

TRUSTWORTHY KEYWORD SEARCH OVER ENCRYPTED DATA WITH TWO SIDE MULTI ACCESS ENVIRONMENT

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Engineering degree in Computer Science and
Engineering by

MANUKONDATHRIVENUBABU (Roll. No. 17SCS0263)

ATLURI. ANIL KUMAR REDDY (Roll. No. 17SCS5023)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

MARCH 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MANUKONDA THRIVENUBABU (Roll. No. 17SCS0263)** and **ATLURI. ANIL KUMAR REDDY (Roll. No. 17SCS5023)** who carried out the project **entitled “TRUSTWORTHY KEYWORD SEARCH OVER ENCRYPTED DATA WITH TWO SIDE MULTI ACCESS ENVIRONMENT”** under Our supervision from January 2021 to March 2021.

INTERNAL GUIDE
Dr. M. SELVI., B.E., M.TECH., (Ph. D).,

HEAD OF THE DEPARTMENT
Dr.S.VIGNESHWARI,M.E.,Ph.D
Dr. L. Lakshmanan, M.E.,Ph.D

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **MANUKONDA THRIVENUBABU (Roll. No. 17SCS0263)** and **ATLURI. ANIL KUMAR REDDY (Roll. No. 17SCS5023)** hereby declare that the Project Report entitled **“TRUSTWORTHY KEYWORD SEARCH OVER ENCRYPTED DATA WITH TWO SIDE MULTI ACCESS ENVIRONMENTS SYSTEM”** done by us under the guidance of **Dr. M. SELVI., B.E., M.TECH., (Ph. D).,** at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 19/03/2021
PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.SASIKALA, M.E.,Ph.D., Dean**, School of Computing, **Dr.L.Lakshmanan, M.E.,Ph.D** and **Dr.S.VIGNESHWARI, M.E.,Ph.D., Head of the Department**, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.M.SELVI.,B.E.,M.TECH.,(Ph.D).**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

The untrustworthiness of cloud server and the data privacy of users it is necessary to encrypted the data before outsource the cloud Aiming to realize secure keyword search over encrypted data against malicious users and malicious cloud service providers, we find a compromised method by into the block chain into SSE the cloud storage used in searchable symmetric encryption schemes (SSE) is provided in a private way, which cannot be seen as a true cloud. Moreover, the cloud server is thought to be credible WE begin by pointing out the importance of storing the data in a public chain We introduce a system that leverages blockchain technology to provide a secure distributed data storage with keyword search service The System allows the client to upload their in encrypted form distributes the data content to cloud nodes and ensure data availability using cryptographic techniques we introduce a system that leverages blockchain technology to provide a secure distributed data storage with keyword search service. TKSE realizes server-side verifiability which protects honest cloud servers from being framed by malicious data owners in the data storage phase. Furthermore, blockchain technologies and hash functions are used to enable payment fairness of search fees without introducing any third party even if the user or the cloud is malicious. Our security analysis and performance evaluation indicate that TKSE is secure and efficient and it is suitable for cloud computing.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO.
	ABSTRACT	v
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
	2.1 Identity-based encryption with outsourced revocation in cloud computing	3
	2.2 New publicly verifiable databases with efficient updates	4
	2.3 A searchable symmetric encryption scheme using blockchain	4
	2.4 Blockchain Based system for secure data storage with private Key word search	5
	2.5. Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems	6
3	AIM AND SCOPE OF THE PROJECT	7
	3.1. Objective	7
	3.2. Aim	7
	3.3 Existing System	7
	3.3.1. Disadvantages of Existing System	8
	3.4. Proposed System	8
	3.4.1. Advantages of Proposed System	8
	3.5. System Architecture	9
4	METHODOLOGY	10
	4.1. Hardware Requirements	10
	4.2. Software Requirements	10
	4.3. Introduction of Java	10
	4.3.1. Application of Java	11
	4.3.2. Features of Java	11
	4.4. Collection Framework	13
	4.5. MySQL	14
	4.6. Feasibility Study	15
	4.7. Economic Feasibility	15
	4.8. Technical Feasibility	16

4.9. Social Feasibility	16
4.10. MODULES	16
4.11. Authentication scheme	17
4.11.1. Two-Side Verification	18
4.12. SYSTEM DESIGN	18
4.12.1. Introduction	18
4.12.2. Design Structure	18
4.12.2.1. Input Design	18
4.12.2.2. Output Design	19
4.13. SYSTEM TESTING	19
4.13.1. Testing Process	19
4.13.2. TYPES OF TESTS	20
4.13.2.1. Unit Testing	20
4.13.2.2. Integration Testing	21
4.13.2.3. Functional Testing	21
4.13.2.4. System Testing	21
4.13.2.5. White Box Testing	22
4.13.2.6. Black Box Testing	22
4.14. TEST STRATEGY AND APPROACH	23
4.14.1. Test Objectives	23
4.14.2. Integration Testing	23
4.14.3. Acceptance Testing	23
4.14.4. Alpha Testing	24
4.14.5. Beta Testing	24
4.14.6. Uml Diagram	24
4.14.6.1. Advantages	24
4.15. PROJECT DESCRIPTION	25
4.15.1. USECASE DIAGRAM	25
4.15.2. Class Diagram	26
4.15.3. Sequence Diagram	27
4.15.4. Activity diagram	28
4.15.5. REQUIREMENT ANALYSIS	30
4.15.6. Functional Requirements	30

	4.15.6.1. Non- Functional Requirements	31
5	CONCLUSION AND REFERNCES	32
	APPENDIX	34
	SOURCE CODE	34
	SCREEN SHOTS	47

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.5	System architecture	9
4.13.2.1	Unit Testing	20
4.13.2.5	White box Testing	22
4.13.2.6	Black box Testing	22
4.15.1	Use case Diagram	26
4.15.2	Class Diagram	27
4.15.3	Sequence Diagram	28
4.15.4	Activity diagram	29

CHAPTER 1

INTRODUCTION

In recent years, cloud computing technologies have gotten rapid developments and a line of studies have been done on security issues in cloud computing, such as access control and privacy protection. As a typical service in cloud computing, cloud storage needs both data security and search functionality. In fact, user-side verifiability takes into consideration that the cloud server may be malicious, that is, the cloud server may only return part of search results or maliciously return incorrect results. The issue of user-side verifiability is firstly addressed in. However, these two schemes cannot support server-side verifiability and fair payment without any trusted third party. Furthermore, server-side verifiability takes into consideration that the data owner may be malicious, that is, the data owner may maliciously outsource invalid data in the data storage phase and fraudulently claim compensation later. This concern has not been addressed and even has received little attention in the literature. Last but not least, most of the previous schemes are bank-dependent. Specifically, either the payment issue is not considered or the default traditional payment mechanism is exploited in which a trusted third party (TTP) such as a trustworthy bank has to be introduced for payment fairness. Payment fairness can promote the honest behaviors of users and cloud servers. If a malicious behavior is detected based on the user-side verifiability (resp. server-side verifiability), the data owner (resp. cloud server) should get adequate compensation from the cloud server (resp. data owner) no matter what the cloud server (resp. data owner) does. Therefore, fair payment without any third party is a meaningful and challenging task and it remains in SSE.

In order to thoroughly address the aforementioned challenging issues in cloud computing, we propose TKSE, a Trustworthy Keyword Search scheme over Encrypted data without needing any third party. TKSE is proven secure and our performance evaluation shows its efficiency. In particular, TKSE is characterized by the following desirable features.

- Keyword Search over Encrypted Data. The encrypted data index based on the

Elliptic Curve Digital Signature Algorithm (ECDSA) allows a user to search over the out sourced encrypted data.

- **User-side Verifiability.** In TKSE, a data owner can embed search requirements in to the output script of a join transaction such that the transaction can be redeemed by the cloud server if and only if the output script evaluates to true based on the returned search result. Therefore, TKSE enables the data owner to resist malicious cloud servers and user-side verifiability is realized.

- **Server-side Verifiability.** Similar to user-side verifiability, the public verification of digital signature enables the cloud server to check the validness of the outsourced encrypted data from the data owner in the data storage phase. Thus, malicious data owners can be detected by the cloud server, which realizes server-side verifiability.

Fair Payment and No TTP. Based on hash functions and ECDSA, TKSE is compatible with blockchains such as the Bitcoin blockchain and the Ethereum blockchain. The global consensus and distributed nature of a blockchain enable a fair payment mechanism in TKSE without introducing any TTP.

CHAPTER 2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is taken into account for developing the proposed system.

The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

2.1. Identity - based encryption with outsourced revocation in cloud computing

Identity-Based Encryption (IBE) which simplifies the public key and certificate management at Public Key Infrastructure (PKI) is an important alternative to public key encryption. However, one of the main efficiency drawbacks of IBE is the overhead computation at Private Key Generator (PKG) during user revocation. Efficient revocation has been well studied in traditional PKI setting, but the cumbersome management of certificates is precisely the burden that IBE strives to alleviate. In this paper, aiming at tackling the critical issue of identity revocation, we introduce outsourcing computation into IBE for the first time and propose a revocable

IBE scheme in the server-aided setting. Our scheme offloads most of the key generation related operations during key-issuing and key-update processes to a Key Update Cloud Service Provider, leaving only a constant number of simple operations for PKG and users to perform locally. This goal is achieved by utilizing a novel collusion-resistant technique: we employ a hybrid private key for each user, in which an AND gate is involved to connect and bound the identity component and the time component. Furthermore, we propose another construction which is provable secure under the recently formulized Refereed Delegation of Computation model. Finally, we provide extensive experimental results to demonstrate the efficiency of our proposed construction.

2.2. New publicly verifiable databases with efficient updates

The notion of verifiable database (VDB) enables a resource-constrained client to securely outsource a very large database to an untrusted server so that it could later retrieve a database record and update it by assigning a new value. Also, any attempt by the server to tamper with the data will be detected by the client. Very recently, Catalano and Fiore [17] proposed an elegant framework to build efficient VDB that supports public verifiability from a new primitive named vector commitment. In this paper, we point out Catalano-Fiore's VDB framework from vector commitment is vulnerable to the so-called forward automatic update (FAU) attack. Besides, we propose a new VDB framework from vector commitment based on the idea of commitment binding. The construction is not only public verifiable but also secure under the FAU attack. Furthermore, we prove that our construction can achieve the desired security properties.

2.3. A searchable symmetric encryption scheme using blockchain

At present, the cloud storage used in searchable symmetric encryption schemes (SSE) is provided in a private way, which cannot be seen as a true cloud. Moreover, the cloud server is thought to be credible, because it always returns the search result to the user, even they are not correct. In order to really resist this malicious adversary and accelerate the usage of the data, it is necessary to store the data on a public chain, which can be seen as a decentralized system. As the increasing amount of the data, the search problem becomes more and more intractable, because there

does not exist any effective solution at present. In this paper, we begin by pointing out the importance of storing the data in a public chain. We then innovatively construct a model of SSE using blockchain (SSE-using-BC) and give its security definition to ensure the privacy of the data and improve the search efficiency. According to the size of data, we consider two different cases and propose two corresponding schemes. Lastly, the security and performance analyses show that our scheme is feasible and secure.

2.4. Blockchain Based system for secure data storage with private

Key word search

Traditional cloud storage has relied almost exclusively on large storage providers, who act as trusted third parties to transfer and store data. This model poses a number of issues including data availability, high operational cost, and data security. In this paper, we introduce a system that leverages blockchain technology to provide a secure distributed data storage with keyword search service. The system allows the client to upload their data in encrypted form, distributes the data content to cloud nodes and ensures data availability using cryptographic techniques. It also provides the data owner a capability to grant permission for others to search on her data. Finally, the system supports private keyword search over the encrypted dataset.

2.5. Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems

Electronic Health Records (EHRs) are entirely controlled by hospitals instead of patients, which complicates seeking medical advices from different hospitals. Patients face a critical need to focus on the details of their own healthcare and restore management of their own medical data. The rapid development of blockchain technology promotes population healthcare, including medical records as well as patient-related data. This technology provides patients with comprehensive, immutable records, and access to EHRs free from service providers and treatment websites. In this paper, to guarantee the validity of EHRs encapsulated in blockchain, we present an attribute-based signature scheme with multiple authorities, in which a patient endorses a message according to the attribute while disclosing no information other than the evidence that he has attested to it.

Furthermore, there are multiple authorities without a trusted single or central one to generate and distribute public/private keys of the patient, which avoids the escrow problem and conforms to the mode of distributed data storage in the blockchain. By sharing the secret pseudorandom function seeds among authorities, this protocol resists collusion attacks out of N from $N - 1$ corrupted authorities. Under the assumption of the computational bilinear Diffie-Hellman, we also formally demonstrate that, in terms of the unforgeability and perfect privacy of the attribute-signer, this attribute-based signature scheme is secure in the random oracle model. The comparison shows the efficiency and properties between the proposed method and methods proposed in other studies.

CHAPTER 3

AIM AND SCOPE OF THE PROJECT

3.1. OBJECTIVE

The main objective of this system, our security analysis and performance evaluation indicate that TKSE is secure and efficient and it is suitable for cloud computing.

3.2. AIM

The main aim of this project, Two-Side verification is a process that involves two authentication methods performed one after the other to verify that someone or something requesting access is who or what they are declared to be.

3.3. EXISTING SYSTEM

Furthermore, blockchain technologies and hash functions are used to enable payment fairness of search fees without introducing any third party even if the user or the cloud is malicious. In TKSE, the encrypted data index based on digital signature allows a user to search over the outsourced encrypted data and check whether the search result returned by the cloud fulfills the pre-specified search requirements. Our security analysis and performance evaluation indicate that TKSE is secure and efficient and it is suitable for cloud computing. First proposed an SSE scheme with user-side verifiability based on. User side verifiability has also been realized in SSE. In addition, fair payment is fulfilled based on blockchain technologies and hash function without introducing any TTP.

3.3.1. Disadvantages for Existing System

- Data confidentiality and privacy has been achieved but identity privacy neglected.
- Less security
- It is not having any data content

3.4. PROPOSED SYSTEM

In order to preserve the search functionality searchable encryption technologies have been developed in two representative setting including the symmetric key setting furthermore the idea in cannot be directly combined with blockchain technologies in that the condition of redeeming search fees should be specified by user and CSP and it requires the MAC secret key Cryptography hash function server as a fundamental building block of information security and are used in numerous security application and protocol such as digital signature schemes construction of MAC and random number generation for ensuring data integrity and data origin authentication Hashing algorithms are used in all sorts of ways – they are used for storing passwords, in computer vision, in databases. Here we are using SHA stands for Secure Hashing Algorithm to reduce the block of data and need to improve the data security.

3.4.1. Advantages for proposed system

- Save data management cost
- To protect user privacy and data security
- Message authentication code
- Integrity protection

3.5. SYSTEM ARCHITECTURE

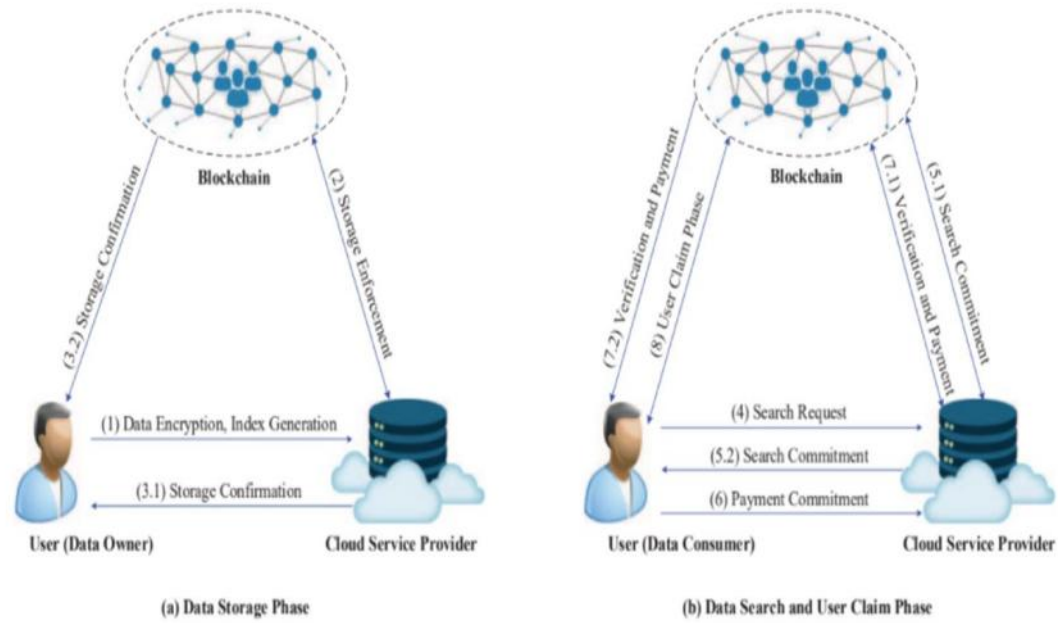


Fig.3.5. System architecture

CHAPTER 4

METHODOLOGY

4.1. HARDWARE REQUIREMENTS

- System - Pentium-IV
- Speed - 2.4GHZ
- Hard disk - 40GB
- RAM - 512MB

4.2. SOFTWARE REQUIREMENTS

- Operating System - Windows XP
- Coding language - Java
- IDE - NetBeans
- Database - MYSQL

4.3. INTRODUCTION TO JAVA

Java is one of the world's most important and widely used computer languages, and it has held this distinction for many years. Unlike some other computer languages whose influence has wearied with passage of time, while Java's has grown.

4.3.1. Application of Java

Java is widely used in every corner of world and of human life. Java is not only used in software's but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usages of Java:

- Developing Desktop Applications
- Web Applications like LinkedIn.com, Snapdeal.com etc.
- Mobile Operating System like Android
- Embedded Systems
- Robotics and games etc.

4.3.2. Features of Java

The prime reason behind creation of Java was to bring portability and security feature into a computer language. Beside these two major features, there were many other features that played an important role in molding out the final form of this outstanding language. Those features are;

(i) Simple

Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.

E.g. Pointers and Operator Overloading are not there in java but were an important part of C++.

(ii) Object Oriented

In java everything is Object which has some data and behavior. Java can be easily extended as it is based on Object Model.

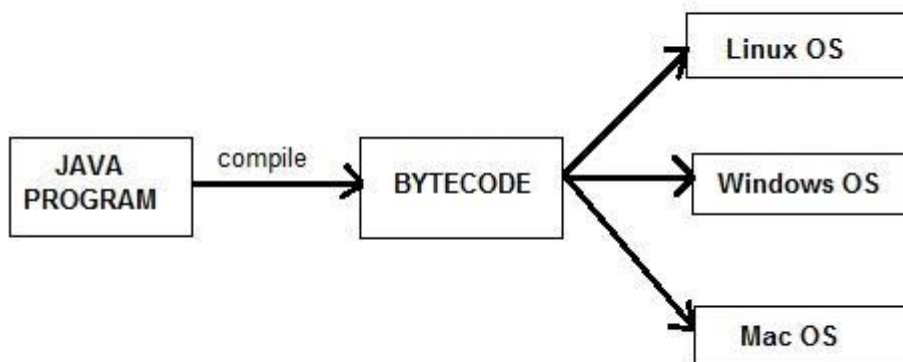
(iii) Robust

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.

(iv) **Platform Independent**

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

On compilation Java program is compiled into byte code. This byte code is platform independent and can be run on any machine, plus this byte code format also provide security. Any machine with Java Runtime Environment can run Java Programs.



(v) **Secure**

When it comes to security, Java is always the first choice. With java secure features it enables us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

(vi) **Multi-Threading**

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

(vii) **Architectural Neutral**

Compiler generates byte codes, which have nothing to do with a particular computer architecture, hence a Java program is easy to interpret on any machine.

(viii) **Portable**

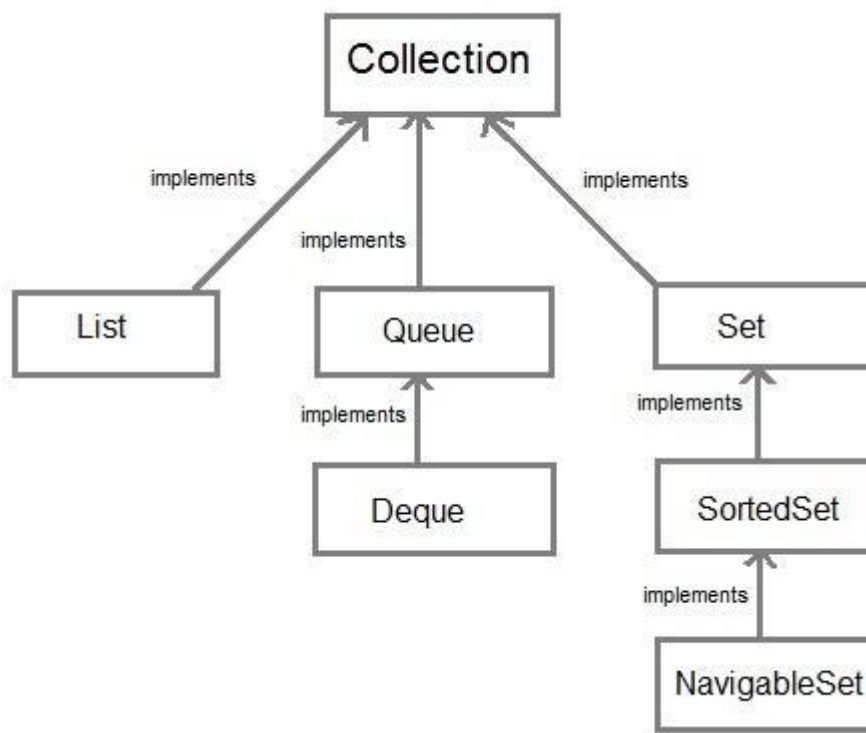
Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

(ix) **High Performance**

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But Java enables high performance with the use of just-in-time compiler.

4.4. Collection Framework

Collection framework was not part of original Java release. Collections was added to J2SE 1.2. Prior to Java 2, Java provided achoo classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects. Collection framework provides many important classes and interfaces to collect and organize group of alike objects.



4.5. MySQL

MySQL, officially, but also called "My Sequel" is 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, though SQLite probably has more total embedded deployments. 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, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr and YouTube.

4.6. Feasibility Study

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

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

4.7. Economic Feasibility

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

4.8. Technical Feasibility

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

4.9. Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

4.10. MODULES

- (i) Login
- (ii) Registration
- (iii) Create Secrete Key
- (iv) Authentication Scheme
- (v) Two-Side Verification

(i) Login

Logins are used by websites, computer applications, and mobile apps. They are a security measure designed to prevent unauthorized access to confidential data. When a login fails (i.e., the username and password combination does not match a user account), the user is disallowed access. Many systems block users from even trying to log in after multiple failed login attempts.

(ii) **Registration**

A registered user is a user of a website, program, or other system who has previously registered. Registered user normally provide some sort of credentials (such as a username or e-mail address, and a password) to the system in order to prove their identity: this is known as logging in. Systems intended for use by the general public often allow any user to register simply by selecting a register or sign-up function and providing these credentials for the first time. Registered users may be granted privileges beyond those granted to unregistered users.

(iii) **Create Secret Key**

Cryptography is the practice and study of secure communication in the presence of third parties. In the past cryptography referred mostly to encryption. Encryption is the process of converting plain text information to cipher text. Reverse is the decryption. Encryption is a mechanism to make the information confidential to anyone except the wanted recipients. Cipher is the pair of algorithms that creates encryption and decryption. Cipher operation is depending on algorithm and the key. Key is the secret that known by communicants.

4.11. Authentication scheme

It is used to solve the problem of authenticating the keys of the person (say "person B") to whom some other person ("person A") is talking to or trying to talk to. In other words, it is the process of assuring that the key of "person A" held by "person B" does in fact belong to "person A" and vice versa.

This is usually done after the keys have been shared among the two sides over some secure channel, although some of the algorithms share the keys at the time of authentication also.

The simplest solution for this problem is for the two users concerned to meet face-to-face and exchange keys. However, for systems in which there are a large number

of users or in which the users do not personally know each other (e.g., Internet shopping) this is not practical. There is various algorithm for both symmetric keys and asymmetric public key cryptography to solve this problem.

4.11.1. Two-Side Verification

In this module, Two-Side verification is a process that involves two authentication methods performed one after the other to verify that someone or something requesting access is who or what they are declared to be.

4.12. SYSTEM DESIGN

4.12.1. Introduction

Design is a multi- step that focuses on data structure software architecture, procedural details, algorithm etc., and interface between modules. The design process also translates the requirements into presentation of software that can be accessed for quality before coding begins. Computer software design change continuously as new methods; better analysis and border understanding evolved. Software design is at relatively early stage in its revolution.

Therefore, software design methodology lacks the depth, flexibility and quantitative nature that are normally associated with more classical engineering disciplines. However, techniques for software designs do exist, criteria for design qualities are available and design notation can be applied.

4.12.2. Design Structure

4.12.2.1. Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The

input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

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

4.12.2.2. Output Design

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

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

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

4.13. SYSTEM TESTING

4.13.1. Testing Process

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

4.13.2. TYPES OF TESTS

4.13.2.1. Unit Testing

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

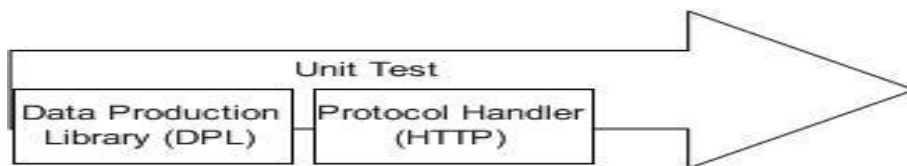


Fig.4.13.2.1. Unit Testing

4.13.2.2. Integration Testing

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

4.13.2.3. Functional Testing

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

Functional testing is centered on the following items:

Valid Input is used to identify classes of valid input that must be accepted.

Invalid Input is used to identify classes of invalid input that must be rejected.

Functions are used to identify functions that must be exercised.

Output is used to identify classes of application outputs.

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

4.13.2.4. System Testing

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

4.13.2.5. White Box Testing

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

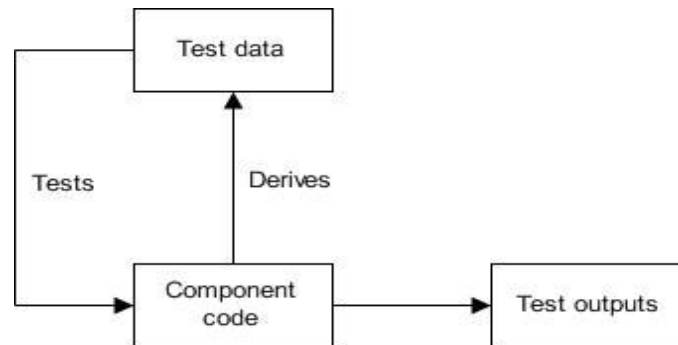


Fig. 4.13.2.5. White box Testing

4.13.2.6. Black Box Testing

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

Considering how the software works.

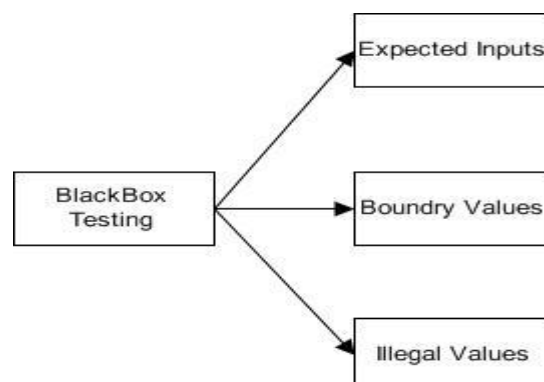


Fig. 4.13.2.6. Black box Testing

4.14. TEST STRATEGY AND APPROACH

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

4.14.1. Test Objectives

All field entries must work properly.

- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

4.14.2. Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications.

4.14.3. Acceptance Testing

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

4.14.4. Alpha Testing

In software development, alpha test will be a test among the teams to confirm that your product works. Originally, the term alpha test meant the first phase of testing in a software development process. The first phase includes unit testing, component testing, and system testing. It also enables us to test the product on the lowest common denominator machines to make sure download times are acceptable and pre loaders work.

4.14.5. Beta Testing

In software development, a beta test is the second phase of software testing in which a sampling of the intended audience tries the product out. Beta testing can be considered "pre-release testing." Beta test versions of software are now distributed to curriculum specialists and teachers to give the program a "real world" test.

4.14.6. Uml Diagram

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a very important part of developing objects-oriented software and the software development process. UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

UML is simply another graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

4.14.6.1. Advantages

- To represent complete systems (instead of only the software portion) using object-oriented concepts
- To establish an explicit coupling between concepts and executable code
- To take into account the scaling factors that are inherent to complex and critical systems
- To creating a modeling language usable by both humans and machines

UML defines several models for representing systems

- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements of the user
- The interaction model represents the scenarios and messages flows
- The implementation model shows the work units
- The deployment model provides details that pertain to process allocation

4.15. PROJECT DESCRIPTION

4.15.1. USECASE DIAGRAM

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

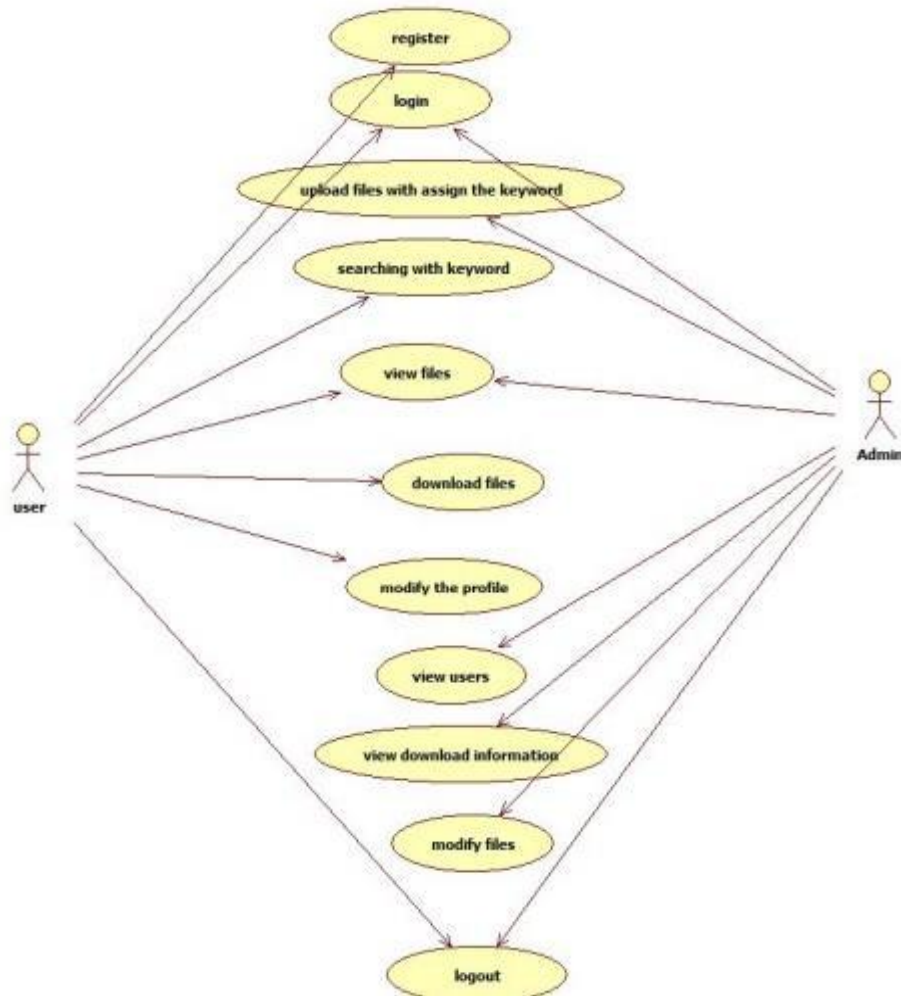


Fig.4.15.1. Use case Diagram

4.15.2. Class Diagram

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. Class diagrams are arguably the most used UML diagram type. It is the main building block of any object-oriented solution. It shows the classes

in a system, attributes and operations of each class and the relationship between each class. In most modeling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many classes related classes are grouped together to create class diagrams. Different relationships between diagrams are show by different types of Arrows. Below is a image of a class diagram. Follow the link for more class diagram examples.

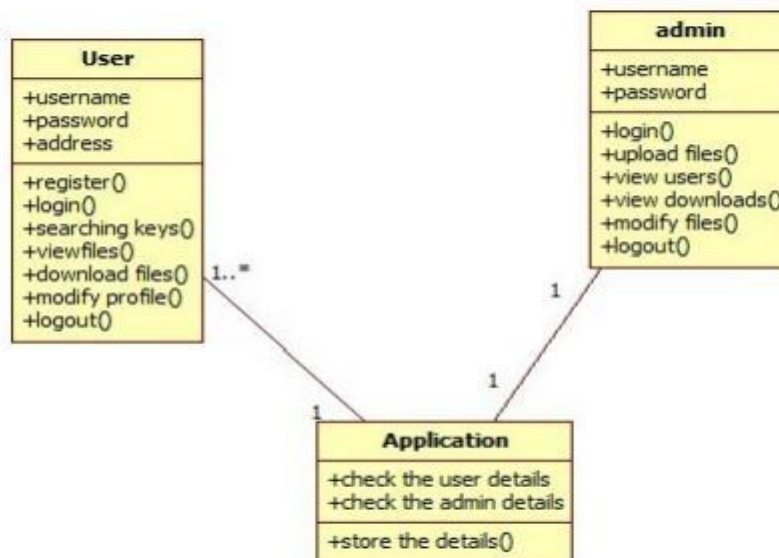


Fig.4.15.2. Class Diagram

4.15.3. Sequence Diagram

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows. This article explains the purpose and the basics of Sequence diagrams.

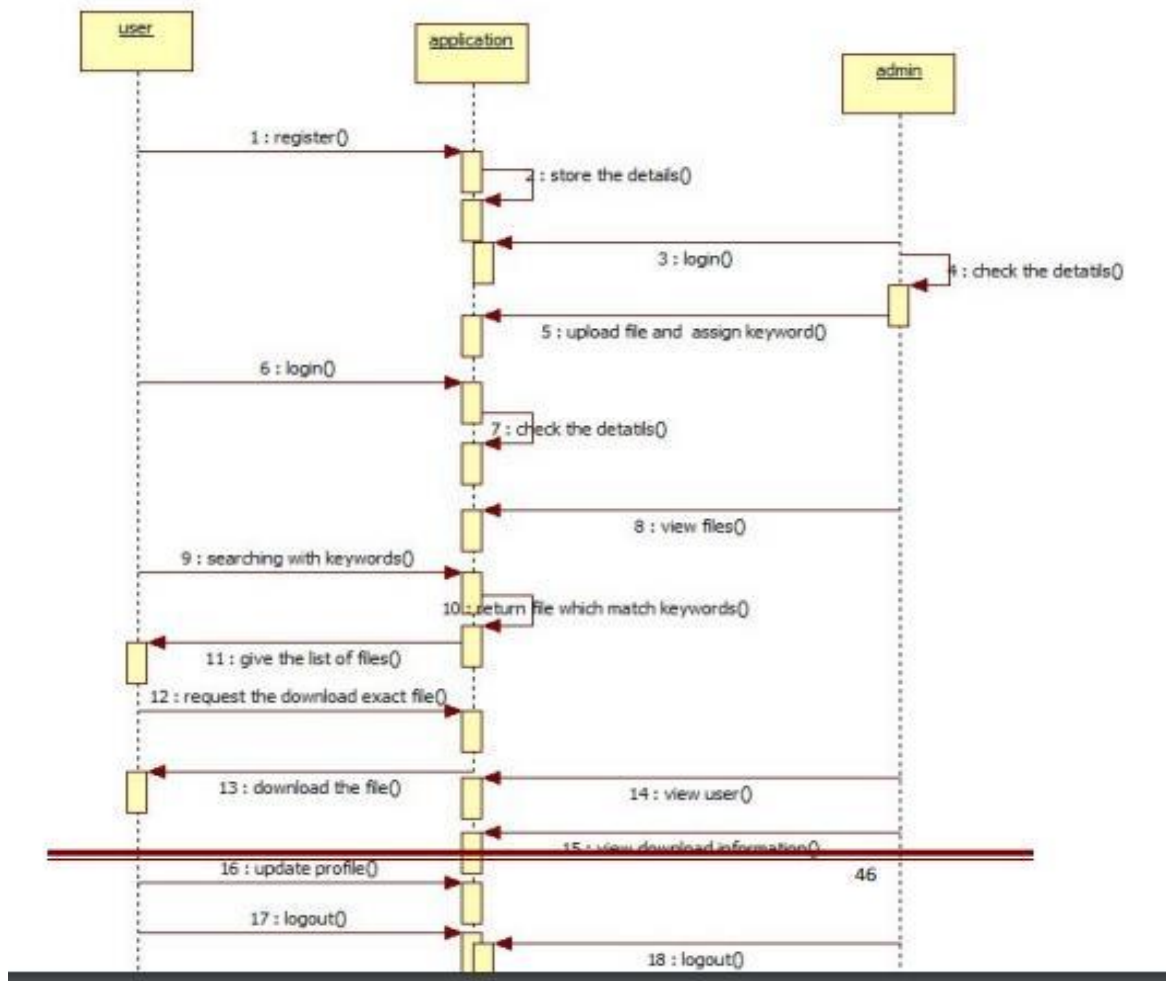
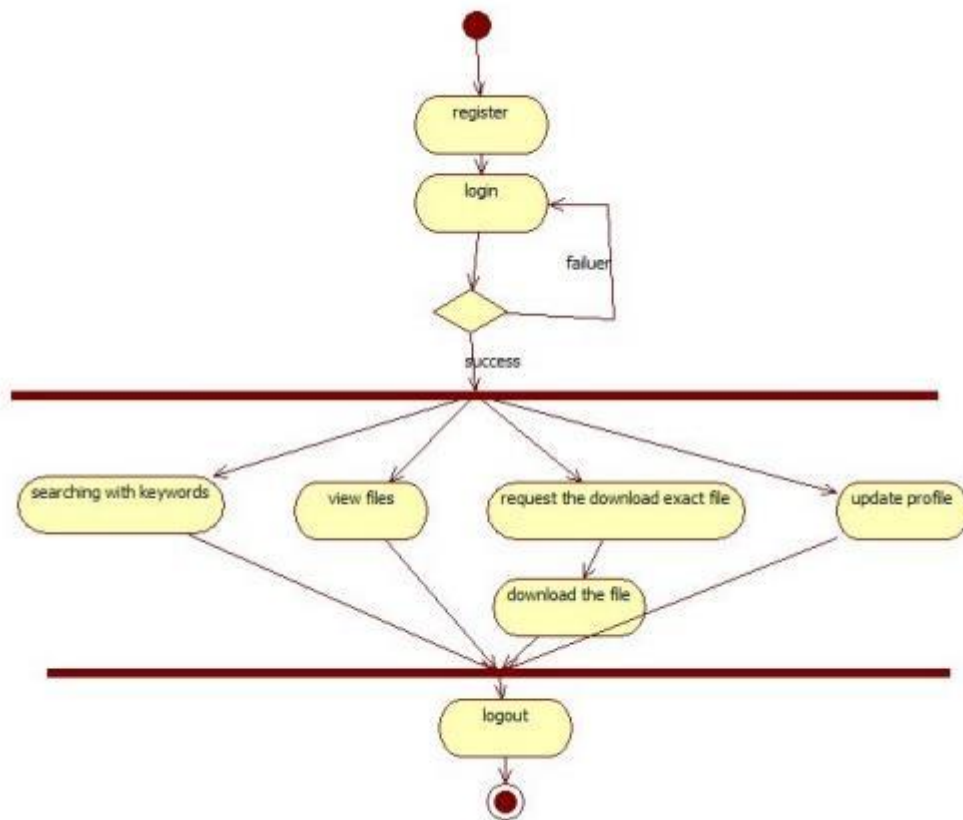


Fig. 4.15.3. Sequence Diagram

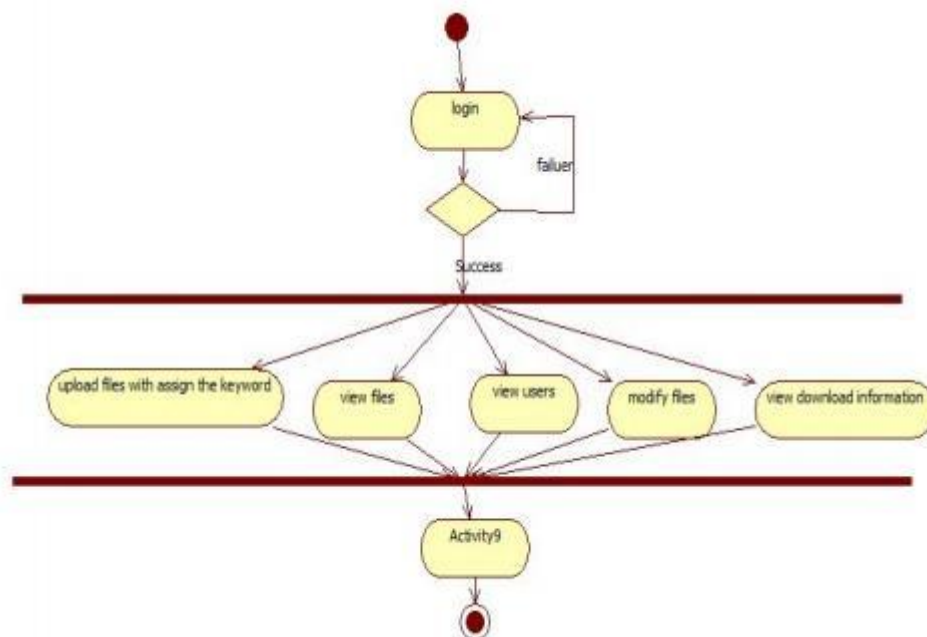
4.15.4. Activity diagram

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.

Activity Diagram for User



Activity Diagram for Admin



4.15.5. Requirement Analysis

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analyzing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

4.15.6. Functional Requirements

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

(i) Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

(ii) Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

(iii) Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

(iv) Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working

around any failure.

(v) **Compatibility**

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

(vi) **Flexibility**

The flexibility of the project is provided in such a way that it has the ability to run on different environments being executed by different users.

(vii) **Safety**

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

4.15.6.1. Non- Functional Requirements

(i) **Portability**

It is the usability of the same software in different environments. The project can be run in any operating system.

(ii) **Performance**

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

(iii) **Accuracy**

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

(iv) **Maintainability**

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyses, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

CHAPTER 5

CONCLUSION AND REFERENCES

5.1. CONCLUSION

The System allows the client to upload there in encrypted form distributes the data content to cloud nodes and ensure data availability using cryptographic techniques we introduce a system that leverages blockchain technology to provide a secure distributed data storage with keyword search service. TKSE realizes server-side verifiability which protects honest cloud servers from being framed by malicious data owners in the data storage phase. Furthermore, blockchain technologies and hash functions are used to enable payment fairness of search fees without introducing any third party even if the user or the cloud is malicious. Our security analysis and performance evaluation indicate that TKSE is secure and efficient and it is suitable for cloud computing.

REFERENCES

[1] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 425–437, 2015

[2] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 546–556, 2015

[3] H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using blockchain," *arXiv preprint, 2017*. [Online]. Available: <https://arxiv.org/pdf/1711.01030.pdf>

[4] H. G. Do and W. K. Ng, "Blockchain based system for secured data storage with private keyword search," in *Services (SERVICES), 2017 IEEE World Congress on. IEEE, 2017*, pp. 90–93.

[5] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature

scheme with multiple authorities for blockchain in electronic health records systems,”
IEEE Access, vol. 776, no. 99, pp. 1–12, 2018.

APPENDIX

A) SOURCE CODE

ENCRYPTION

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package com.secure.multi.action;

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JOptionPane;
import sun.misc.BASE64Encoder;

public class encryption {

    public String encrypt(String text, SecretKey secretkey) {
        String plainData = null, cipherText = null;
        try {
            plainData = text;

            Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
            aesCipher.init(Cipher.ENCRYPT_MODE, secretkey);//initiating cipher
            encryption using secretkey

            byte[] byteDataToEncrypt = plainData.getBytes();
            byte[] byteCipherText =
            aesCipher.doFinal(byteDataToEncrypt);//encrypting data

            cipherText = new
            BASE64Encoder().encode(byteCipherText);//converting encrypted data to
            string
```

```
        System.out.println("\n Given text : " + plainData + " \n Cipher Data : " + cipherText);
```

```
    } catch (Exception e) {  
        System.out.println(e);  
    }  
    return cipherText;  
}  
  
}
```

DECRYPTION

```
/*  
 * To change this template, choose Tools | Templates  
 * and open the template in the editor.  
 */  
package com.secure.multi.action;  
  
/**  
 *  
 * @author SARA  
 */  
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;  
import java.io.ByteArrayOutputStream;  
import java.io.FileInputStream;  
import java.io.FileWriter;  
import java.util.Scanner;  
  
import javax.crypto.Cipher;  
import javax.crypto.KeyGenerator;  
import javax.crypto.SecretKey;  
import javax.crypto.spec.SecretKeySpec;  
import javax.swing.JOptionPane;  
import sun.misc.BASE64Decoder;  
import sun.misc.BASE64Encoder;  
  
public class decryption {  
    public String decrypt(String txt, String skey) {  
        String decryptedtext = null;  
        try {  
  
            //converting string to secretkey
```

```

byte[] bs = Base64.decode(skey);
SecretKey sec = new SecretKeySpec(bs, "AES");
    System.out.println("converted string to seretkey:" + sec);

    System.out.println("secret key:" + sec);

    Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
    aesCipher.init(Cipher.ENCRYPT_MODE, sec);//initiating cipher
encryption using secretkey

    byte[] byteCipherText = new BASE64Decoder().decodeBuffer(txt);
//encrypting data

    // System.out.println("ciper text:"+byteCipherText);
    aesCipher.init(Cipher.DECRYPT_MODE, sec,
aesCipher.getParameters());//initiating cipher decryption

    byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);
    decryptedtext = new String(byteDecryptedText);

    System.out.println("Decrypted Text:" + decryptedtext);
} catch (Exception e) {
    System.out.println(e);
}
return decryptedtext;
}
}

```

UPLOAD

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.secure.multi.action;

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.BufferedReader;

```

```

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory; import
org.apache.commons.fileupload.FileUploadException; import
org.apache.commons.fileupload.disk.DiskFileItemFactory; import
org.apache.commons.fileupload.servlet.ServletFileUpload;

```

```
/**
```

```
*

```

```
* @author SARA

```

```
*/

```

```
public class Upload extends HttpServlet {

```

```

    private static java.sql.Date getCurrentDate() {
        java.util.Date today = new java.util.Date();
        return new java.sql.Date(today.getTime());
    }

```

```
/**

```

```

    * Processes requests for both HTTP <code>GET</code> and
    <code>POST</code>

```

```
    * methods.

```

```
    *

```

```
    * @param request servlet request

```

```
    * @param response servlet response

```

```
    * @throws ServletException if a servlet-specific error occurs

```

```

    * @throws IOException if an I/O error occurs
    */
    protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            Connection con;
            PreparedStatement pstmt = null;
            String fname = "";
            String keyword = "";
            String cd = "";
            String a = (String) request.getSession().getAttribute("email");
            System.out.println("User Name : " + a);
            try {
                boolean isMultipartContent =
ServletFileUpload.isMultipartContent(request);
                if (!isMultipartContent) {
                    return;
                }
                FileItemFactory factory = new DiskFileItemFactory();
                ServletFileUpload upload = new ServletFileUpload(factory);
                try {
                    List<FileItem> fields = upload.parseRequest(request);
                    Iterator<FileItem> it = fields.iterator();
                    if (!it.hasNext()) {
                        return;
                    }
                    while (it.hasNext()) {
                        FileItem fileItem = it.next();
                        if (fileItem.getFieldName().equals("fname")) {
                            fname = fileItem.getString();
                            System.out.println("File Name" + fname);
                        } else if (fileItem.getFieldName().equals("fkey")) {
                            keyword = fileItem.getString();
                            System.out.println("File Keyword" + keyword);
                        } else {
                        }
                        boolean isFormField = fileItem.isFormField();
                        if (isFormField) {
                        } else {

```

```

try {
    con = Dbconnection.getConnection();
    pstmt = con.prepareStatement("insert into files (file,
keyword, filetype, filename, CDate, owner, size, data, frank,
file_key)values(?,?,?,?,?,?,?,?,?,?)");
    System.out.println("getD " + fileItem.getName());
    String str =
getStringFromInputStream(fileItem.getInputStream());
    // secretkey generating
    KeyGenerator keyGen = KeyGenerator.getInstance("AES");
    keyGen.init(128);
    SecretKey secretKey = keyGen.generateKey();
    System.out.println("secret key:" + secretKey);
    //converting secretkey to String
    byte[] be = secretKey.getEncoded();//encoding secretkey
    String skey = Base64.encode(be);
    System.out.println("converted secretkey to string:" + skey);
    String cipher = new encryption().encrypt(str, secretKey);
    System.out.println(str);
    //for get extension from given file
    String b =
fileItem.getName().substring(fileItem.getName().lastIndexOf('.'));
    System.out.println("File Extension" + b);
    pstmt.setBinaryStream(1, fileItem.getInputStream());
    pstmt.setString(2, keyword);
    pstmt.setString(3, b);
    pstmt.setString(4, fname);
    pstmt.setDate(5, getDate());
    pstmt.setString(6, a);
    pstmt.setLong(7, fileItem.getSize());
    pstmt.setString(8, cipher);
    pstmt.setString(9, "0");
    pstmt.setString(10, skey);
    /*Cloud Start*/
    File f = new File(fileItem.getName());
    FileWriter fw = new FileWriter(f);
    fw.write(cipher);
    fw.close();
    Ftpcon ftpcon = new Ftpcon();
    ftpcon.upload(f, fname);
    /*Cloud End*/
    int i = pstmt.executeUpdate();
    if (i == 1) {

```

```

        response.sendRedirect("owner.jsp?msg1=success");
    } else {
        response.sendRedirect("owner.jsp?msgg=failed");
    }
    con.close();
} catch (Exception e) {
    out.println(e.toString());
}
}
}
} catch (FileUploadException e) {
} catch (Exception ex) {
    Logger.getLogger(Upload.class.getName()).log(Level.SEVERE,
null, ex);
}
} finally {
    out.close();
}
}
}
}

```

```

private static String getStringFromInputStream(InputStream is) {
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();
    String line;
    try {
        br = new BufferedReader(new InputStreamReader(is));
        while ((line = br.readLine()) != null) {
            sb.append(line + "\n");
        }
    } catch (IOException e) {
    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (IOException e) {
            }
        }
    }
    return sb.toString();
}
}

```

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on

the + sign on the left to edit the code.">

```
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
} // </editor-fold>

}
```

DOWNLOAD

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.secure.multi.action;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author SARA
 */
public class download extends HttpServlet {

    /**
     * Processes requests for both HTTP GET and
     * POST
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs

```

```

    * @throws IOException if an I/O error occurs
    */
    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException, SQLException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            String[] filedetails = request.getQueryString().split(",");
            String fname = null, fkey = null;
            InputStream is = null;

            Connection con = Dbconnection.getConnection();
            Statement st = con.createStatement();
            ResultSet rt = st.executeQuery("select * from files where filename='" +
filedetails[0] + "' AND owner='" + filedetails[1] + "'");
            if (rt.next()) {
                fname = rt.getString("filename");
                fkey = rt.getString("file_key");
                is = (InputStream) rt.getAsciiStream("data");
            } else {
                out.println("error while retrieving data");
            }
            BufferedReader br = new BufferedReader(new InputStreamReader(is));
            String temp = null;
            StringBuffer sb = new StringBuffer();
            while ((temp = br.readLine()) != null) {
                sb.append(temp + "\n");
            }
            String content = new decryption().decrypt(sb.toString(), fkey);
            response.setHeader("Content-Disposition", "attachment;filename=\"\" +
fname + "\"");
            out.write(content);
        }
    }
}

```

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/**

* Handles the HTTP <code>GET</code> method.

*

* @param request servlet request

* @param response servlet response

```

    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(download.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(download.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }

```

```
    } // </editor-fold>

}
```

DB CONNECTION

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.secure.multi.action;

import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author SARA
 */
public class Dbconnection {

    public static Connection getConnection() {
        Connection con = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/secure", "root",
"root");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return con;
    }
}
```

FTP CONNECTION

```

package com.secure.multi.action;

import java.io.File;
import java.io.FileInputStream;
import org.apache.commons.net.ftp.FTPClient;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author SARA
 */
public class Ftpcon {

    FTPClient client = new FTPClient();
    FileInputStream fis = null;
    boolean status;

    public boolean upload(File file,String fname) {
        try {
            client.connect("ftp.drivehq.com");
            client.login("projecttesting", "projecttesting");
            client.enterLocalPassiveMode();
            fis = new FileInputStream(file);
            status = client.storeFile("/files/" + fname, fis);
            client.logout();
            fis.close();

        } catch (Exception e) {
            System.out.println(e);
        }
        if (status) {
            System.out.println("success");
            return true;
        } else {
            System.out.println("failed");
            return false;
        }
    }
}

```

B) Screenshots

Home Page: - The above screenshot gives the view of homepage in the webpage. The homepage consists a sub options of owner details and we have to enter the details of owner and user.

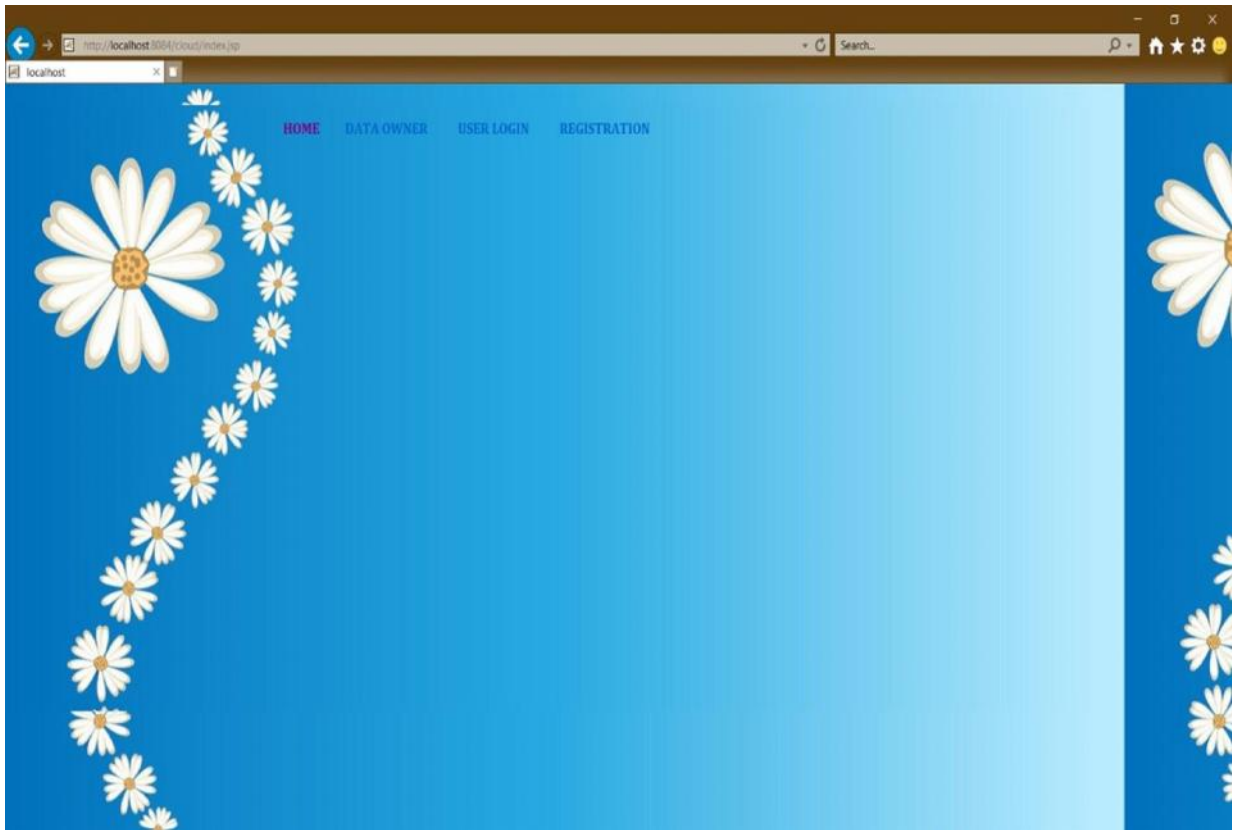


Fig.5.1 Homepage

Owner Registration: - The below screenshot shows the owner registration clicks on name, password, email, DOB, Gender, male, role, location and Registered Successfully.

A screenshot of a web browser displaying a registration form titled "Registration Form". The browser's address bar shows "http://localhost:8084/cloud/index.php#". The page has a blue background with a decorative border of white daisies. The navigation menu at the top includes "HOME", "DATA OWNER", "USER LOGIN", and "REGISTRATION". The form fields are as follows:

Field	Value
Name	Akshaya
Password	*****
Email	akshayadantha@gmail.com
Date of Birth	22/08/1999
Gender	Female
Role	Owner
Location	Guntur

At the bottom of the form are "Submit" and "Reset" buttons.

Fig.5.2. Owner Registration

User Registration: - The below screenshot shows the view of user registration same as owner in case role option user choose user option and registered successfully.

A screenshot of a web browser displaying a registration form titled "Registration Form". The browser's address bar shows "http://localhost:8084/cloud/index.php?msg=success#". The page has a blue background with a decorative border of white daisies. The navigation menu at the top includes "HOME", "DATA OWNER", "USER LOGIN", and "REGISTRATION". The form fields are as follows:

Field	Value
Name	Nike
Password	*****
Email	Ni
Date of Birth	30/08/2000
Gender	Female
Role	User
Location	Vadalamudi

At the bottom of the form are "Submit" and "Reset" buttons.

Fig.5.3. User Registration

Data owner Login page: - The below screenshot shows the data owner login access through username and password registered with owner details and two side verification secret keys generated automatically in MySQL.



Fig.5.4. Data owner Login page

Welcome: - The below screenshot shows the view of owner welcome page.



Fig.5.5. Welcome

Data owner Uploading File: - The below screenshot shows the view of uploading files and the file is uploaded can see in next screen.

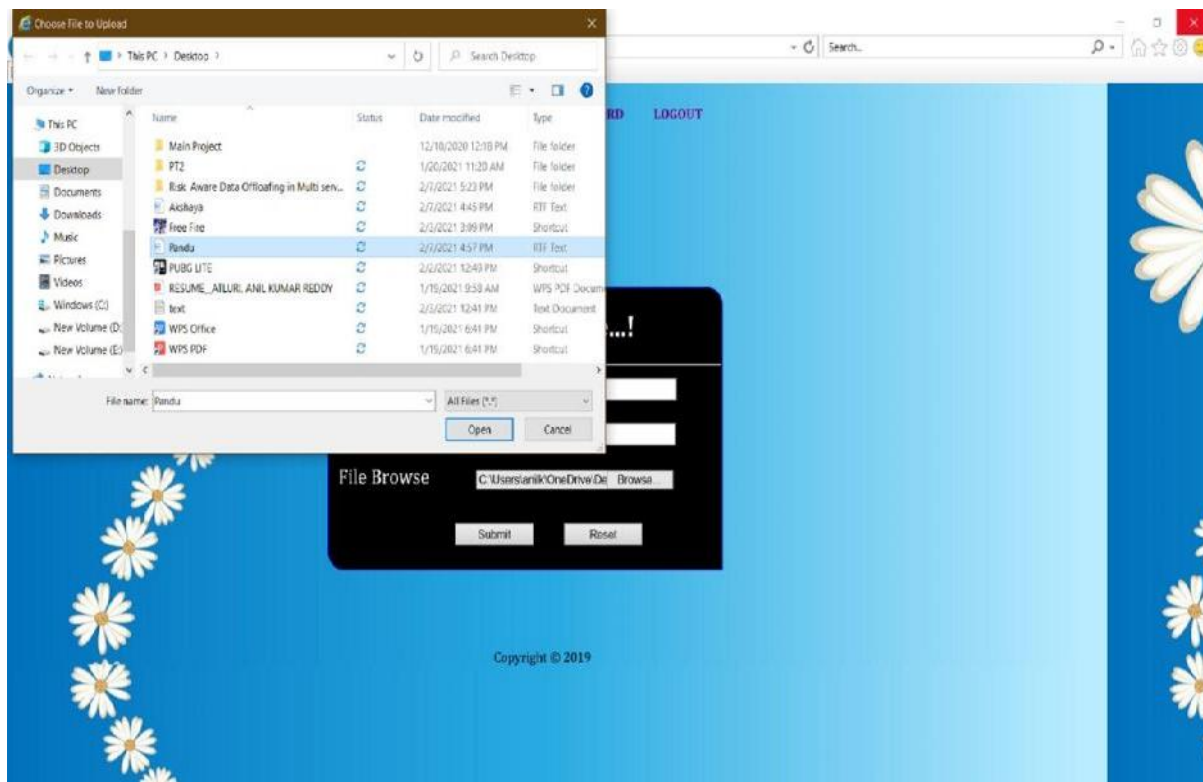


Fig.5.5. Data owner Uploading File

Password Request: - The below screenshot shows the view of changing password through registered mail id.

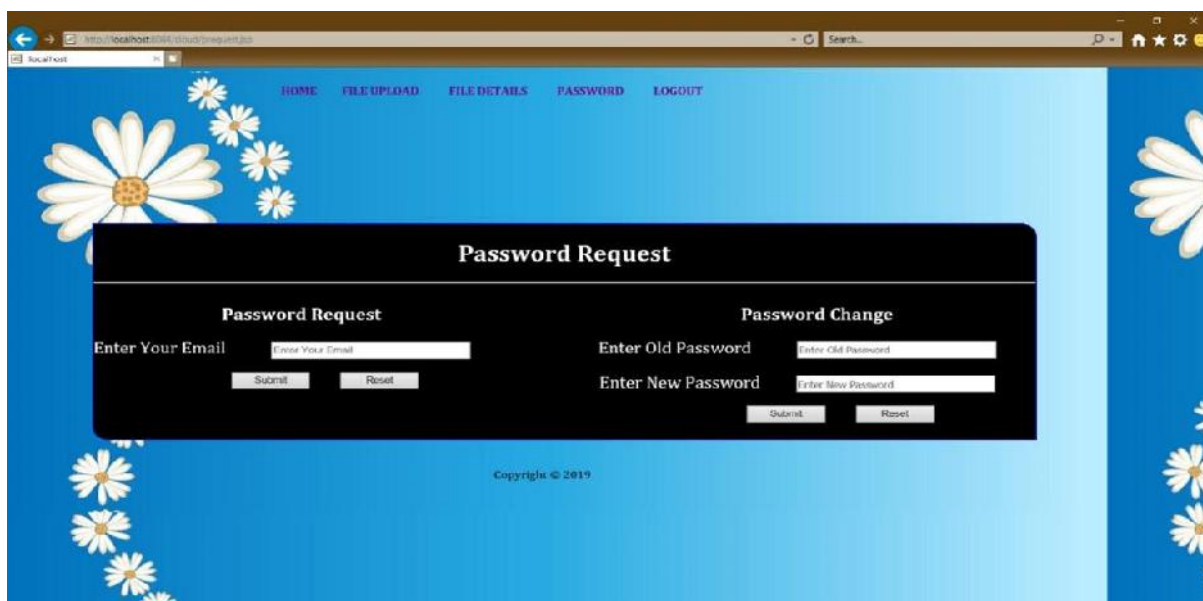
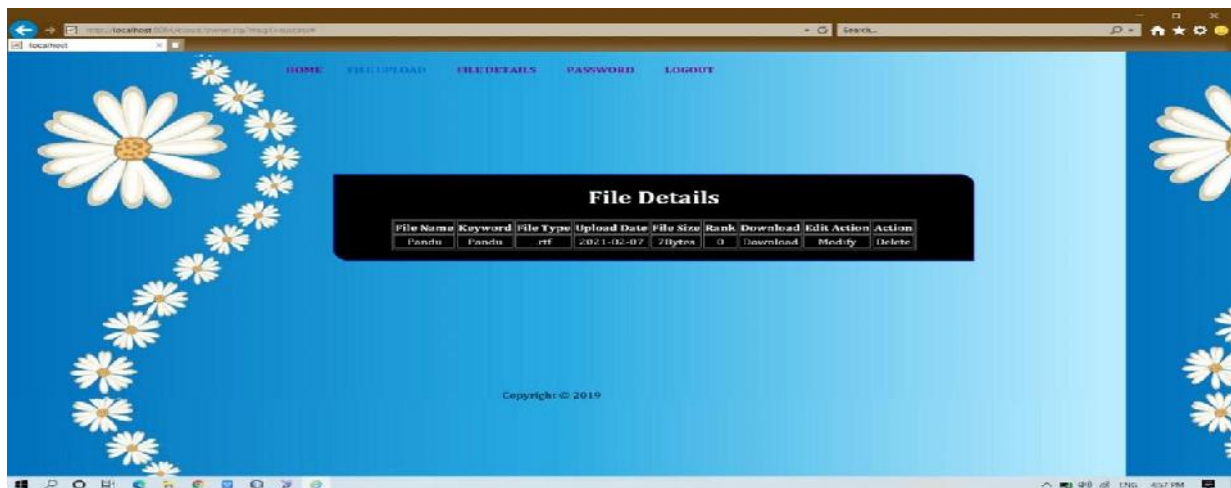


Fig.5.6. Password Request

Uploaded File Details: - The below screenshot shows the view of uploaded file details.

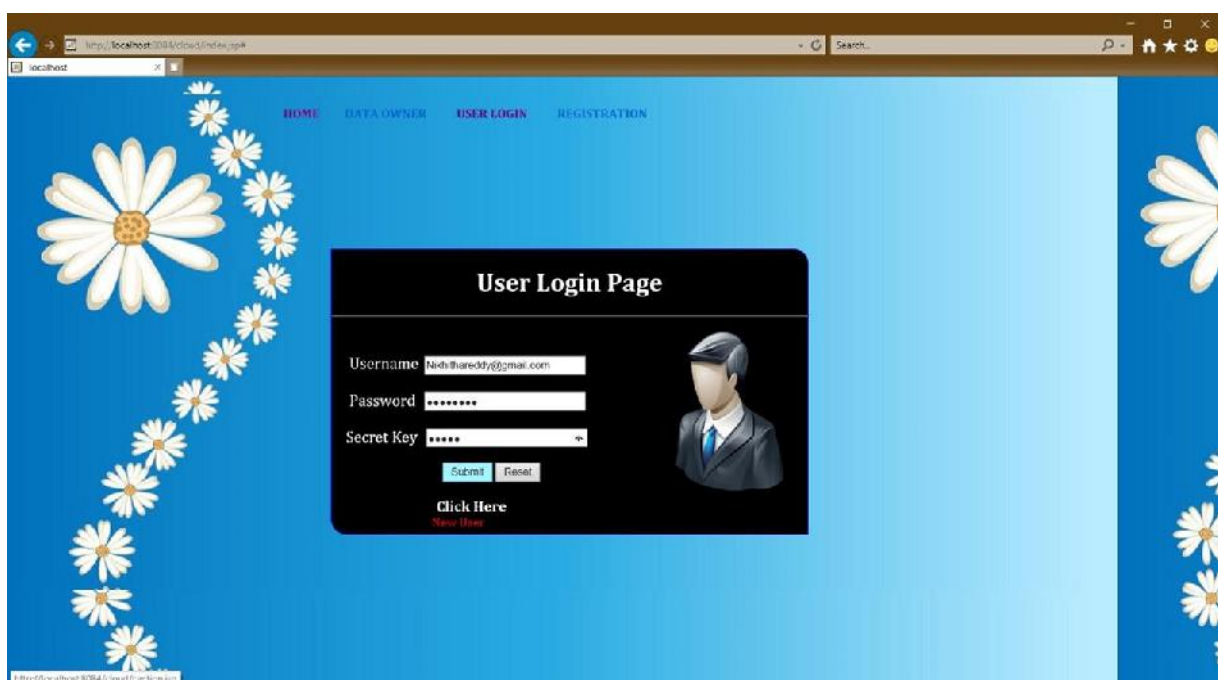


File Name	Keyword	File Type	Upload Date	File Size	Rank	Download	Edit Action	Action
Pandu	Pandu	rar	2021-02-07	71bytes	0	Download	Modify	Delete

Copyright © 2019

Fig.5.7. Uploaded File Details

User Login: - The below screenshot shows the view of user registered with usernames and passwords same as owner & user also two side verification secret keys generated automatically.



User Login Page

Username:

Password:

Secret Key:

[Click Here](#)
New User

Fig.5.8. User Login

File Search: - The user needs to search as file keyword and there will be a secret key user login needs to give secret key and then we can able to see file name as per uploaded file (e.g. amma).

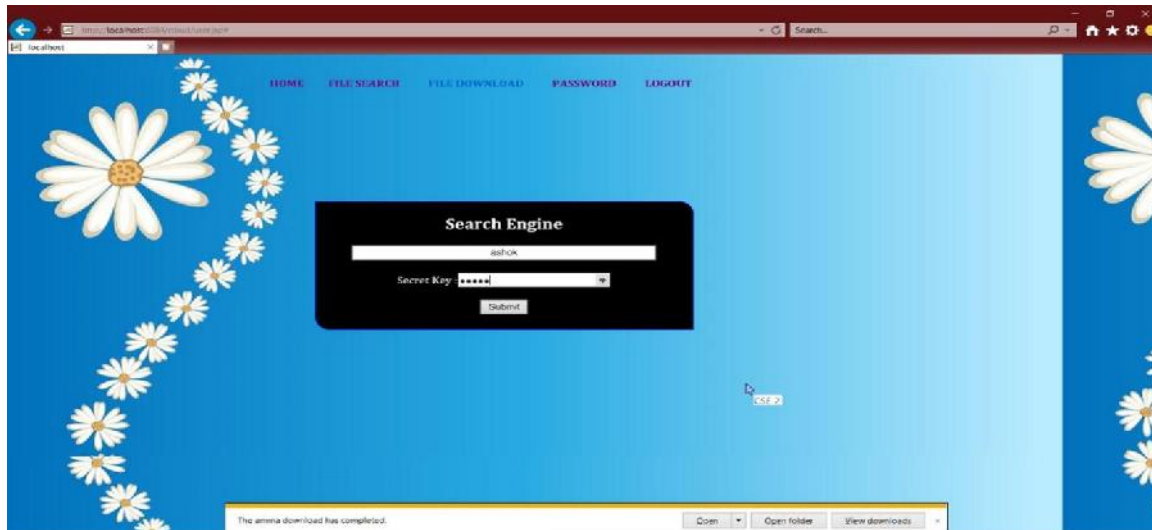


Fig.5.9. File Search

Search Result: - The below screenshot shows the view of search results as amma given as e.g. in above search file and then click on amma, send request to data owner to download the file.

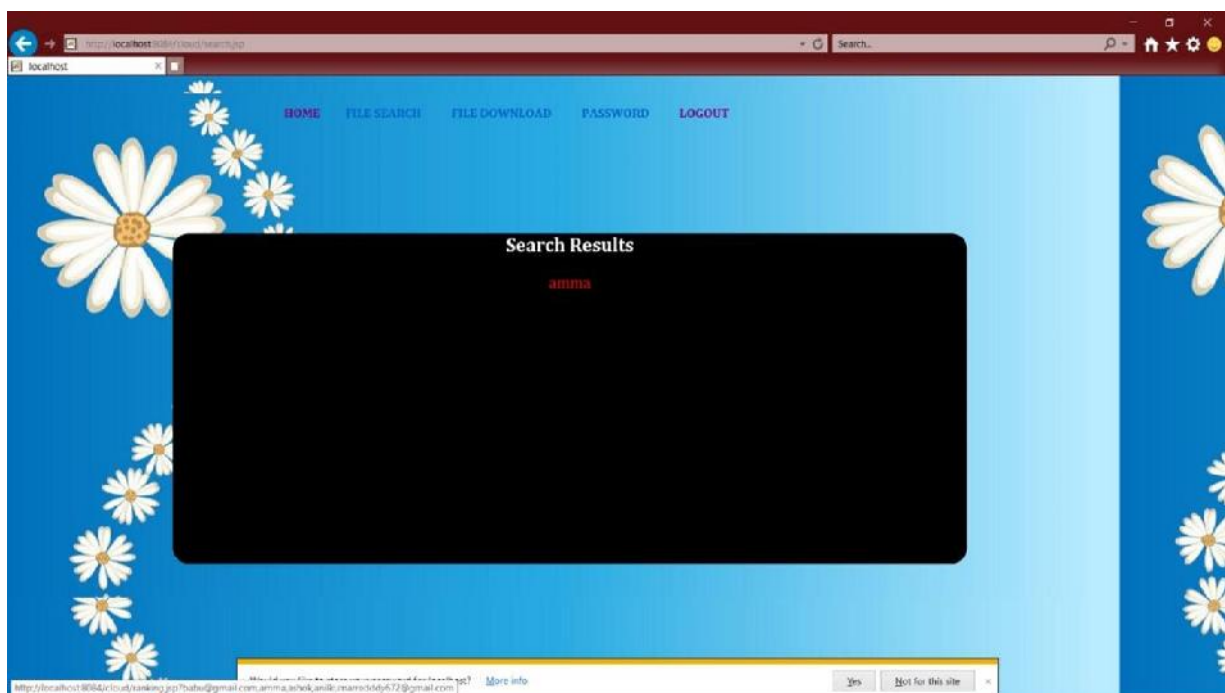
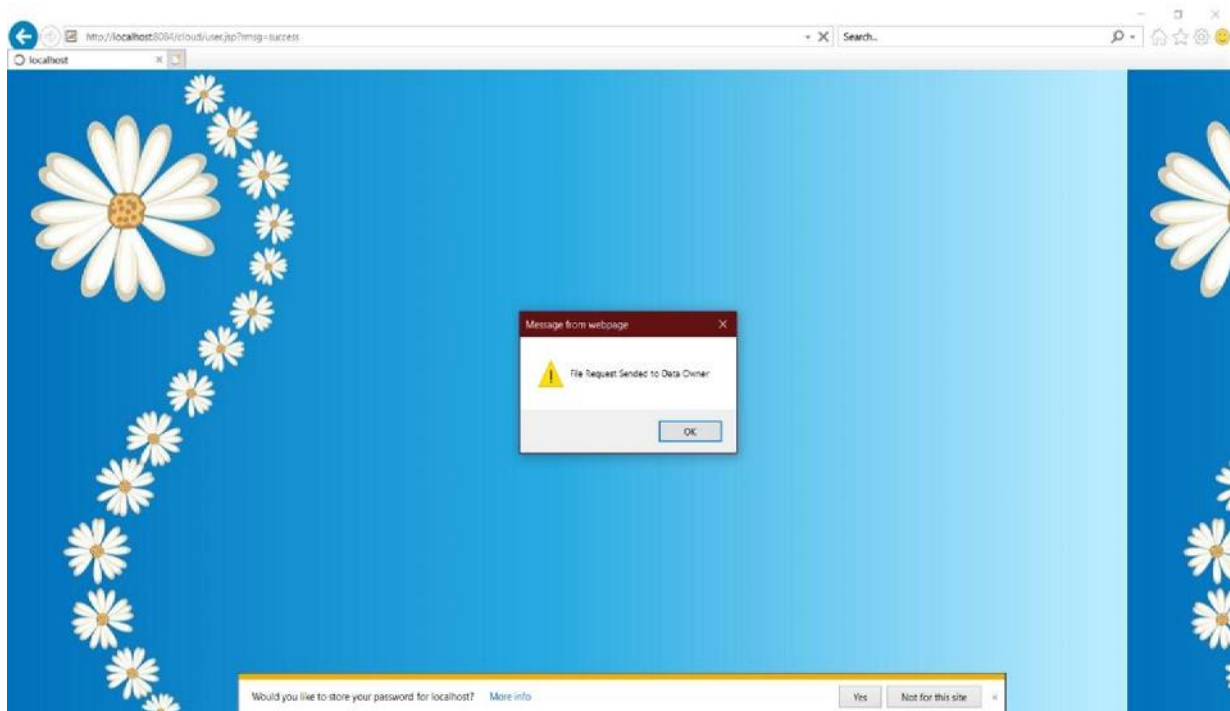
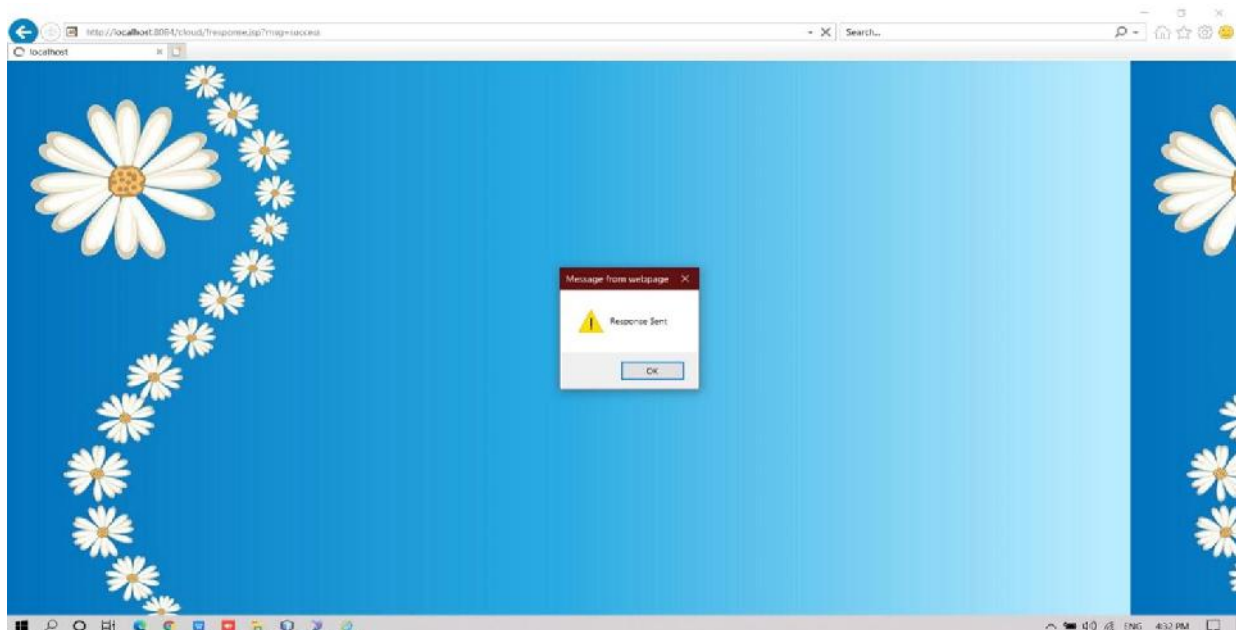


Fig.5.10. Search Result

File Request: - The below screenshot shows the view of file request sanded to data owner.



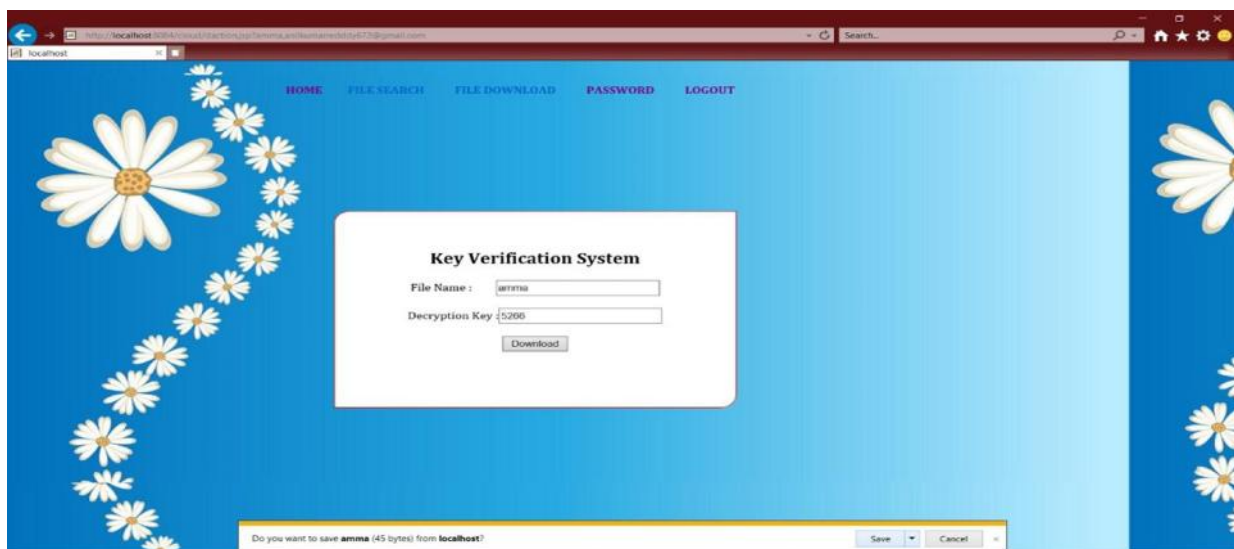
Request Accepted: - The below screenshot shows the request waiting by the data owner and request granted to user.



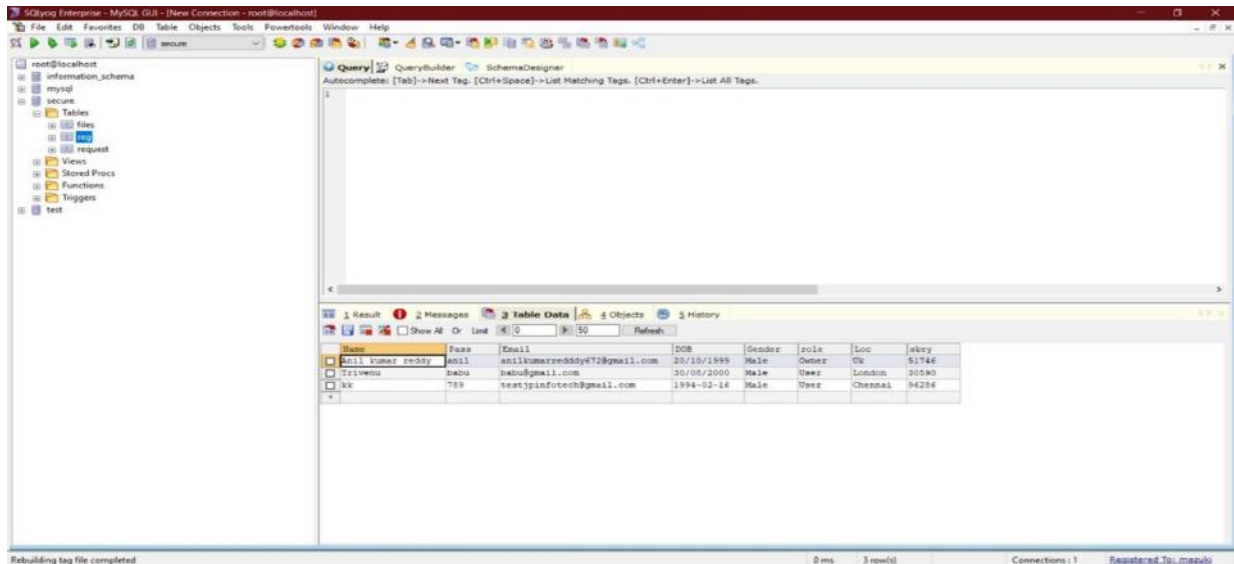
File Details: - The below screenshot shows the view of file request granted by the data owner to download the file. Click on File Download.



Key Verification system: - The below screenshot shows the view of the file download verification system decrypted automatically generated, when is in waiting list decrypted show (0000) and owner granted the request (5206) randomly.

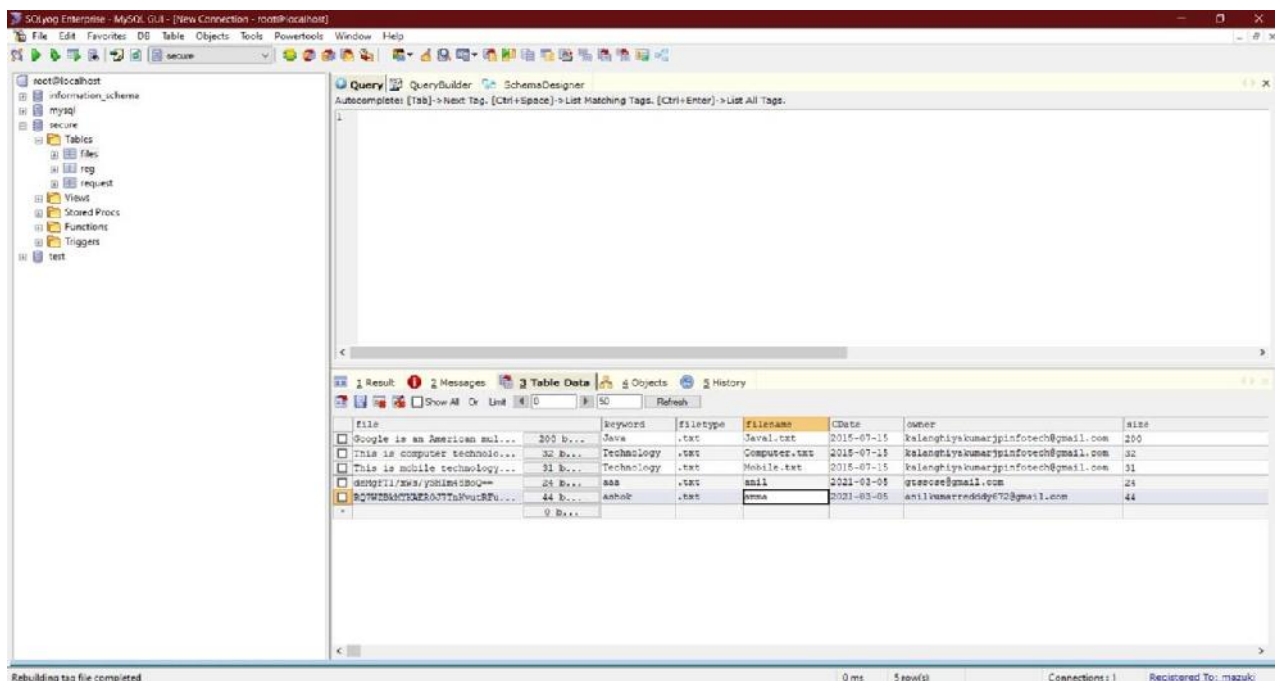


Registered Database: - The below screenshots show the view of registration of owner and user database name, DOB, mail, gender, username, password, secret key.



Name	Pass	Email	DOB	Gender	role	Loc	skkey
mail kumar reddy	anil	anilumarreddy672@gmail.com	20/10/1999	Male	Owner	Or	51744
Trivasa	babu	babu@gmail.com	20/08/2000	Male	User	London	20590
kk	789	testgpinfotech@gmail.com	1994-02-16	Male	User	Chennai	94286

File Uploaded Database: - The below screenshots show the view of data owner uploaded file.



file	keywords	filetype	filename	CDate	owner	size
Google is an American sci...	Java	.txt	Java1.txt	2015-07-15	kalengthyakumarjpinfotech@gmail.com	200
This is computer technolo...	Technology	.txt	Computer.txt	2015-07-15	kalengthyakumarjpinfotech@gmail.com	32
This is mobile technology...	Technology	.txt	Mobile.txt	2015-07-15	kalengthyakumarjpinfotech@gmail.com	31
gntp771/xw/yshmt880w=	aaa	.txt	aaa1	2021-03-05	gtsere@gmail.com	24
QW7W2B6KOTXPR677Ta9uc8F...	anahit	.txt	anahit	2021-03-05	anilumarreddy672@gmail.com	44

File Requested to Data owner Database: - The below screenshot shows the user requested file to owner granted database.

