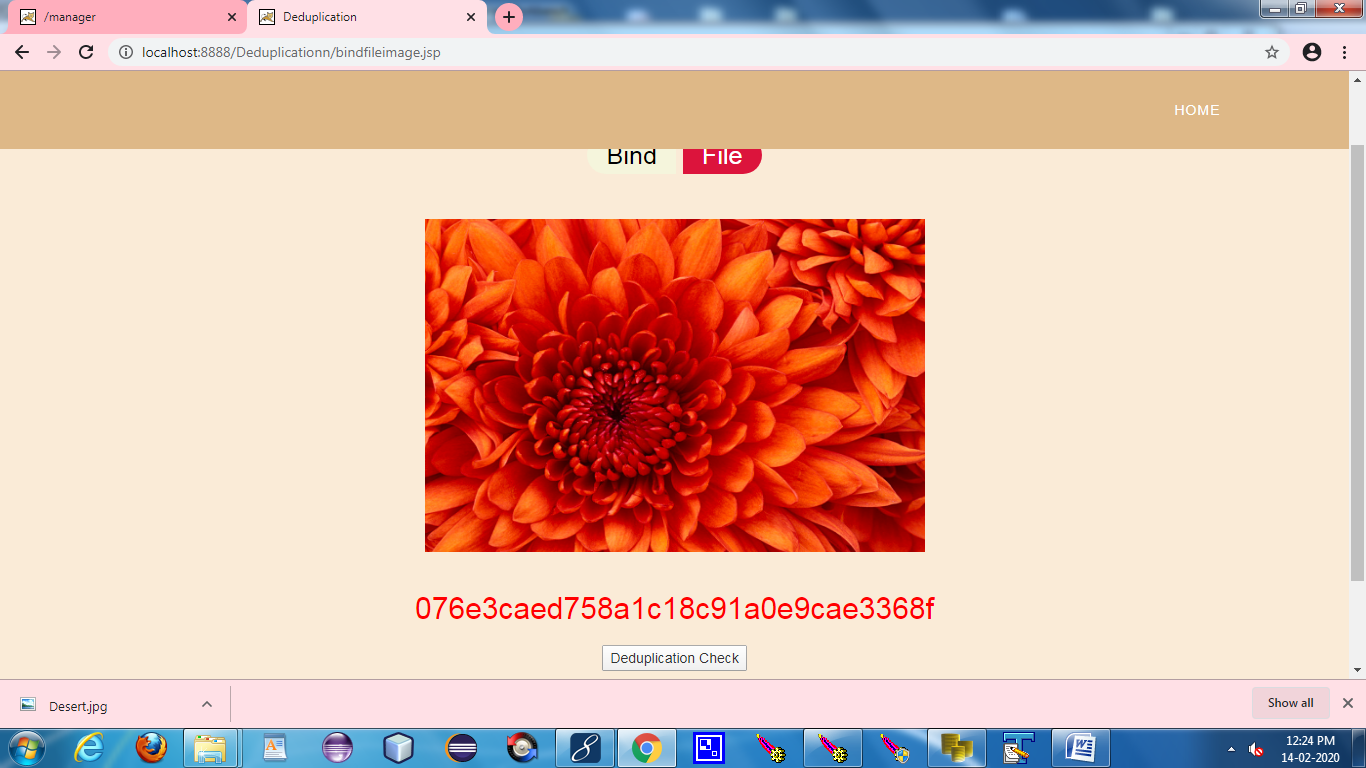


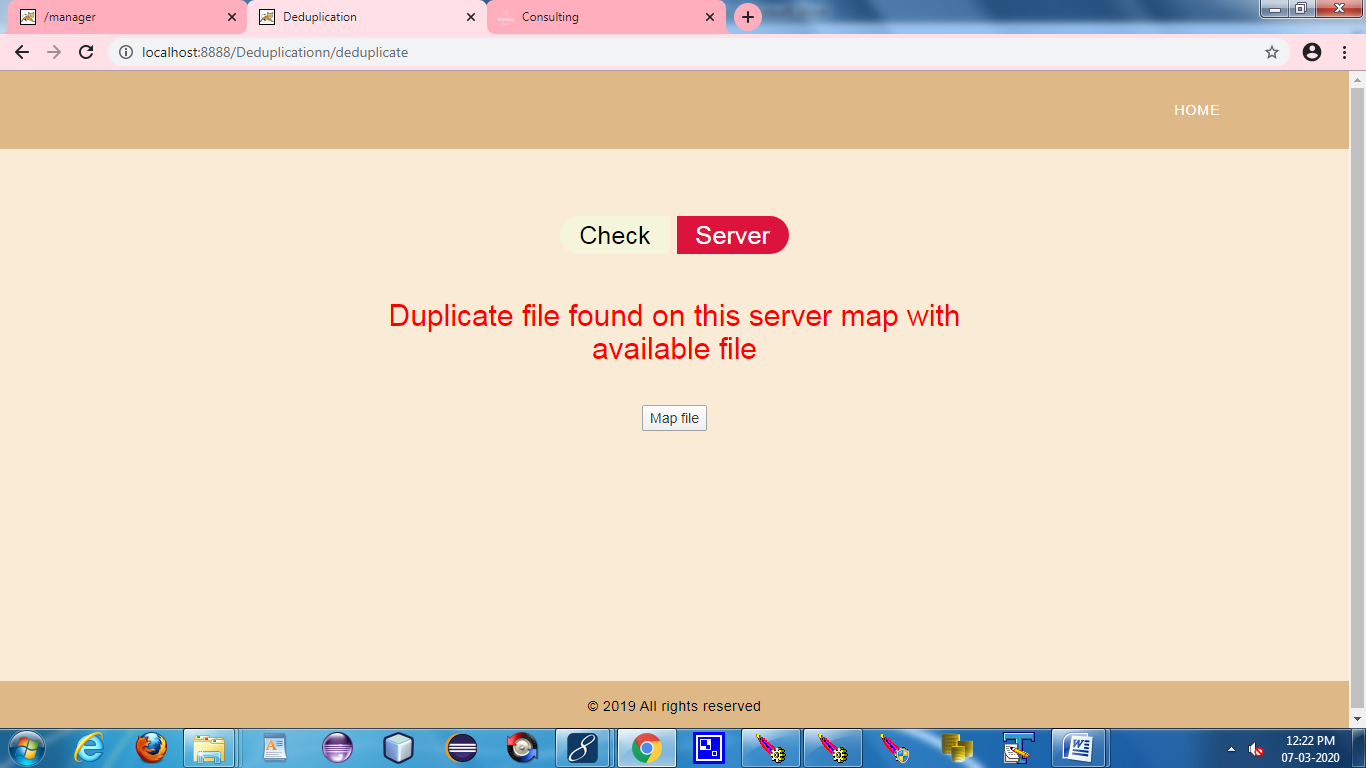


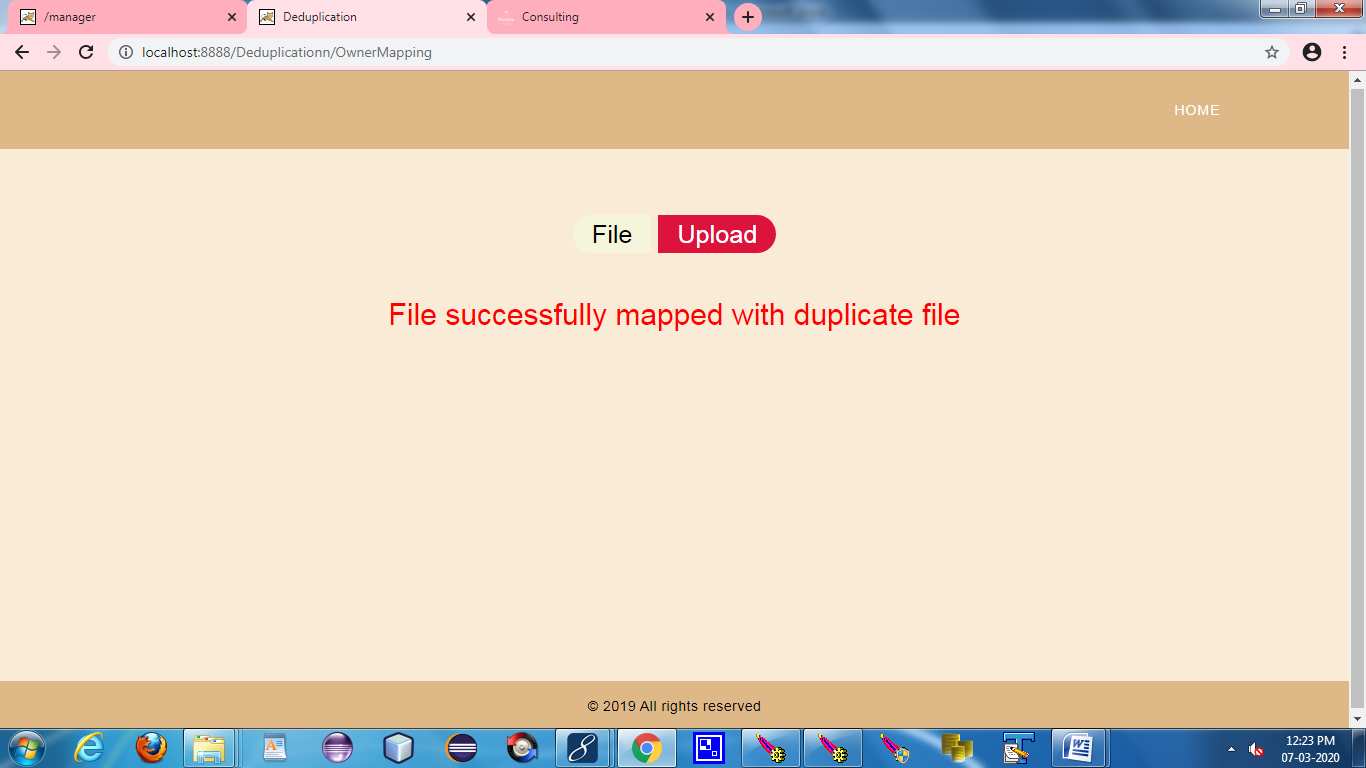
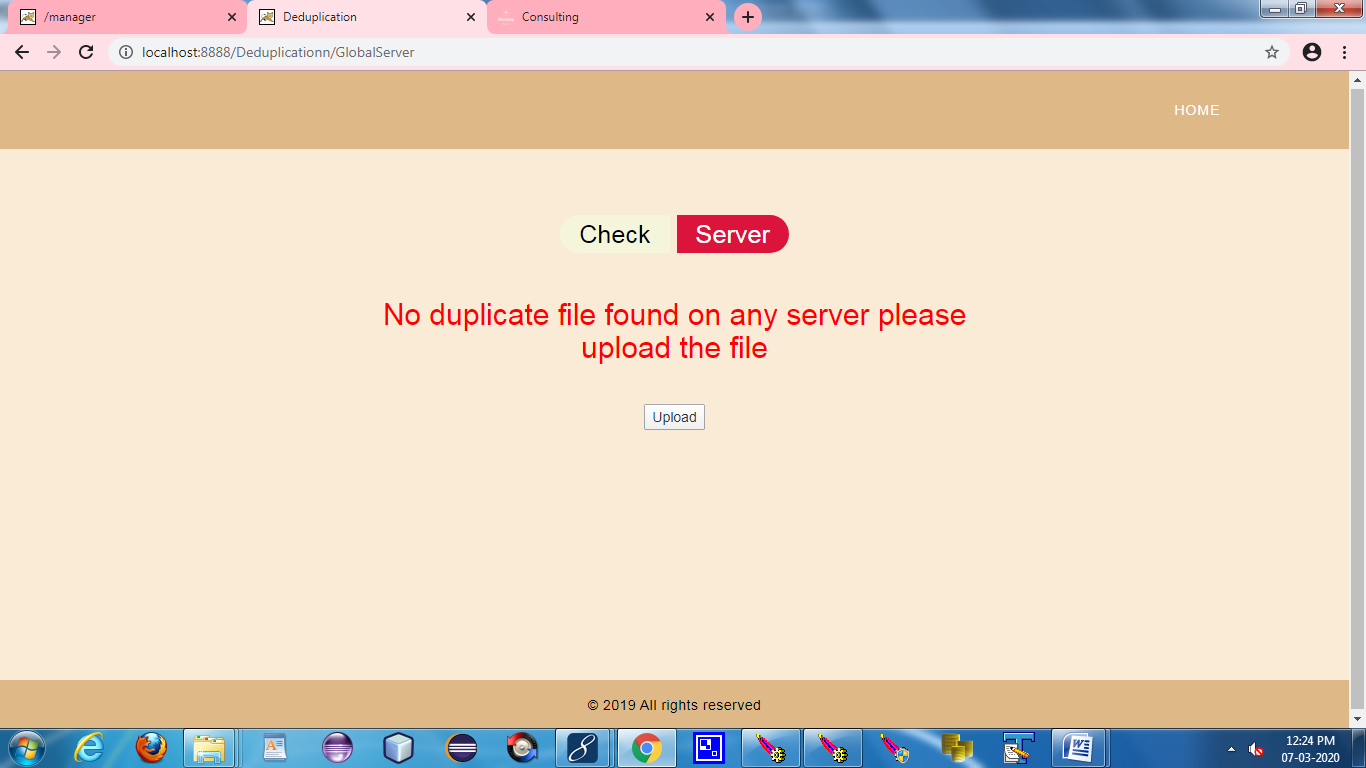


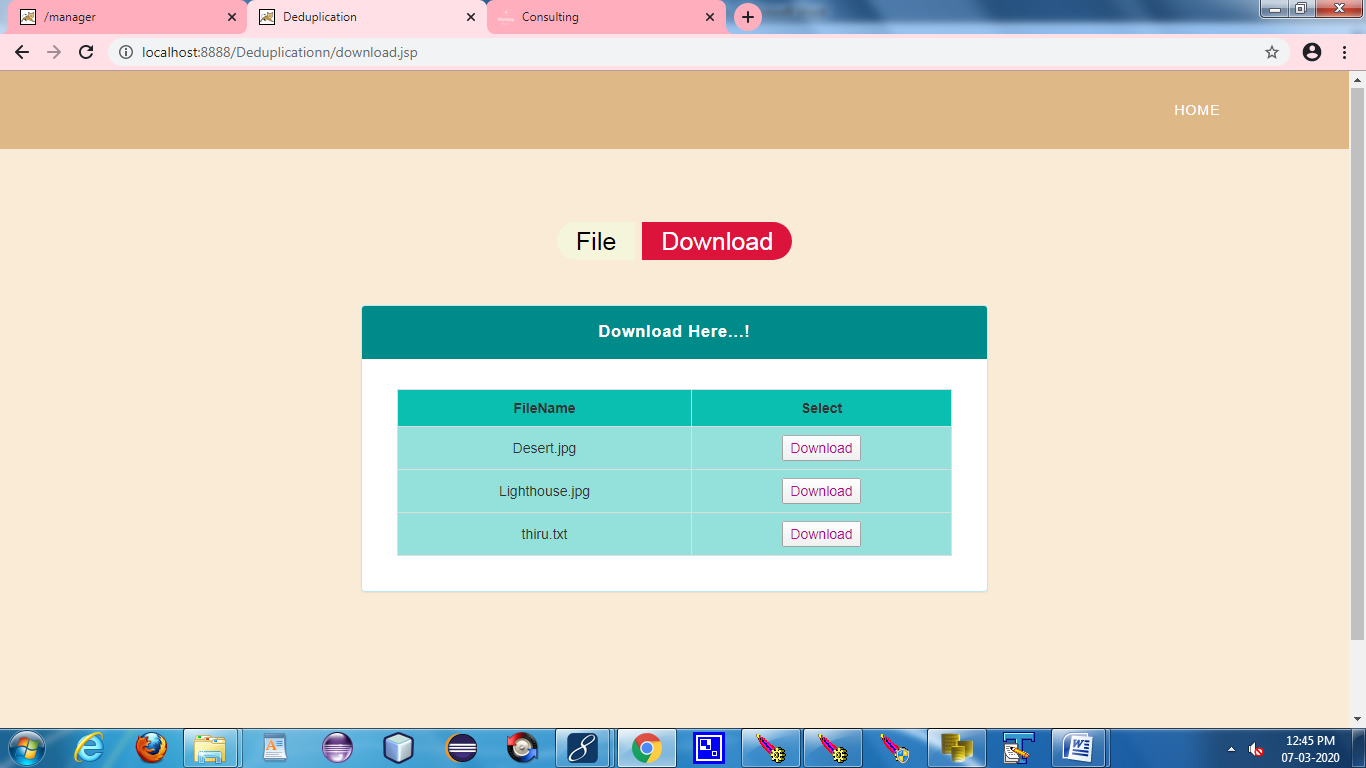


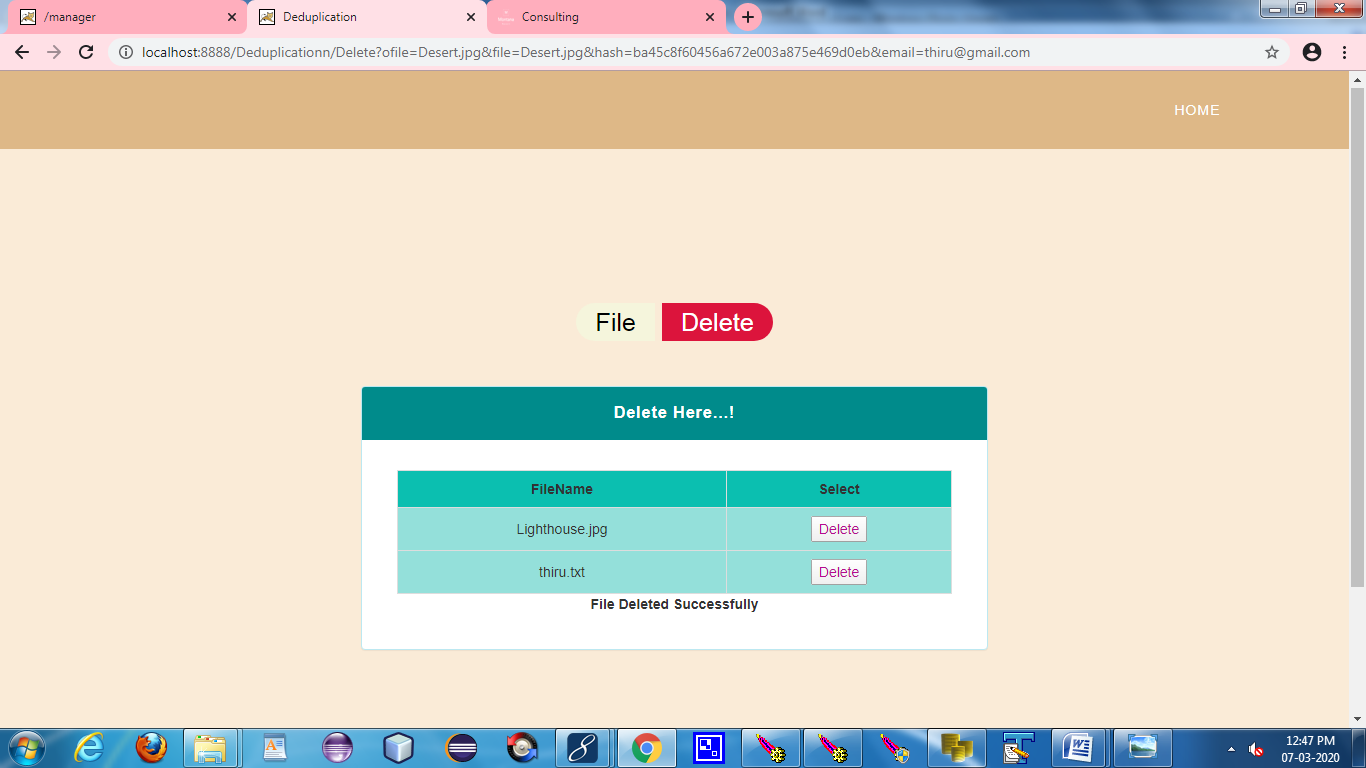












**REFERENCES**

**REFERENCES**

[1] R. Chow, et al., “Controlling data in the cloud: outsourcing computation without outsourcing control,” in Proc. ACM Workshop Cloud Comput. Secur., 2009, pp. 85–90.

[2] S. Kamara, and K. Lauter, “Cryptographic cloud storage,” Financ. Crypto. Data Secur., 2010, pp. 136–149.

[3] Q. Liu, C. C. Tan, J. Wu, and G. Wang, “Efficient information retrieval for ranked queries in cost-effective cloud environments,” in Proc. IEEE INFOCOM, 2012, pp. 2581–2585.

[4] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, “Plutus: scalable secure file sharing on untrusted storage,” in Proc. USENIX Conf. File Storage Technol., 2003, pp. 29–42.

[5] E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh, “SiRiUS: securing remote untrusted storage,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2003, pp. 131–145.

[6] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute- based encryption,” in Proc. IEEE Symp. Secur. Privacy, 2007, pp. 321–334.

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proc. 13th ACM Comput. Commun. Secur., 2006, pp. 89–98.

[8] S. Muller, S. Katzenbeisser, and C. Eckert, “Distributed attributebased encryption,” in Proc. 11th Annu. Int. Conf. Inf. Secur. Crypto., 2008, pp. 20–36.

[9] A. Sahai, and B.Waters, “Fuzzy identity-based encryption,” in Proc. 24th Int. Conf. Theory App. Cryptographic Tech., 2005, pp. 457–473.

[10] S. C. Yu, C. Wang, K. Ren, and W. J. Lou, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” in Proc. IEEE INFOCOM, 2010, pp. 534–542.