

**An Efficient Search Method for Encrypted Cloud Data Using
Feature to Match Joint Keyword**

A Major Project Report

Submitted to the faculty of Engineering of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA

In partial fulfillment of the requirements for the awards of the degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By



G. Naga Lakshmi
(20481A1256)

G. Dinesh Kumar
(20481A1254)

B. Archana
(20485A1203)

B. Sravanthi
(20481A1213)

Under the Esteemed guidance of

Sri Y. K. Viswanadham M.Tech.,(Ph.D.,)

Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with permanent Affiliation to JNTUK, Kakinada)

SESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU-521356

ANDHRA PRADESH

2023-2024

DEPARTMENT OF INFORMATION TECHNOLOGY
SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK, KAKINADA)

SESHADRI RAO KNOWLEDGE VILLAGE
GUDLAVALLERU-521356



CERTIFICATE

This is to certify that a major project report entitled “**An Efficient Search Method for Encrypted Cloud Data Using Feature to Match Joint Keyword**” is a bonafide record of work carried out by **G. Naga Lakshmi (20481A1256), G. Dinesh Kumar (20481A1254), B. Archana (20485A1203) and B. Sravanthi (20481A1213)** under my guidance and supervision in partial fulfillment of requirements for the award of degree of Bachelor of Technology in **Information Technology** of **Jawaharlal Nehru Technological University Kakinada, Kakinada** during the year of 2023-24.

GUIDE

Sri Y. K. Viswanadham M.Tech., (Ph.D.)
Associate Professor

Head of the Department

Dr.D.N.V.S.L.S.IndiraM.Tech.,Ph.D.,
Professor& H.O.D

External Examiner

ACKNOWLEDGEMENT

We are glad to express our deep sense of gratitude to **Y.K.Viswanadham** , Associate Professor and Head of the Department in *INFORMATION TECHNOLOGY* for his guidance and cooperation in completing this major project. Through this, we want to convey our sincere thanks to him for inspiring assistance during our major project.

We express our heartfelt gratitude and deep indebtedness to our beloved Head of the Department **Dr.D.N.V.S.L.S.Indira** for her great help and encouragement in doing our major project successfully.

We also express our gratitude to our principal **Dr. B. Karuna Kumar**, for his encouragement and facilities provided during the course of major project.

We express our heartfelt gratitude to all Faculty Members, all Lab Technicians, who helped us in all aspects of lab work. We thank one and all who have rendered help to us directly or indirectly in the completion of this major project.

Project Associates

G. Naga Lakshmi (20481A1256)

G. Dinesh Kumar (20481A1254)

B. Archana (20485A1203)

B. Sravanthi (20481A1213)

ABSTRACT

With the continuous improvement of the security of cloud storage, more users upload private data to the cloud. However, a large number of encrypted data using independent keywords to create indexes not only directly increase the storage overhead, but also lead to the decline of search efficiency. Therefore, this paper proposes an efficient search method using features to match joint keywords (FMJK) on encrypted cloud data. This method proposes that each d keyword is randomly selected from the non-duplicated keywords, which are extracted from the documents of the data owner, to generate a joint keyword, and all joint keywords form a keyword dictionary. Each joint keyword is matched with the feature of the document and the query keyword respectively, and the result obtained by the former is regarded as a dimension of the document index, while the result obtained by the latter is regarded as a dimension of the query trapdoor. Finally, the BM25 algorithm is used to calculate the inner product of the document index and the trapdoor, and then sort them and the top k results are returned. Theoretical analysis and experimental results show that the proposed method is more feasible and more effective than the compared schemes.

INDEX

CONTENTS	PAGE NO
1. INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Existing System	4
2. SYSTEM ANALYSIS	5
2.1 Feasibility study	5
2.1.1 Technical Feasibility	6
2.1.2 Economic Feasibility	8
2.1.3 Operational Feasibility	6
2.2 Literature Survey	9
3. SYSTEM SPECIFICATION	15
3.1 Hardware Requirement	15
3.2 Software Requirement	15
4. SOFTWARE DESCRIPTION	17
4.1 Python	17
4.2 HTML, CSS	18
4.3 Tools	19
5. SYSTEM DESIGN	20
5.1 Use case Diagram	22
5.2. ER Diagram	23
5.3 Flow Diagram	24
5.4 Sequence Diagram	26
5.5 Class Diagram	21
6. SYSTEM IMPLEMENTATION	28
6.1 Implementing Frontend	28
6.2 Implementing Backend	30
6.3 Modules and Libraries	31
6.4 Libraries	32
6.5 Source code	35
7. SYSTEM TESTING	62
7.1 Unit Testing	63

7.2 Integration Testing	64
7.3 System Testing	66
7.4 Acceptance Testing	68
7.5 Security Testing	70
7.6 Performance Testing	71
7.7 Usability Testing	73
8. Screenshots	74
9. CONCLUSION	83
10. FUTURE ENHANCEMENT	84
11. REFERENCES	86
12. PROJECT WORK MAPPING WITH PROGRAMME OUTCOMES	88
12.1 Mapping table	90

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
5.1.1	Usecase diagram of cloud server	20
5.1.2	Usecase diagram of Data Owner	20
5.1.3	Usecase diagram of Data User	21
5.2.1	E R Diagram	22
5.3.1	Flow Diagram	23
5.4.1	Sequence Diagram of cloud server	24
5.4.2	Sequence Diagram of Data Owner	24
5.4.3	Sequence Diagram of Data User	25
5.5.1	Class Diagram	26
8.1	Data owner login page	74
8.2	Data owner registration	75
8.3	Data owner upload files	75
8.4	Joint keywords generation	76
8.5	File Encryption	77
8.6	Data User login	77
8.7	Data User registration	78
8.8	Downloading files	78
8.9	Downloading request	79
8.10	Decryption using security key	79
8.11	Cloud Server login	80
8.12	Dashboard	80
8.13	Data Owner Details	81
8.14	Data User Details	81
8.15	Documents Uploaded	82
8.16	Documents Downloaded	82

1.INTRODUCTION

1.1 introduction

With the rapid development of science and technology, enterprises or individual users increasingly rely on storing a large number of data documents on cloud servers in order to share data quickly and remotely. However, with the increasing demand, the cost of cloud server storage increases, the efficiency of search decreases, and privacy protection has become a focus of research. In most of the existing cipher-text sorting retrieval methods, KNN (K Nearest Neighbor) technology is used to create indexes supporting cipher-text retrieval. In the process of massive data encryption search, most of the search encryption schemes have high time complexity and large storage space, which are closely related to the encrypted key, the document index and query request dimensions. Reducing high dimensional data encryption is a solution to improve search efficiency. Some researchers try to study how to enrich the flexibility of retrieval, however they still cannot meet the retrieval requirements of a large number of data, and they cannot sort and filter useful data for authorized users. Therefore, in the face of different user needs, it is urgent to find a scheme that can not only guarantee privacy, but also improve retrieval efficiency and ensure query accuracy.

We propose an efficient search method using features to match joint keywords (FMJK) on encrypted cloud data based on the MRSE (Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data) scheme. First, it is necessary to extract the features of each document to accurately express its theme. Then each d keywords are randomly selected from the non-duplicated keywords, which are extracted from the documents of the data owner, to generate a joint keyword, and all joint keywords form a keyword dictionary and then the features of each document are matched with the joint keywords to create an index. The authorized user enters the query keywords to match the joint keywords to create a trapdoor, and finally calculates the safe inner product of the trapdoor and the index to return the top k results. The contributions of this article are as follows:

- (1) Each d randomly selected keywords form a joint keyword, which matches with the feature of the document to be mapped to one dimension of the index, which reduces the dimension size of the key, the index and the trapdoor, simplifies the matrix operation during encryption, and improves search efficiency.
- (2) Through the improved BM25 algorithm to calculate the inner product of the document index and the trapdoor, which not only sorts quickly but also ensures query accuracy.
- (3) The randomness of joint keywords and the encryption process of expanding and splitting ensure privacy protection.

1.2 Problem Statement

The model of the searchable encryption system consists of three different entities, which are the cloud server, the data owner, and the data user. The data owner first takes all data files as documents, and extracts keywords from these documents. Then combines each d randomly selected keywords into a joint keyword, and all joint keywords form a keyword dictionary. Thirdly, creates a document index for each document using its exacted keywords and the keyword dictionary and encrypts the document and its encrypted index by means of the key encryption. Finally, the data owner uploads the encrypted documents and their indexes to the cloud server for storage. While querying, the authorized data user inputs multiple keywords, the corresponding keyword trapdoor will be created and submitted to the cloud server. After receiving the search request, the cloud server first calculates the security inner product of the keyword trapdoor and each document index to get the score of each document, and then returns the top k encrypted documents with the highest score to the authorized the data user. After receiving the returned documents, the data user decrypts the encrypted documents using the keys provided by the data owner to obtain the required data information.

1.3 Existing System

In most of the existing cipher-text sorting retrieval methods, KNN (K Nearest Neighbor) technology is used to create indexes supporting cipher-text retrieval. In the process of massive data encryption search, most of the search encryption schemes have high time complexity and large storage space, which are closely related to the encrypted key, the document index and query request dimensions. Reducing high dimensional data encryption is a solution to improve search efficiency. Some researchers try to study how to enrich the flexibility of retrieval, however they still cannot meet the retrieval requirements of a large number of data, and they cannot sort and filter useful data for authorized users. Therefore, in the face of different user needs, it is urgent to find a scheme that can not only guarantee privacy, but also improve retrieval efficiency and ensure query accuracy.

2. SYSTEM ANALYSIS

2.1 FEASIBILITY STUDY

A feasibility study of an emotion recognition system would involve assessing the technical, economic, legal, operational, and scheduling factors related to the development and implementation of such a system. Here are some key factors that would need to be considered in a feasibility study of an emotion recognition system:

1. TECHNICAL FEASIBILITY
2. OPERATIONAL FEASIBILITY
3. ECONOMIC FEASIBILITY

A feasibility study assesses the operational, technical and economic merits of the proposed project. The feasibility study is intended to be a preliminary review of the facts to see if it is worthy of proceeding to the analysis phase. From the systems analyst perspective, the feasibility analysis is the primary tool for recommending whether to proceed to the next phase or to discontinue the project. The feasibility study is a management-oriented activity. The objective of a feasibility study is to find out if an information system project can be done and to suggest possible alternative solutions.

Projects are initiated for two broad reasons:

1. Problems that lend themselves to systems solutions
2. Opportunities for improving through:
 - (a) upgrading systems
 - (b) altering systems
 - (c) installing new systems

A feasibility study should provide management with enough information to decide:

- Whether the project can be done
- Whether the final product will benefit its intended users and organization
- What are the alternatives among which a solution will be chosen?
- Is there a preferred alternative?

TECHNICAL FEASIBILITY

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements. The analyst must find out whether current technical resources can be upgraded or added to in a manner that fulfils the

request under consideration. This is where the expertise of system analysts is beneficial, since using their own experience and their contact with vendors they will be able to answer the question of technical feasibility.

The essential questions that help in testing the operational feasibility of a system include the following:

- Is the project feasible within the limits of current technology?
- Does technology exist at all?
- Is it available within given resource constraints?
- Is it a practical proposition?
- Manpower- programmers, testers & debuggers
- Software and hardware
- Are the current technical resources sufficient for the new system?
- Can they be upgraded to provide the level of technology necessary for the new system?
- Do we possess the necessary technical expertise, and is the schedule reasonable?
- Can technology be easily applied to current problems?
- Does technology have the capacity to handle the solution?
- Do we currently possess the necessary technology?

OPERATIONAL FEASIBILITY

Operational feasibility is dependent on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented.

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

Operational feasibility reviews the willingness of the organization to support the proposed system. This is probably the most difficult of the feasibilities to gauge. In order to determine this feasibility, it is important to understand the management commitment to the proposed project. If the request was initiated by management, it is likely that there is management support, and the system will be accepted and used.

However, it is also important that the employee base accepts the change.

The essential questions that help in testing the operational feasibility of a system include the following:

- Does the current mode of operation provide adequate throughput and response time?

- Does current mode provide end users and managers with timely, pertinent, accurate and useful formatted information?
- Does the current mode of operation provide cost-effective information services to the business?
- Could there be a reduction in costs and or an increase in benefits?
- Does current mode of operation offer effective controls to protect against fraud and to guarantee accuracy and security of data and information?
- Does current mode of operation make maximum use of available resources, including people, time, and flow of forms?
- Does the current mode of operation provide reliable services?
- Are the services flexible and expandable?
- Are the current work practices and procedures adequate to support the new system?
- If the system is developed, will it be used?
- Manpower problems
- Labour objections
- Manager resistance
- Organizational conflicts and policies
- Social acceptability
- Government regulations
- Does management support the project?
- Are the users not happy with current business practices?
- Will it reduce the time (operation) considerably?
- Have the users been involved in the planning and development of the project?
- Will the proposed system really benefit the organization?
- Does the overall response increase?
- Will accessibility of information be lost?
- Will the system affect the customers in a considerable way?
- Legal aspects
- How do the end-users feel about their role in the new system?
- What end-users or managers may resist or not use the system?
- How will the working environment of the end-user change?
- Can or will end-users and management adapt to the change?

ECONOMIC FEASIBILITY

Economic analysis could also be referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

Possible questions raised in economic analysis are:

- Is the system cost effective?
- Do benefits outweigh costs?
- The cost of doing full system study
- The cost of business employee time
- Estimated cost of hardware
- Estimated cost of software/software development
- Is the project possible, given the resource constraints?
- What is the savings that will result from the system?
- Cost of employees' time for study
- Cost of packaged software/software development
- Selection among alternative financing arrangements (rent/lease/purchase)

The concerned business must be able to see the value of the investment it is pondering before committing to an entire system study. If short-term costs are not overshadowed by long-term gains or produce no immediate reduction in operating costs, then the system is not economically feasible, and the project should not proceed any further. If the expected benefits equal or exceed costs, the system can be judged to be economically feasible. Economic analysis is used for evaluating the effectiveness of the proposed system.

The economic feasibility will review the expected costs to see if they are in-line with the projected budget or if the project has an acceptable return on investment. At this point, the projected costs will only be a rough estimate. The exact costs are not required to determine economic feasibility. It is only required to determine if it is feasible that the project costs will fall within the target budget or return on investment. A rough estimate of the project schedule is required to determine if it would be feasible to complete the systems project within the required timeframe. The required timeframe would need to be set by the organization.

2.2 LITERATURE SURVEY

TITLE: VPSearch: Achieving verifiability for privacy preserving multi-keyword search over encrypted cloud data,’

ABSTRACT: Although cloud computing offers elastic computation and storage resources, it poses challenges on verifiability of computations and data privacy. In this work we investigate verifiability for privacy-preserving multi-keyword search over outsourced documents. As the cloud server may return incorrect results due to system faults or incentive to reduce computation cost, it is critical to offer verifiability of search results and privacy protection for outsourced data at the same time. To fulfill these requirements, we designed a Verifiable Privacy-preserving keyword Search scheme, called VPSearch, by integrating an adapted homomorphic MAC technique with a privacy-preserving multi-keyword search scheme. The proposed scheme enables the client to verify search results efficiently without storing a local copy of the outsourced data. We also propose a random challenge technique with ordering for verifying top- k search results, which can detect incorrect top- k results with probability close to 1. We provide detailed analysis on security, verifiability, privacy, and efficiency of the proposed scheme. Finally, we implement VPSearch using Matlab and evaluate its performance over three UCI bag-of-words data sets. Experiment results show that authentication tag generation incurs about 3 percent overhead only and a search query over 300,000 documents takes about 0.98 seconds on a laptop. To verify 300,000 similarity scores for one query, VPSearch costs only 0.29 seconds.

TITLE: “Blockchain based verifiable multi-keyword ranked search on encrypted cloud with fair payment,’

ABSTRACT: In traditional cloud computing system, searchable encryption is deemed as a core technology to realize data confidentiality protection and information retrieval functions. However, the online payment problem and mutual distrust between cloud platforms and users may hinder the wide adoption of cloud services. In this paper, we construct a blockchain based multi-keyword ranked search with fair payment (BMFP) system, which leverages smart contracts to verify the correctness and completeness of the search result, and automatically execute the fair payment operations. The system realizes public verifiability on a multi-keyword ranked search result. The data owner manages the search authority, and a concrete fair payment smart contract is designed. The BMFP is compatible with Ethereum, and the verification algorithm executed by the smart contract is cost-efficient.

TITLE: ‘Privacy-preserving multi keyword ranked search over encrypted cloud data,’

ABSTRACT: With the rapid development of cloud computing services, more and more individuals and enterprises prefer to outsource their data or computing to clouds. In order to preserve data privacy, the data should be encrypted before outsourcing and it is a challenge to perform searches over encrypted data. We propose a privacy-preserving multi-keyword ranked search scheme over encrypted data in hybrid clouds, which is denoted as MRSE-HC. The keyword dictionary of documents is clustered into balanced partitions by a bisecting k-means clustering based keyword partition algorithm. According to the partitions, the keyword partition-based bit vectors are adopted for documents and queries which are utilized as the index of searches. The private cloud filters out the candidate documents by the keyword partition-based bit vectors, and then the public cloud uses the trapdoor to determine the result in the candidates. On the basis of the MRSE-HC scheme, an enhancement scheme EMRSE-HC is proposed, which adds complete binary pruning tree to further improve search efficiency. The security analysis and performance evaluation show that MRSE-HC and EMRSE-HC are privacy-preserving multi-keyword ranked search schemes for hybrid clouds and outperform the existing scheme FMRS in terms of search efficiency.

TITLE: ‘Secure kNN computation on encrypted databases,’

ABSTRACT: Service providers like Google and Amazon are moving into the SaaS (Software as a Service) business. They turn their huge infrastructure into a cloud-computing environment and aggressively recruit businesses to run applications on their platforms. To enforce security and privacy on such a service model, we need to protect the data running on the platform. Unfortunately, traditional encryption methods that aim at providing "unbreakable" protection are often not adequate because they do not support the execution of applications such as database queries on the encrypted data. In this paper we discuss the general problem of secure computation on an encrypted database and propose a SCONEDB (Secure Computation ON an Encrypted DataBase) model, which captures the execution and security requirements. As a case study, we focus on the problem of k-nearest neighbor (kNN) computation on an encrypted database. We developed a new asymmetric scalar-product-preserving encryption (ASPE) that preserves a special type of scalar product. We use ASPE to construct two secure schemes that support kNN computation on encrypted data; each of these schemes is shown to resist practical attacks of a different background knowledge level, at a different overhead cost. Extensive performance studies are carried out to evaluate the overhead and the efficiency of the schemes.

TITLE: “Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing,”

ABSTRACT: Cloud computing is becoming increasingly popular. A large number of data is outsourced to the cloud by data owners motivated to access the large-scale computing resources and economic savings. To protect data privacy, the sensitive data should be encrypted by the data owner before outsourcing, which makes the traditional and efficient plaintext keyword search technique useless. So how to design an efficient, in the two aspects of accuracy and efficiency, searchable encryption scheme over encrypted cloud data is a very challenging task. In this paper, for the first time, we propose a practical, efficient, and flexible searchable encryption scheme which supports both multi-keywords ranked search and parallel search. To support multi-keyword search and result relevance ranking, we adopt Vector Space Model (VSM) to build the searchable index to achieve accurate search results. To improve search efficiency, we designed a tree-based index structure which supports parallel search to take advantage of the powerful computing capacity and resources of the cloud server. With our designed parallel search algorithm, the search efficiency is well improved. We propose two secure searchable encryption schemes to meet different privacy requirements in two threat models. Extensive experiments on the real-world dataset validate our analysis and show that our proposed solution is very efficient and effective in supporting multi-keyword ranked parallel searches.

TITLE: “A secure and dynamic multikeyword ranked search scheme over encrypted cloud data,”

ABSTRACT: Mobile cloud computing has entered a new era of technology which would help in keeping large quantity of data at a lesser cost. Platforms for accessing and computing large amounts of data are much more viable than in previous days. Researchers have been using large volumes of data for producing outputs and they consider data as an asset. Now a days data is being outsourced for conducting research in different sectors like market study, military purposes, tackling terrorist attacks, whether forecasting etc this increased the value of data, and it has to be kept safe. To preserve the nature of data high level encryption should be done while transferring the data but this may cause problems when multiple users try to access data from a cloud source. Researchers are trying to develop new encryption techniques to reduce the risk of security while transferring the data. Solutions that are being discussed are a multi keyword search scheme supporting result ranking by adopting k nearest neighbors knn technique and another is a dynamic searchable encryption scheme through blind storage to conceal access pattern of the search user.

TITLE: ‘Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking,’

ABSTRACT: With the growing popularity of cloud computing, a huge amount of documents are outsourced to the cloud for reduced management cost and ease of access. Although encryption helps protect user data confidentiality, it leaves the well-functioning yet practically efficient secure search functions over encrypted data a challenging problem. In this paper, we present a verifiable privacy-preserving multi-keyword text search (MTS) scheme with similarity-based ranking to address this problem. To support multi-keyword search and search result ranking, we propose to build the search index based on term frequency- and the vector space model with cosine similarity measure to achieve higher search result accuracy. To improve the search efficiency, we propose a tree-based index structure and various adaptive methods for multi-dimensional (MD) algorithm so that the practical search efficiency is much better than that of linear search. To further enhance the search privacy, we propose two secure index schemes to meet the stringent privacy requirements under strong threat models, i.e., known ciphertext model and known background model. In addition, we devise a scheme upon the proposed index tree structure to enable authenticity check over the returned search results. Finally, we demonstrate the effectiveness and efficiency of the proposed schemes through extensive experimental evaluation.

TITLE: “A novel fast dimension-reducing ranked query method with high security for encrypted cloud data,”

ABSTRACT: We propose a novel Fast dimension reducing ranked query method (FDRQM) with high security for encrypted cloud data. We use Principal component analysis (PCA) algorithm to improve the speed of data encryption and search efficiency. Moreover, a random threshold of accumulated contribution rate of principal components is set to realize the randomness of data dimension reduction and improve data security further. Besides, we introduce a unit matrix before dimension reduction of index, which can not only improve the security of the system, but also ensure the accuracy of query. We demonstrate that our algorithm is more effective and efficient than existing algorithms.

TITLE: “Catch you if you misbehave: Ranked keyword search results verification in cloud computing,’

ABSTRACT: With the advent of cloud computing, more and more people tend to outsource their data to the cloud. As a fundamental data utilization, secure keyword search over encrypted cloud data has attracted the interest of many researchers recently. However, most of existing research is

based on an ideal assumption that the cloud server is “curious but honest”, where the search results are not verified. In this paper, we consider a more challenging model, where the cloud server would probably behave dishonestly. Based on this model, we explore the problem of result verification for the secure ranked keyword search. Different from previous data verification schemes, we propose a novel deterrent-based scheme. With our carefully devised verification data, the cloud server cannot know which data owners, or how many data owners exchange anchor data which will be used for verifying the cloud server's misbehavior. With our systematically designed verification construction, the cloud server cannot know which data owners' data are embedded in the verification data buffer, or how many data owners' verification data are used for verification. All the cloud server knows is that, once he behaves dishonestly, he would be discovered with a high probability, and punished seriously once discovered. Furthermore, we propose to optimize the value of parameters used in the construction of the secret verification data buffer. Finally, with thorough analysis and extensive experiments, we confirm the efficacy and efficiency of our proposed schemes.

TITLE: ‘Cross-lingual multi-keyword rank search with semantic extension over encrypted data,’

ABSTRACT: With the advent of cloud computing, more and more people tend to outsource their data to the cloud. As a fundamental data utilization, secure keyword search over encrypted cloud data has attracted the interest of many researchers recently. However, most of existing research is based on an ideal assumption that the cloud server is “curious but honest”, where the search results are not verified. In this paper, we consider a more challenging model, where the cloud server would probably behave dishonestly. Based on this model, we explore the problem of result verification for the secure ranked keyword search. Different from previous data verification schemes, we propose a novel deterrent-based scheme. With our carefully devised verification data, the cloud server cannot know which data owners, or how many data owners exchange anchor data which will be used for verifying the cloud server's misbehavior. With our systematically designed verification construction, the cloud server cannot know which data owners' data are embedded in the verification data buffer, or how many data owners' verification data are actually used for verification. All the cloud server knows is that, once he behaves dishonestly, he would be discovered with a high probability, and punished seriously once discovered. Furthermore, we propose to optimize the value of parameters used in the construction of the secret verification data buffer. Finally, with thorough analysis and extensive experiments, we confirm the efficacy and efficiency of our proposed schemes.

3.SYSTEM SPECIFICATION

The purpose of system requirement specification is to produce the specification analysis of the task and also to establish complete information about the requirement, behavior and other constraints such as functional performance and so on. The goal of system requirement specification is to completely specify the technical requirements for the product in a concise and unambiguous manner.

3.1 HARDWARE REQUIREMENTS

- Processor - Intel i5 above
- Speed - 3.19 GHZ
- RAM - minimum 4GB
- SSD/HHD - 256 GB above

3.2 SOFTWARE REQUIREMENTS

- Operating System - Windows 10
- Development Software - Python 3.10
- Front End - HTML5, CSS, Java Script
- Back End - Django, MySQL
- Database Software - XAMPP
- Tool - Pycharm

4. SOFTWARE DESCRIPTION

4.1 PYTHON

Python is a high-level programming language that is widely used for web development, scientific computing, data analysis, artificial intelligence, machine learning, and other applications. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in the world, due to its simplicity, readability, and versatility.

Python has a large standard library, which makes it easy to develop complex applications quickly. The language also supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python code is interpreted, rather than compiled, which allows for rapid development and testing.

Some key features of Python include:

- Simple and easy to learn syntax, which makes it a good language for beginners.
- Dynamically typed, which means that variable types are determined at runtime, rather than during compilation.
- Automatic memory management, which makes it easy to write complex programs without worrying about memory allocation and deallocation.
- Cross-platform compatibility, which means that Python programs can run on different operating systems, such as Windows, Mac, and Linux.
- Large and active community, which provides extensive documentation, tutorials, and support for Python users.
- Python also has numerous libraries and frameworks that make it easy to develop complex applications in various domains. For example, some popular libraries for data analysis include NumPy, Pandas, and Matplotlib, while popular web development frameworks include Django and Flask.

It is the procedure of describing the construction, components, modules, interfaces, and data for a system to satisfy specified requirements. One could understand it as the solicitation of systems theory to product development. There is nearly join with the disciplines of system analysis, systems architecture and systems engineering.

4.2 HTML, CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are two core technologies used to create and design websites.

HTML is a markup language that defines the structure and content of a webpage. It consists of a set of tags and attributes that describe the various elements of a webpage, such as headings, paragraphs, images, links, forms, and tables. HTML tags are used to indicate the beginning and end of an element and can be nested to create more complex structures.

CSS is a style sheet language that is used to define the visual appearance and layout of a webpage. It consists of a set of rules that specify how HTML elements should be displayed, such as the color, font, size, position, and spacing. CSS rules are applied to HTML elements using selectors, which identify the elements to which the rules should be applied.

HTML and CSS are typically edited using a text editor, such as Sublime Text, Atom, or Visual Studio Code. The code is saved as a text file with the extension **.html** for HTML files and **.css** for CSS files.

There are also several software tools available for creating and editing HTML and CSS, such as:

- **Adobe Dreamweaver:** A web design and development tool that includes a visual editor and a code editor for creating and editing HTML and CSS.
- **Notepad++:** A free source code editor for Windows that supports several programming languages including HTML and CSS.
- **Brackets:** A free open-source text editor for web design and development that supports live preview of HTML and CSS.
- **Visual Studio:** A full-featured integrated development environment (IDE) that supports web development with HTML and CSS.
- **Web-flow:** A drag-and-drop website builder that generates HTML and CSS code automatically based on the user's design.

4.3 TOOLS

PYCHARM

PyCharm is an integrated development environment (IDE) for the Python programming language. It is designed to help developers write and debug Python code more efficiently and effectively.

Here are some of the key features of PyCharm:

Code editor: PyCharm includes a code editor that provides syntax highlighting, code completion, and code analysis. It also supports code refactoring, which allows you to easily rename variables, functions, and classes.

Debugger: PyCharm includes a powerful debugger that allows you to step through your code line by line, set breakpoints, and inspect variables and data structures.

Version control: PyCharm integrates with popular version control systems, such as Git, allowing you to manage your code and collaborate with other developers more effectively.

Test runner: PyCharm includes a built-in test runner that allows you to write and run tests for your code. It supports popular testing frameworks, such as pytest and unittest.

Profiler: PyCharm includes a profiler that allows you to analyze the performance of your code, identify bottlenecks, and optimize your code for better performance.

Plugin system: PyCharm supports a wide range of plugins that can be installed to extend its functionality. This includes plugins for web development, scientific computing, and machine learning, among others.

5. SYSTEM DESIGN

It is the procedure of describing the construction, components, modules, interfaces, and data for a system to satisfy specified requirements. One could understand it as the solicitation of systems theory to product development. There is nearly join with the disciplines of system analysis, systems architecture and systems engineering.

5.1 USECASE DIAGRAM

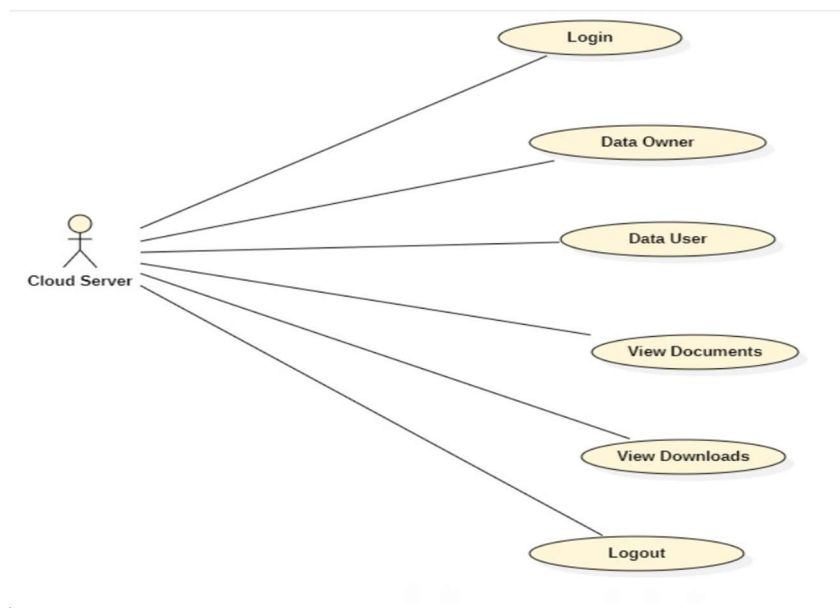


Fig 5.1.1 Usecase Diagram of Cloud Server

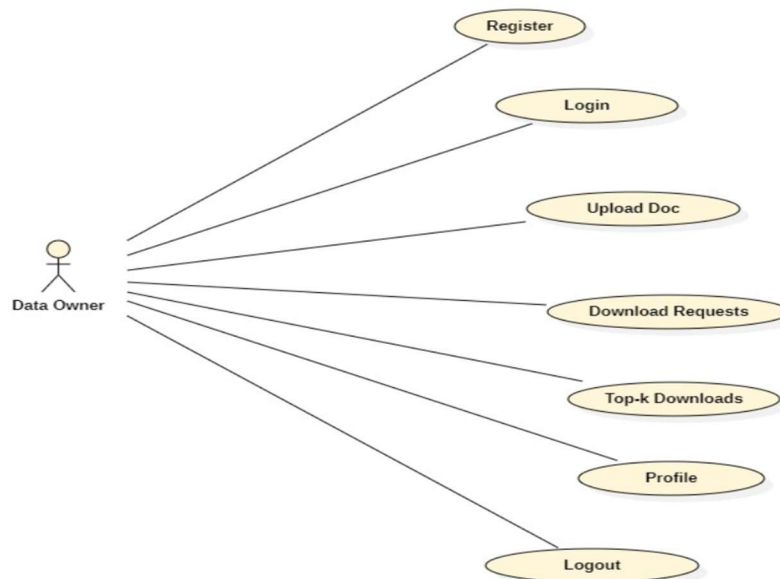


Fig 5.1.2 Usecase Diagram of Data Owner

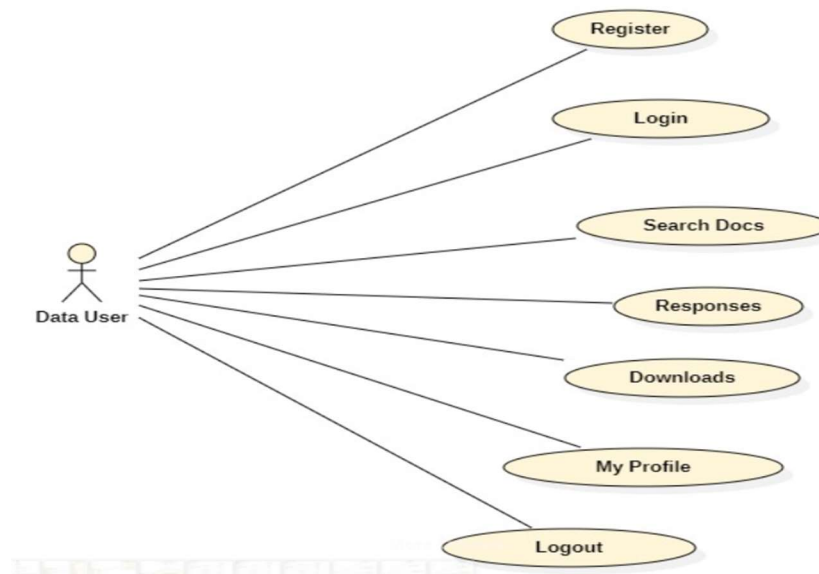


Fig 5.1.3 Usecase Diagram of Data User

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

5.2 ER DIAGRAM

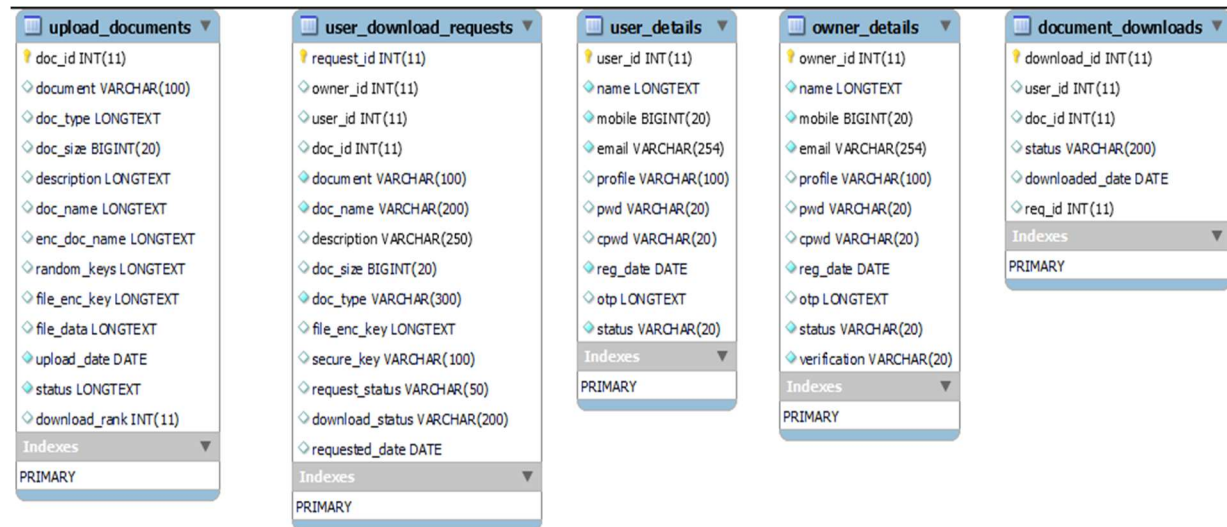


Fig 5.2.1 ER diagram

ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.

5.3 FLOW DIAGRAM

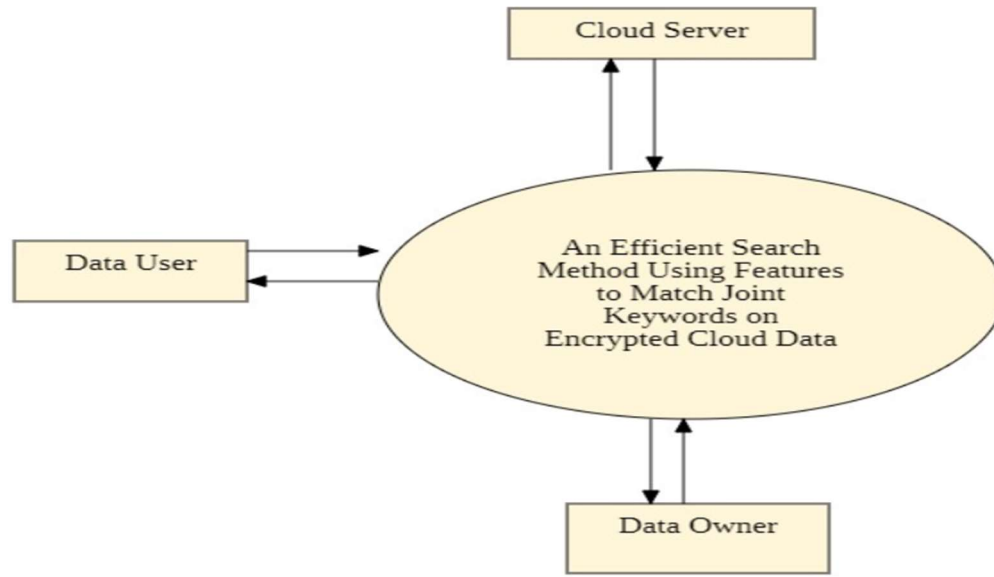


Fig 5.3.1 Flow Diagram

A flow diagram, also known as a flowchart, is a visual representation of a process or workflow. It uses symbols and arrows to illustrate the steps or actions in the process and the flow of information or materials between them.

5.4 SEQUENCE DIAGRAM

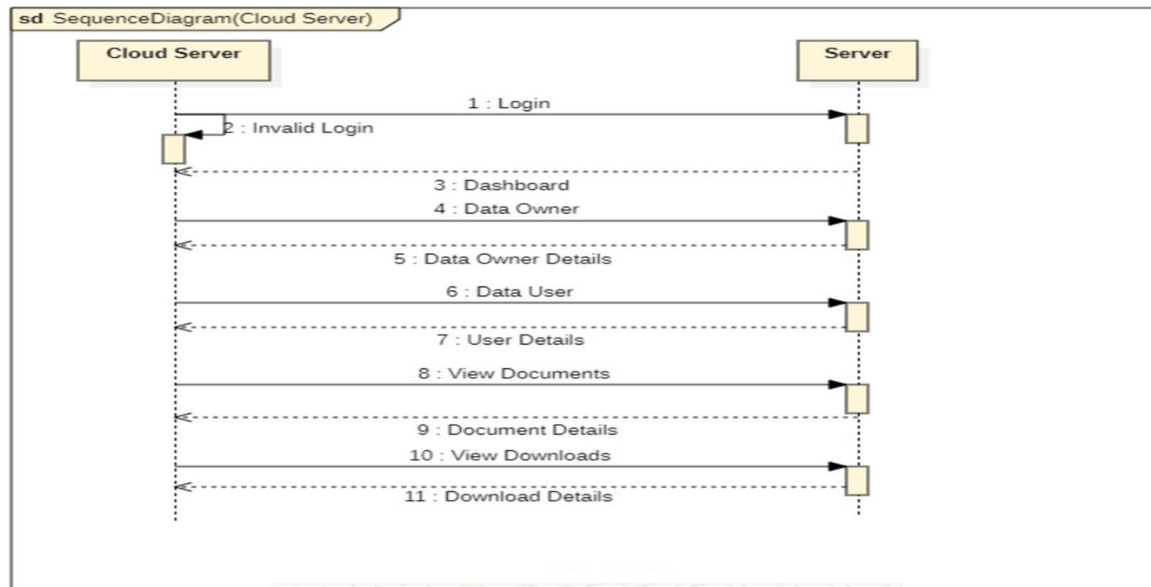


Fig 5.4.1 Sequence Diagram of Cloud Server

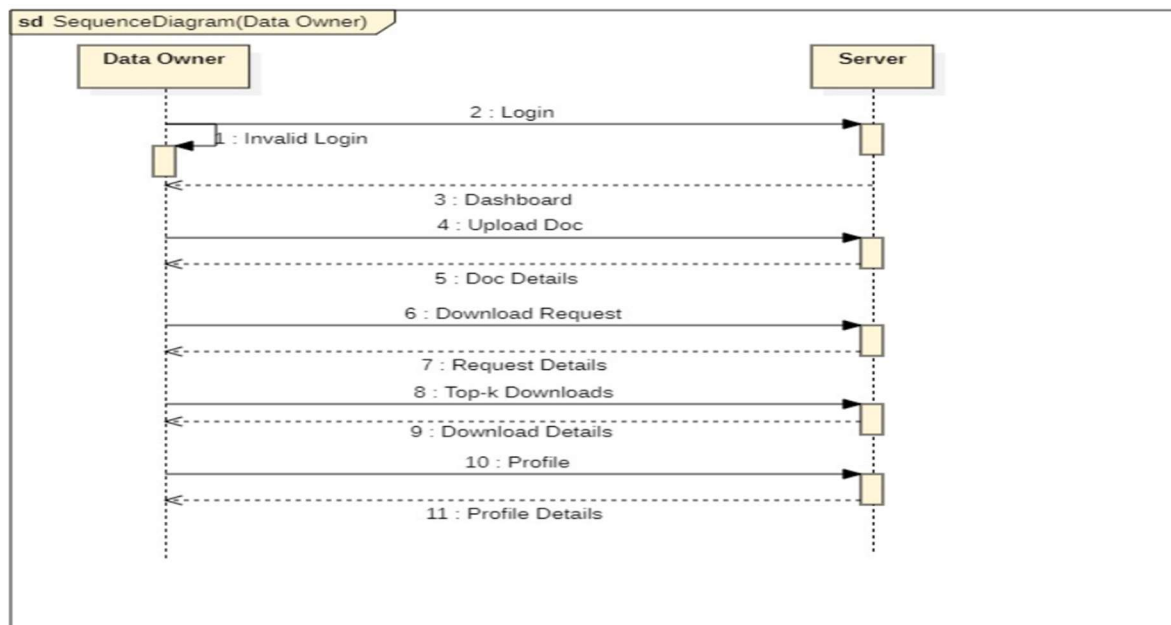


Fig 5.4.2 Sequence Diagram of Data owner

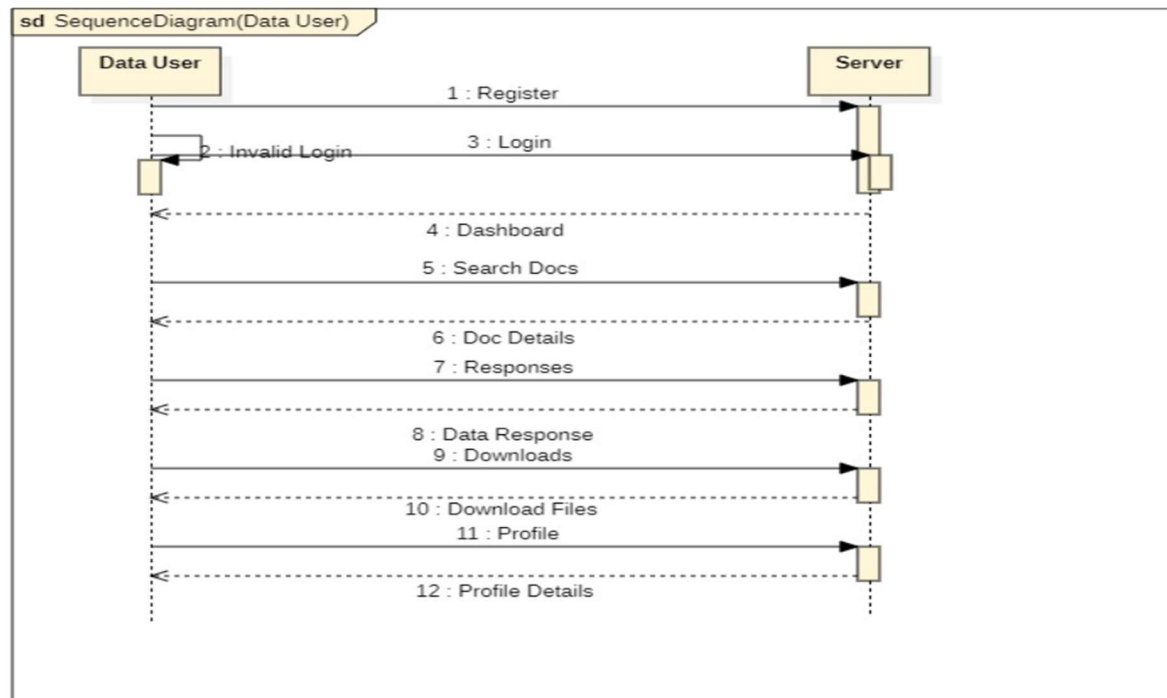


Fig 5.4.3 Diagram of Data User

A sequence diagram is a type of UML (Unified Modeling Language) diagram that is used to model the interactions between objects or components in a system. It shows the sequence of messages exchanged between the objects and the order in which they occur. Sequence diagrams are useful for understanding the flow of a system, identifying potential bottlenecks, and testing system interactions.

5.5 CLASS DIAGRAM

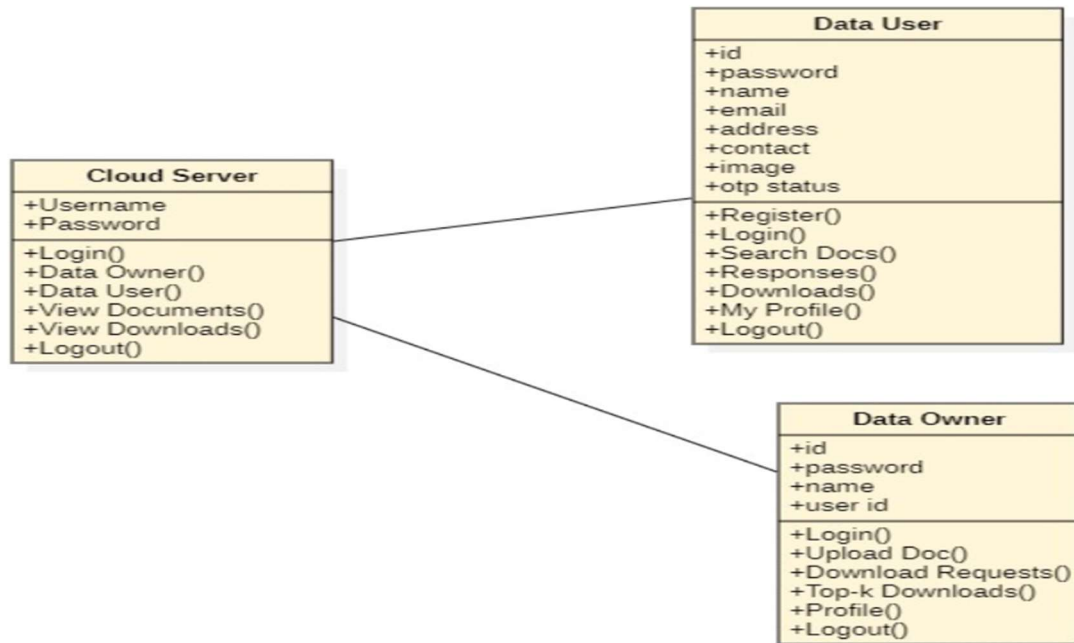


Fig 5.5.1 Class Diagram

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

5.6 SYSTEM ARCHITECTURE:

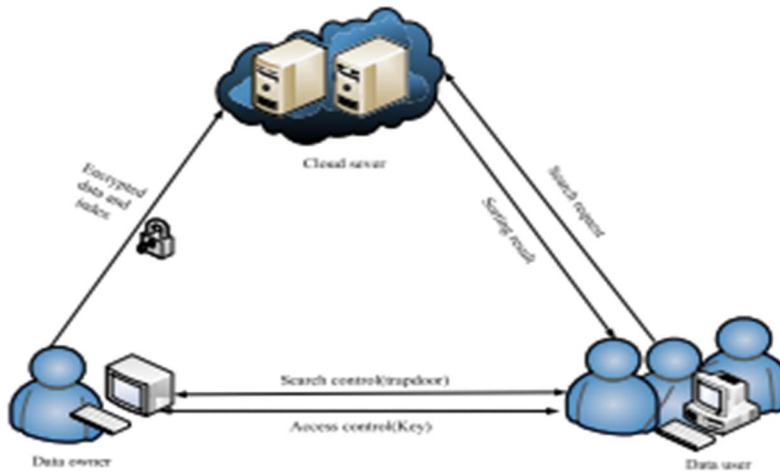


Fig 5.6.1 System architecture

6. SYSTEM IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned-out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

6.1 IMPLEMENTING FRONT ENDHTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colors, layout, and fonts, thus making our web pages presentable to the users. CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language.

ACRONYM

Cascading: Falling of Styles.

Style : Adding designs/Styling our HTML tags. Sheets :

Writing our style in different documents.

Java Script:

JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval), object introspection (via for...in and Object utilities), and source-code recovery (JavaScript functions store their source text and can be retrieved through toString())

6.2 IMPLEMENTING BACK END

MYSQL:

MySQL is an open-source relational database management system (RDBMS). A relational database organizes data into one or more data tables in which data may be related to each other; these relations help structure the data. SQL is a language that programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often, MySQL is used with other programs to implement applications that need relational database capability. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is used by many popular websites such as Twitter, Facebook, Youtube.

Django FRAMEWORK:

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Ridiculously fast. Django was designed to help developers take applications from concept to completion as quickly as possible.

Reassuringly secure. Django takes security seriously and helps developers avoid many common security mistakes.

Exceedingly scalable.

Some of the busiest sites on the web leverage Django's ability to quickly and flexibly scale.

6.3 MODULES AND LIBRARIES

There are 3 modules:

1. Data User
2. Data Owner
3. Cloud Server

Data Owner: -

- Register
- Login
- Upload Doc
- Download Requests
- Top-K Download Files
- Profile
- Logout

Data User: -

- Register
- Login
- Search Docs
- Responses
- Downloads
- My Profile
- Logout

Cloud Server: -

- Login
- Data Users
- Data Owners
- View Documents
- View Download

6.3 LIBRARIES

Sys : Provides functions and variables which are used to manipulate.

Matplotlib : To display the result of our predictive outcome.

Os : To access the file system to read the image from the train and test directory from our machines

Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPython and Tkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open-source equivalent of MATLAB.

OS :

The OS module in Python comes with various functions that enable developers to interact with the Operating system that they are currently working on. In this article we'll be learning mainly to create and delete a directory/folder, rename a directory and even basics of file handling.

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality. It is built on top of NumPy, SciPy, and matplotlib, and it integrates well with other Python libraries such as pandas for data manipulation.

Django libraries:

Django Import Export

Project Example: Tablib

Description: Facilitates seamless data import and export processes, supporting multiple formats.

Common use cases: One of the best Django packages for migrating data between a variety of environments or systems and bulk data operations.

Benefits and considerations: Streamlines data operations but demands cautious handling to prevent data inconsistencies.

Django shortcut functions

The package `django.shortcuts` collects helper functions and classes that “span” multiple levels of MVC. In other words, these functions/classes introduce controlled coupling for convenience’s sake.

`render () :`

`render (request, template_name, context=None, content_type=None, status=None, using=None)`

Combines a given template with a given context dictionary and returns an `HttpResponse` object with that rendered text.

Django does not provide a shortcut function which returns a `TemplateResponse` because the constructor of `TemplateResponse` offers the same level of convenience as `render ()`.

Required arguments.

Request:

The request object used to generate this response.

`template_name`

The full name of a template to use or sequence of template names. If a sequence is given, the first template that exists will be used. See the template loading documentation for more information on how templates are found.

Optional arguments:

Context:

A dictionary of values to add to the template context. By default, this is an empty dictionary. If a value in the dictionary is callable, the view will call it just before rendering the template.

`content_type:`

The MIME type to use for the resulting document. Defaults to 'text/html'.

Status:

The status code for the response. Defaults to 200.

Using:

The NAME of a template engine to use for loading the template.

Cryptography:

cryptography is a package which provides cryptographic recipes and primitives to Python developers. Our goal is for it to be your “cryptographic standard library”. It supports Python 3.7+ and PyPy3 7.3.11+.

cryptography includes both high level recipes and low-level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions.

XAMPP:

XAMPP is a cross-platform web server that is free and open-source. XAMPP is a short form for Cross-Platform, Apache, MySQL, PHP, and Perl. XAMPP is a popular cross-platform web server that allows programmers to write and test their code on a local webserver. It was created by Apache Friends, and the public can revise or modify its native source code. It includes MariaDB, Apache HTTP Server, and interpreters for PHP and Perl, among other computer languages. Because of XAMPP’s simplicity of deployment, a developer can quickly and easily install a WAMP or LAMP stack on an operating system, with the added benefit that common add-in apps like WordPress and Joomla can also be loaded.

Need for a XAMPP

XAMPP is simply a local host or server.

This local server runs on your personal computer, whether it’s a desktop or a laptop.

It is used to test clients or websites before publishing them to a remote web server.

On a local computer, the XAMPP server software provides a suitable environment for testing MYSQL, PHP, Apache, and Perl projects. Because most real-world web server deployments share the same components as XAMPP, moving from a local test server to a live server is straightforward.

6.5. SOURCE CODE

APPLICATION

Main app:

Views.py:

```
from django.shortcuts import render,redirect
# Create your views here.
def main_index(request):
    return render(request,'main/index.html')
def about(request):
    return render(request,'main/about.html')
```

app.py:

```
from django.apps import AppConfig
class MainappConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'mainapp'
```

Data user:

Views.py:

```
from asyncio.windows_events import NULL
from distutils.command.upload import upload
from xml.dom.minidom import Document
from pyexpat.errors import messages
from django.shortcuts import get_object_or_404, render,redirect
import random
import requests
from django.contrib import messages
from userapp.models import UserModel,DownloadRequestModel,DownloadsModel
from ownerapp.models import UploadDocumentsModel,OwnerModel
from django.db.models import Q,F
from cryptography.fernet import Fernet
import base64
import csv
from FileSecure.settings import DEFAULT_FROM_EMAIL
from django.core.mail import EmailMultiAlternatives
import urllib.request
```

```

from django.core.files.storage import FileSystemStorage
from FileSecure import check_internet
# Create your views here.
def user_register(request):
    if request.method == 'POST' and request.FILES["profile"]:
        print("post method")
        name = request.POST.get('name')
        mobile = request.POST.get('mobile')
        email = request.POST.get('email')
        profile = request.FILES['profile']
        pwd = request.POST.get('pwd')
        cpwd = request.POST.get('cpwd')
        if UserModel.objects.filter(email=email,status="verified").exists():
            print("email already exists")
            messages.error(request,"already taken")
            return redirect('user-register')
        elif UserModel.objects.filter(email=email,status="pending").exists():
            messages.warning(request,"Already Registered, Just verify your account!")
            if pwd == cpwd:
                otp=random.randint(2222,4444)
                UserModel.objects.filter(email=email,status='pending').update(otp=otp)
                if check_internet.connect():
                    html_content = "<br/> WellCome To Data User, <br/> <p> This Message Sent From
FileSecure You Have Recieved a OTP <strong> " +str(otp)+ " </strong> on FileSecure Cloud Based File
Services. </strong> <b> You Can Use online services 24/7 <strong> Thank You For Your
Registration.</p>"
                    from_mail = DEFAULT_FROM_EMAIL
                    to_mail = [email]
                    # if send_mail(subject,message,from_mail,to_mail):
                    msg = EmailMultiAlternatives("Account Registration Status", html_content, from_mail,
to_mail)
                    msg.attach_alternative(html_content, "text/html")
                    print(email)
                    if msg.send()

```



```

print("Sent")
print('your success')
# url = "https://www.fast2sms.com/dev/bulkV2"
# # create a dictionary
# my_data = {
# # Your default Sender ID

```

Models.py:

```

from django.db import models
from ownerapp.models import OwnerModel, UploadDocumentsModel
# Create your models here.
class UserModel(models.Model):
    user_id = models.AutoField(primary_key=True)
    name = models.TextField(max_length=200,default=True)
    mobile = models.BigIntegerField(default=True)
    email = models.EmailField(default=True)
    profile = models.ImageField(upload_to='user/',null=True)
    pwd = models.CharField(max_length=20,null=True)
    cpwd = models.CharField(max_length=20,null=True)
    reg_date = models.DateField(auto_now_add=True)
    otp = models.TextField(null=True)
    status = models.CharField(default="pending",max_length=20)
class Meta:
    db_table = 'user_details'
class DownloadRequestModel(models.Model):
    request_id=models.AutoField(primary_key=True)
    owner_id=models.IntegerField(null=True)
    user_id=models.IntegerField(null=True)
    doc_id=models.IntegerField(null=True)
    document=models.FileField(upload_to='files/')
    doc_name=models.CharField(max_length=200)
    description=models.CharField(max_length=250, null=True)
    doc_size=models.BigIntegerField(null=True)
    doc_type=models.CharField(max_length=300)
    # search_rank = models.CharField(default="0",null=True, max_length=200)

```

```

# download_rank = models.CharField(default="0",null=True, max_length=250)
# otp=models.IntegerField(null=True)
file_enc_key = models.TextField(null=True)
secure_key = models.CharField(max_length=100,null=True)
request_status=models.CharField(max_length=50,default='pending',null="True")
download_status = models.CharField(max_length=200,default='pending',null=True)
requested_date=models.DateField(auto_now_add=True, null=True)
class Meta:
    db_table = "user_download_requests"
class DownloadsModel(models.Model):
    download_id = models.AutoField(primary_key=True)
    user_id = models.IntegerField(null=True)
    doc_id = models.IntegerField(null=True)
    req_id = models.IntegerField(null=True)
    status = models.CharField(default="pending",null=True,max_length=200)
    # download_rank = models.IntegerField(null=True)
    downloaded_date=models.DateField(auto_now_add=True, null=True)
class Meta:
    db_table = "document_downloads"
app.py
from django.apps import AppConfig
class UserappConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'userapp'
Migration.py
from django.db import migrations
class Migration(migrations.Migration):
    dependencies = [
        ('userapp', '0002_downloadsmodel_req_id'),]
    operations = [
        migrations.RemoveField(
            model_name='downloadsmodel',
            name='download_rank',
        ),

```

```

]
from django.db import migrations, models
class Migration(migrations.Migration):
dependencies = [
    ('userapp', '0001_initial'),
]
operations = [
    migrations.AddField(
        model_name='downloadsmodel',
        name='req_id',
        field=models.IntegerField(null=True),
    ),
]
__init__.py:
from django.db import migrations, models
class Migration(migrations.Migration):
    initial = True
dependencies = [

operations = [
    migrations.CreateModel(
        name='DownloadRequestModel',
        fields=[
            ('request_id', models.AutoField(primary_key=True, serialize=False)),
            ('owner_id', models.IntegerField(null=True)),
            ('user_id', models.IntegerField(null=True)),
            ('doc_id', models.IntegerField(null=True)),
            ('document', models.FileField(upload_to='files/')),
            ('doc_name', models.CharField(max_length=200)),
            ('description', models.CharField(max_length=250, null=True)),
            ('doc_size', models.BigIntegerField(null=True)),
            ('doc_type', models.CharField(max_length=300)),
            ('file_enc_key', models.TextField(null=True)),

```

```

        ('secure_key', models.CharField(max_length=100, null=True)),
        ('request_status', models.CharField(default='pending', max_length=50, null=True)),
        ('download_status', models.CharField(default='pending', max_length=200, null=True)),
        ('requested_date', models.DateField(auto_now_add=True, null=True)),
    ],
    options={
        'db_table': 'user_download_requests',
    },
),
migrations.CreateModel(
    name='DownloadsModel',
    fields=[
        ('download_id', models.AutoField(primary_key=True, serialize=False)),
        ('user_id', models.IntegerField(null=True)),
        ('doc_id', models.IntegerField(null=True)),
        ('status', models.CharField(default='pending', max_length=200, null=True)),
        ('download_rank', models.IntegerField(null=True)),
        ('downloaded_date', models.DateField(auto_now_add=True, null=True)),
    ],
    options={
        'db_table': 'document_downloads',
    },
),
migrations.CreateModel(
    name='UserModel',
    fields=[
        ('user_id', models.AutoField(primary_key=True, serialize=False)),
        ('name', models.TextField(default=True, max_length=200)),
        ('mobile', models.BigIntegerField(default=True)),
        ('email', models.EmailField(default=True, max_length=254)),
        ('profile', models.ImageField(null=True, upload_to='user/')),
        ('pwd', models.CharField(max_length=20, null=True)),
        ('cpwd', models.CharField(max_length=20, null=True)),
        ('reg_date', models.DateField(auto_now_add=True)),
    ],

```

```

        ('otp', models.TextField(null=True)),
        ('status', models.CharField(default='pending', max_length=20)),
    ],
    options={
        'db_table': 'user_details',
    },
),
]

```

Data Owner app:

Views.py:

```

import imp
from operator import indexOf
import os
from tabnanny import filename_only
from xml.dom.minidom import Document
from django.conf import settings
from django.shortcuts import render, redirect, get_object_or_404
from ownerapp.models import *
from userapp.models import *
from django.contrib import messages
import random
import requests
import string
import random
from userapp import practice as p
from cryptography.fernet import Fernet
import pathlib
from django.db.models import Q
from FileSecure.settings import DEFAULT_FROM_EMAIL
from django.core.mail import EmailMultiAlternatives
from FileSecure import check_internet
from FileSecure.check_internet import *
from django.core.files.storage import FileSystemStorage
# Create your views here.

```

```

def owner_register(request):
    if request.method == 'POST' and request.FILES["profile"]:
        print("post method")
        name = request.POST.get('name')
        mobile = request.POST.get('mobile')
        email = request.POST.get('email')
        profile = request.FILES['profile']
        pwd = request.POST.get('pwd')
        cpwd = request.POST.get('cpwd')
        if OwnerModel.objects.filter(email=email).filter(verification="verified").exists():
            print("email already exists")
            messages.error(request, "already taken")
            return redirect('owner-register')
        elif OwnerModel.objects.filter(email=email).filter(verification = "pending").exists():
            # messages.error(request, "Already Registered, Just verify your account!")
            if pwd == cpwd:
                otp=random.randint(2222,4444)
                OwnerModel.objects.filter(email=email,status='pending').update(otp=otp)
                if connect():
                    html_content = "<br/> WellCome To Data Owner, <br/> <p> This Message Sent From
FileSecure You Have Recieved a OTP <strong> " +str(otp)+ " </strong> on FileSecure Cloud Based File
Services. </strong> <b> You Can Use online services 24/7 <strong> Thank You For Your
Registration.</p>"
                    from_mail = DEFAULT_FROM_EMAIL
                    to_mail = [email]
                    # if send_mail(subject,message,from_mail,to_mail):
                    msg = EmailMultiAlternatives("Account Registration Status", html_content, from_mail,
to_mail)
                    msg.attach_alternative(html_content, "text/html")
                    print(email,"email ")
                    return redirect('owner-otp-verify')
            else:
                messages.warning(request, "password doesn't match")
                return redirect('owner-register')

```

```

else:
    otp = random.randint(1111,9999)
    print(otp)

    if pwd == cpwd:
        owner_register =
OwnerModel.objects.create(name=name,email=email,mobile=mobile,pwd=pwd,cpwd=cpwd,otp=otp,pro
file=profile)
        owner = OwnerModel.objects.get(email=email)
        otp_data = owner.owner_id
        print('session',otp_data)
        request.session['demo'] = otp_data
        print(otp_data)

    if connect():
        html_content = "<br/> WellCome To Data Owner, <br/> <p> This Message Sent From
FileSecure You Have Recieved a OTP <strong> " +str(otp)+ " </strong> on FileSecure Cloud Based File
Services. </strong> <b> You Can Use online services 24/7 <strong> Thank You For Your
Registration.</p>"
        from_mail = DEFAULT_FROM_EMAIL
        to_mail = [owner_register.email]
        # if send_mail(subject,message,from_mail,to_mail):
        msg = EmailMultiAlternatives("Account Registered Status", html_content, from_mail,
to_mail)
        msg.attach_alternative(html_content, "text/html")
        print(email)
        if msg.send():
            print("Sent")
            print('your success')
        return redirect('owner-otp-verify')
    else:
        messages.warning(request,"password doesn't match")
        return redirect('owner-register')
# messages.info(request,"successfully registered")

```

```

return render(request,'owner/owner-register.html')

def owner_otp_verify(request):
    owner_id = request.session['demo']
    print("owner_id",owner_id)
    data = OwnerModel.objects.get(owner_id=owner_id)
    if request.method == 'POST':
        otp = request.POST.get('otp')
        print(otp)
        try:
            check = OwnerModel.objects.get(otp=otp)
            print(check)
            o_id = request.session["owner_id"]=check.owner_id
            otp = check.otp
            if otp == otp:
                print("c")
                OwnerModel.objects.filter(owner_id=o_id).update(verification="verified")
                print("record")
                messages.info(request,"otp-verify")
                return redirect('owner-login')
            else:
                print("d")
                messages.error(request,"wrong-otp")
                return redirect('owner-otp-verify')
        except:
            pass
    return render(request,'owner/owner-otp-verify.html',{'d':data})

def owner_login(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        pwd = request.POST.get('pwd')
        try:
            check = OwnerModel.objects.get(email=email,pwd=pwd)
            print("check")
            id = request.session["owner_id"]=check.owner_id

```



```

status = check.status

if status == "Accepted" and check.verification == "verified" :
    messages.info(request,"log in successfull!")
    print("message....")
    return redirect('owner-home')
elif status == "Rejected" or status=="pending":
    messages.warning(request,"admin not accepted")
else:
    messages.error(request,"something went wrong!")
    return redirect('owner-login')
except:
    print('invalid.....')
    messages.error(request,"Invalid Credentials")
    return redirect("owner-login")

return render(request,'owner/owner-login.html')

def owner_home(request):
    owner_id = request.session["owner_id"]
    document = UploadDocumentsModel.objects.filter(owner_id=owner_id,status="Accepted")
    if request.method == "POST" and 'btn1' in request.POST or 'btn2' in request.POST:
        search = request.POST.get("search")
        document =
UploadDocumentsModel.objects.filter(owner_id=owner_id,status="Accepted").filter(Q(doc_name__icontains=search)|Q(doc_type__icontains=search)|Q(random_keys__icontains=search))
    return render(request,'owner/owner-home.html',{'docs':document})

def owner_upload_docs(request):
    owner_id = request.session['owner_id']
    supported_files = "html,java,py,txt,css,js"
    # print("support",supported_files)
    if request.method == 'POST' and request.FILES['file']:
        print("post method")
        # key = str(Fernet.generate_key(), 'utf-8')
        # crypter = Fernet(key)
        # document = request.FILES['file'].encode()
        # latt = str(crypter.encrypt(document), 'utf-8')

```

```

document=request.FILES['file']
description = request.POST.get('description')
doc_name = document.name
doc_type = document.content_type
file_extension = doc_name
fs = FileSystemStorage()
name = fs.save('files/' + doc_name,document)
print("111111")
print(name)
a = file_extension.index(".")
b = len(file_extension)
x = file_extension[a+1:b]
print(x)
if x in supported_files:
    print("File is supportedddd")
    doc_size = document.size
    owner = OwnerModel.objects.get(owner_id = owner_id)
    files =
UploadDocumentsModel.objects.create(doc_name=doc_name,doc_size=doc_size,doc_type=doc_type,des
cription=description,document=name,owner=owner)
    files.save()
    messages.info(request,"document uploaded")
    doc_id = request.session["doc_id"] = files.doc_id
    print(doc_id)
    return redirect('owner-encrypt-file')
else:
    messages.error(request,"The " + x + " file format is not supported by the cloud")
    return redirect("upload-docs")
return render(request,'owner/owner-upload-docs.html')
def owner_encrypt_file(request):
    owner_id = request.session['owner_id']
    data = "
    document = request.session["doc_id"]
    file = UploadDocumentsModel.objects.get(doc_id=document)

```

```

filename = str(file.doc_name)
# f = filename.replace(' ','_')
path = 'media/files/' + filename
if request.method=="GET":
    print(path)
    try:
        f = open(path,'r')
        print('f')
        data = f.read()
        f.close()
        print(data)
    except Exception as e:
        print(e)
elif request.method=="POST" and 'encrypt' in request.POST:
    print("keywords")
    data1 = request.POST.get("description")
    keywords = request.POST.get('keywords')
    print(keywords)
    # #File Encryption

    # file_enc_key = Fernet.generate_key()
    # print(file_enc_key,"keykeykey")
    # print("aaaa")
    # print(file_enc_key,"dddd")
    # print("aaaa")
    # data_encode = data1.encode()
    # fernet = Fernet(file_enc_key)
    # enc_data_1 = fernet.encrypt(data_encode).decode()
    # print(enc_data_1)
    # #File Encryption
    file_enc_key = Fernet.generate_key()
    #Read File
    print(path)

```

```

file = open(path,'rb')
data = file.read()
print(data)
fernet = Fernet(file_enc_key)
data = fernet.encrypt(data)
print(data)
UploadDocumentsModel.objects.filter(owner_id=owner_id,doc_name=filename).update(file_enc_key=file_enc_key.decode(),file_data=data.decode(),random_keys=keywords)
    with open(path,'wb') as f:
        f.write(data)
    messages.warning(request,"encrypted file uploaded")
    return redirect("owner-home")

    return render(request,"owner/owner-encrypt-file.html",{ 'file': data,'filename': filename})
def owner_download_requests(request):
    owner_id = request.session["owner_id"]
    # docs = UploadDocumentsModel.objects.filter(status="Accepted")
    user_data = UserModel.objects.all()
    document = DownloadRequestModel.objects.filter(owner_id=owner_id).all()
    if request.method == "POST" and 'btn1' in request.POST:
        search = request.POST.get("search")
        document =
DownloadRequestModel.objects.filter(Q(doc_name__icontains=search)|Q(doc_type__icontains=search))
    return render(request,'owner/owner-download-requests.html',{'req':document,'user':user_data})

# key or captcha generator, we can mention any size in size argument
def generate_key(size=10, chars=string.ascii_uppercase + string.ascii_lowercase + string.digits +
'!@#$$%^&*()_+?~'):
    return ''.join(random.choice(chars) for _ in range(size))

def key_generator(request,id):
    key = generate_key()
    print(key)
    accept = get_object_or_404(DownloadRequestModel,request_id =id)
    accept.request_status = "request_accepted"

```

```

    accept.secure_key=key
    accept.save(update_fields=['request_status','secure_key'])
    accept.save()
    #
DownloadRequestModel.objects.filter(request_id=id).update(secure_key=key,request_status="accepted")
    return redirect('owner-download-requests')
def reject_request(request,id):
    reject = get_object_or_404(DownloadRequestModel,request_id =id)
    reject.request_status = "Rejected"
    reject.save(update_fields=['request_status'])
    reject.save()
    return redirect('owner-download-requests')
def owner_topk_downloads(request):
    owner_id = request.session["owner_id"]
    # owner = UploadDocumentsModel.objects.get(owner_id = owner_id)
    # doc_id = owner.doc_id
    Document = UploadDocumentsModel.objects.filter(owner_id=owner_id).order_by("-download_rank")

    # downloads = DownloadsModel.objects.all()
    # request_id = downloads.req_id
    # downloads =
DownloadsModel.objects.filter(Q(doc_name__icontains=search)|Q(doc_type__icontains=search))
    # user = UserModel.objects.all()
    # ddd = DownloadRequestModel.objects.filter(owner_id = owner_id)
    return render(request,'owner/owner-topk-downloads.html',{'downloads':Document,})

def owner_profile(request):
    owner = request.session["owner_id"]
    data = OwnerModel.objects.get(owner_id=owner)
    # owner_data =
OwnerModel.objects.filter(owner_id=owner).exclude(status="paid").exclude(status='Delivered').count()

    obj = get_object_or_404(OwnerModel,owner_id=owner)
    if request.method == "POST" and request.FILES["profile"]:

```

```

name = request.POST.get('username')
mobile = request.POST.get('phone')
email = request.POST.get('email')
# address = request.POST.get('address')
profile = request.FILES['profile']
# pincode = request.POST.get('pincode')
obj.name = name
obj.mobile = mobile
obj.profile = profile
obj.email = email
obj.save(update_fields=['name','profile','mobile','email'])
obj.save()
messages.info(request,"profile Updated successfully..")
if obj.save():
    messages.info(request,"profile Updated successfully..")
else:
    messages.error(request,"something went wrong!")
return redirect("owner-profile")
return render(request,'owner/owner-profile.html',{'owner':data})
def owner_about(request):
    return render(request,'owner/owner-about.html')
Models.py:
from pyexpat import model
from django.db import models
# Create your models here.
class OwnerModel(models.Model):
    owner_id = models.AutoField(primary_key=True)
    name = models.TextField(max_length=200,default=True)
    mobile = models.BigIntegerField(default=True)
    email = models.EmailField(default=True)
    profile = models.ImageField(upload_to='owner/',null=True)
    pwd = models.CharField(max_length=20,null=True)
    cpwd = models.CharField(max_length=20,null=True)
    reg_date = models.DateField(auto_now_add=True)

```

```

otp = models.TextField(null=True)
status = models.CharField(default="pending",max_length=20)
verification = models.CharField(default="pending",max_length=20)
class Meta:
    db_table = 'owner_details'
class UploadDocumentsModel(models.Model):
    doc_id = models.AutoField(primary_key=True)
    owner = models.ForeignKey(OwnerModel,on_delete=models.CASCADE,null=True)
    document = models.ImageField(upload_to='files/',null=True)
    doc_type = models.TextField(null=True)
    doc_size = models.BigIntegerField(null=True)
    description = models.TextField(null=True)
    doc_name = models.TextField(null=True)
    enc_doc_name = models.TextField(null=True)
    random_keys = models.TextField(null=True)
    file_enc_key = models.TextField(null=True)
    file_data = models.TextField(null=True)
    download_rank = models.IntegerField(null=True)
    upload_date = models.DateField(auto_now_add=True)
    status = models.TextField(default="pending")
class Meta:
    db_table='upload_documents'

```

Apps.py :

```

from django.apps import AppConfig
class OwnerappConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'ownerapp'

```

Migration.py

Generated by Django 4.0.4 on 2022-06-25 10:45

```

from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('ownerapp', '0001_initial'),
    ]

```

```

operations = [
    migrations.AddField(
        model_name='uploaddocumentsmodel',
        name='download_rank',
        field=models.IntegerField(null=True),
    ),
]

```

Initial.py:

```
from django.db import migrations, models
```

```
import django.db.models.deletion
```

```
class Migration(migrations.Migration):
```

```
    initial = True
```

```
dependencies = [
]
```

```

operations = [
    migrations.CreateModel(
        name='OwnerModel',
        fields=[
            ('owner_id', models.AutoField(primary_key=True, serialize=False)),
            ('name', models.TextField(default=True, max_length=200)),
            ('mobile', models.BigIntegerField(default=True)),
            ('email', models.EmailField(default=True, max_length=254)),
            ('profile', models.ImageField(null=True, upload_to='owner/')),
            ('pwd', models.CharField(max_length=20, null=True)),
            ('cpwd', models.CharField(max_length=20, null=True)),
            ('reg_date', models.DateField(auto_now_add=True)),
            ('otp', models.TextField(null=True)),
            ('status', models.CharField(default='pending', max_length=20)),
            ('verification', models.CharField(default='pending', max_length=20)),
        ],
        options={
            'db_table': 'owner_details',
        },
    ),
]

```



```

    ),
    migrations.CreateModel(
        name='UploadDocumentsModel',
        fields=[
            ('doc_id', models.AutoField(primary_key=True, serialize=False)),
            ('document', models.ImageField(null=True, upload_to='files/')),
            ('doc_type', models.TextField(null=True)),
            ('doc_size', models.BigIntegerField(null=True)),
            ('description', models.TextField(null=True)),
            ('doc_name', models.TextField(null=True)),
            ('enc_doc_name', models.TextField(null=True)),
            ('random_keys', models.TextField(null=True)),
            ('file_enc_key', models.TextField(null=True)),
            ('file_data', models.TextField(null=True)),
            ('upload_date', models.DateField(auto_now_add=True)),
            ('status', models.TextField(default='pending')),
            ('owner', models.ForeignKey(null=True, on_delete=django.db.models.deletion.CASCADE,
to='ownerapp.ownermodel')),
        ],
        options={
            'db_table': 'upload_documents',
        },
    ),
]

```

Cloud server app:

Views.py:

```

from pyexpat.errors import messages
from django.shortcuts import render,redirect,get_object_or_404
from ownerapp.models import OwnerModel,UploadDocumentsModel
from userapp.models import DownloadsModel, UserModel,DownloadRequestModel
import datetime
from django.contrib import messages
# Create your views here.

```

```

def cloud_login(request):
    if request.method == "POST":
        name= request.POST.get("name")
        pwd = request.POST.get("pwd")
        if name == 'cloud' and pwd == 'cloud':
            messages.info(request, "login success")
            return redirect('cloud-index')
        else:
            messages.warning(request, "wrong details")
    return render(request, 'cloud/cloud-login.html')

def cloud_index(request):
    # current time and date
    # datetime object
    date = datetime.date.today()
    print(date)
    # formating date using strftime
    # print("After formating:", time.strftime("%b %d, %Y"))
    users_count = UserModel.objects.all().count()
    owners_count = OwnerModel.objects.filter(status="accepted").count()
    doc_count = UploadDocumentsModel.objects.filter(status="Accepted").count()
    doc_req_count = UploadDocumentsModel.objects.filter(status="pending").count()
    total_downloads = DownloadsModel.objects.all().count()
    m_downloads = DownloadsModel.objects.filter(downloaded_date=date).count()
    down_req_count = DownloadRequestModel.objects.filter(download_status='pending').count()
    return render(request, 'cloud/index.html', {
        'u_count':users_count,
        'o_count':owners_count,
        'd_count':doc_count,
        'd_r_count':down_req_count,
        'doc_req_count':doc_req_count,
        'tot_doc_count':total_downloads,
        'month_downloads':m_downloads,
    })

def data_owner(request):

```

```

owner = OwnerModel.objects.all().order_by('-reg_date')
    return render(request,'cloud/cloud-data-owner.html',{'owner':owner})
def accept_status(request,id):
    accept = get_object_or_404(OwnerModel,owner_id =id)
    accept.status = "Accepted"
    accept.save(update_fields=['status'])
    accept.save()
    return redirect("data-owner")
def reject_status(request,id):
    accept = get_object_or_404(OwnerModel,owner_id =id)
    accept.status = "Rejected"
    accept.save(update_fields=['status'])
    accept.save()
    return redirect("data-owner")
def data_user(request):
    user = UserModel.objects.all()
    return render(request,'cloud/cloud-data-user.html',{'user':user})
def view_documents(request):
    docs = UploadDocumentsModel.objects.all().order_by('-upload_date')
    return render(request,'cloud/cloud-view-documents.html',{'docs':docs})
def accept_document(request,id):
    accept = get_object_or_404(UploadDocumentsModel,doc_id =id)
    accept.status = "Accepted"
    accept.save(update_fields=['status'])
    accept.save()
    return redirect("view-documents")
def reject_document(request,id):
    accept = get_object_or_404(UploadDocumentsModel,doc_id =id)
    accept.status = "Rejected"
    accept.save(update_fields=['status'])
    accept.save()
    return redirect("view-documents")
def view_downloads(request):
    downloads = DownloadsModel.objects.all()

```

```

user = UserModel.objects.all()
owner = UploadDocumentsModel.objects.all()
return render(request,'cloud/cloud-view-downloads.html',{'d':downloads,'owner':owner,'u':user})

```

Apps.py:

```

from django.apps import AppConfig
class CloudappConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'cloudapp'

```

Emailsettings.py :

```

SET_EMAIL_USE_TLS = True
SET_EMAIL_HOST='smtp.gmail.com'
SET_EMAIL_HOST_USER='projects@codebook.in'
SET_EMAIL_HOST_PASSWORD='frwqvhawrnsxetyk'
SET_EMAIL_PORT=587
SET_EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
SET_DEFAULT_FROM_EMAIL = 'projects@codebook.in'

```

Settings.py:

```

from pathlib import Path
import os
from .emailsettings import SET_EMAIL_USE_TLS, SET_EMAIL_HOST, SET_EMAIL_HOST_USER, \
    SET_EMAIL_HOST_PASSWORD, SET_EMAIL_PORT, SET_EMAIL_BACKEND, \
    SET_DEFAULT_FROM_EMAIL
from django.contrib.messages import constants as messages
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-xhkn_dmpywn@5iq9r+i=2bp%3=976713lgvo46yvku0r9e35-w'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition

```

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'cloudapp',
    'mainapp',
    'ownerapp',
    'userapp',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'FileSecure.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'assets/templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]

```

```

    },
    },
]
WSGI_APPLICATION = 'FileSecure.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'filesecure',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        }
    }
}

# Password validation

# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```

]
# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/
STATIC_URL = '/static/'
STATICFILES_DIRS=[os.path.join(BASE_DIR,'assets/static'),]
MEDIA_URL = '/media/'
MEDIA_ROOT= os.path.join(BASE_DIR,'media/')
# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
EMAIL_USE_TLS = SET_EMAIL_USE_TLS
EMAIL_HOST = SET_EMAIL_HOST
EMAIL_HOST_USER = SET_EMAIL_HOST_USER
EMAIL_HOST_PASSWORD = SET_EMAIL_HOST_PASSWORD
EMAIL_PORT = SET_EMAIL_PORT
EMAIL_BACKEND = SET_EMAIL_BACKEND
DEFAULT_FROM_EMAIL = SET_DEFAULT_FROM_EMAIL
MESSAGE_TAGS = {
    messages.DEBUG : 'alert-secondary',
    messages.INFO: 'alert-info',
    messages.WARNING: 'alert-warning',
    messages.SUCCESS: 'alert-success',
    messages.ERROR: 'alert-danger',
}
Check_internet.py:
import urllib.request
def connect(host='http://google.com'):
    try:

```

```
urllib.request.urlopen(host)
```

```
    return True
```

```
except:
```

```
    return False
```

Urls.py:

```
from django.contrib import admin
```

```
from django.urls import path
```

```
from mainapp import views as mainviews
```

```
from userapp import views as userviews
```

```
from ownerapp import views as ownerviews
```

```
from cloudapp import views as cloudviews
```

```
from django.conf.urls.static import static
```

```
from django.conf import settings
```

```
urlpatterns = [
```

```
    path('admin/', admin.site.urls),
```

```
    # main page url's
```

```
    path("",mainviews.main_index,name="index"),
```

```
    path('about',mainviews.about,name="about"),
```

```
    # user page url's
```

```
    path('user-register',userviews.user_register,name="user-register"),
```

```
    path('user-otp-verify',userviews.otp_verify,name="user-otp-verify"),
```

```
    path('user-login',userviews.user_login,name="user-login"),
```

```
    path('user-home',userviews.user_home,name="user-home"),
```

```
    path('download-requests/<int:id>/',userviews.download_requests,name="download-requests"),
```

```
    path('user-view-docs',userviews.user_search_docs,name="user-view-docs"),
```

```
    path('user-mydownloads',userviews.user_mydownloads,name="user-mydownloads"),
```

```
    path('user-profile',userviews.user_profile,name="user-profile"),
```

```
    path('user-response-docs',userviews.user_response_docs,name="user-response-docs"),
```

```
    path('user-download-document/<int:id>/',userviews.download_doc,name="user-download-document"),
```

```
    path('user-decrypt-document/<int:id>/',userviews.decrypt_document,name="user-decrypt-document"),
```

```
    path('user-aboutus',userviews.user_about,name="user-aboutus"),
```

```
    # owner page url's
```

```
    path('owner-register',ownerviews.owner_register,name="owner-register"),
```

```
    path('owner-otp-verify',ownerviews.owner_otp_verify,name="owner-otp-verify"),
```



```

path('owner-encrypt-file',ownerviews.owner_encrypt_file,name="owner-encrypt-file"),
    path('owner-login',ownerviews.owner_login,name="owner-login"),
    path('owner-home',ownerviews.owner_home,name="owner-home"),
    path('upload-docs',ownerviews.owner_upload_docs,name="upload-docs"),
    path('owner-download-requests',ownerviews.owner_download_requests,name="owner-download-
requests"),
    path('key-generator/<int:id>/',ownerviews.key_generator,name="key-generator"),
    path('reject-request/<int:id>/',ownerviews.reject_request,name="reject-request"),
    path('top-kdownloads',ownerviews.owner_topk_downloads,name="top-kdownloads"),
    path('owner-profile',ownerviews.owner_profile,name="owner-profile"),
    path('owner-aboutus',ownerviews.owner_about,name="owner-aboutus"),
    #cloud page url's
    path('clod-login',cloudviews.cloud_login,name="cloud-login"),
    path('cloud-index',cloudviews.cloud_index,name="cloud-index"),
    path('data-owner',cloudviews.data_owner,name="data-owner"),
    path('data-user',cloudviews.data_user,name="data-user"),
    path('view-documents',cloudviews.view_documents,name="view-documents"),
    path('accept-document/<int:id>/',cloudviews.accept_document,name="accept-document"),
    path('reject-document/<int:id>/',cloudviews.reject_document,name="reject-document"),
    path('view-downloads',cloudviews.view_downloads,name="view-downloads"),
    path('accept-owner/<int:id>/',cloudviews.accept_status,name="accept-owner"),
    path('reject-owner/<int:id>/',cloudviews.reject_status,name="reject-owner"),

]
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

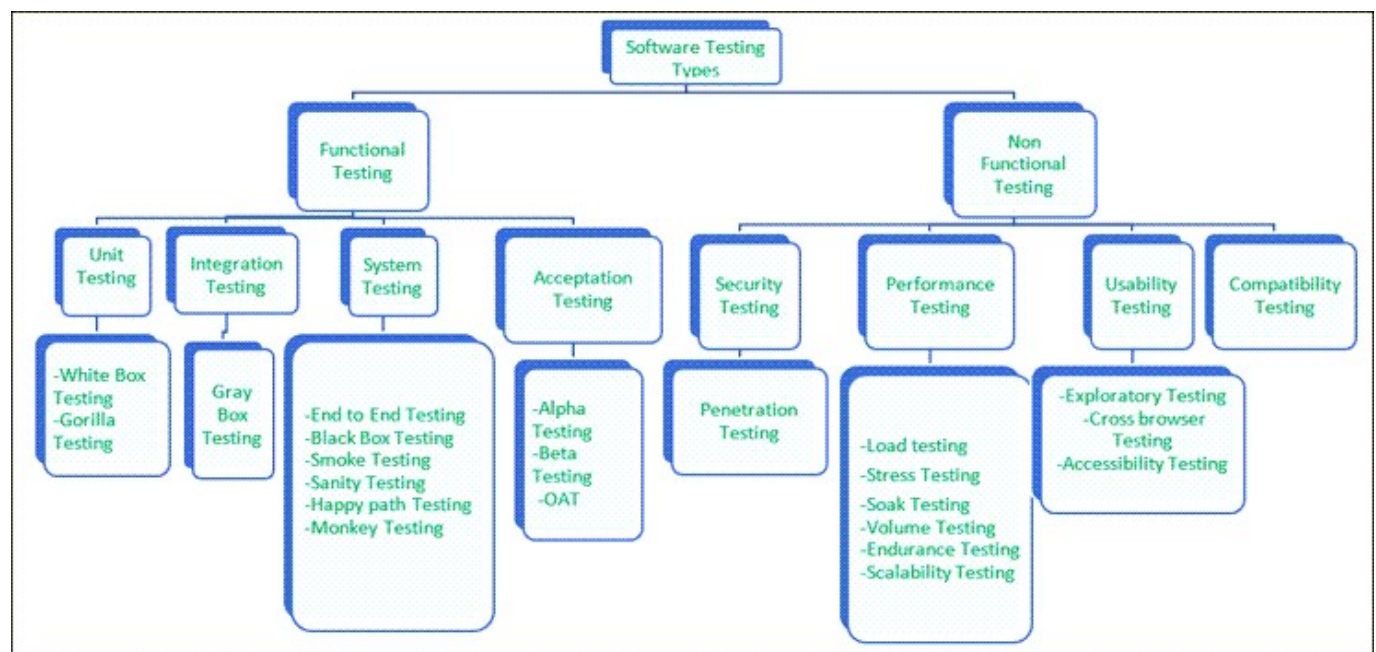
```

7. SYSTEM TESTING

Types of Software Testing: Different Testing Types with Details

We, as testers, are aware of the various types of Software Testing like Functional Testing, Non-Functional Testing, Automation testing, Agile Testing, and their sub-types, etc. Each type of testing has its own features, advantages, and disadvantages as well. However, in this tutorial, we have covered mostly each and every type of software testing which we usually use in our day-to-day testing life.

Different Types of Software Testing



Functional Testing

There are four main types of functional testing.

7.1 Unit Testing

Unit testing is a type of software testing which is done on an individual unit or component to test its corrections. Typically, Unit testing is done by the developer at the application development phase. Each unit in unit testing can be viewed as a method, function, procedure, or object. Developers often use test automation tools such as NUnit, Xunit, JUnit for the test execution. Unit testing is important because we can find more defects at the unit test level. **For example**, there is a simple calculator application. The developer can write the unit test to check if the user can enter two numbers and get the correct sum for addition functionality.

a) White Box Testing

White box testing is a test technique in which the internal structure or code of an application is visible and accessible to the tester. In this technique, it is easy to find loopholes in the design of an application or fault in business logic. Statement coverage and decision coverage/branch coverage are examples of white box test techniques.

b) Gorilla Testing

Gorilla testing is a test technique in which the tester and/or developer test the module of the application thoroughly in all aspects. Gorilla testing is done to check how robust your application is. **For example**, the tester is testing the pet insurance company's website, which provides the service of buying an insurance policy, tag for the pet, Lifetime membership. The tester can focus on any one module, let's say, the insurance policy module, and test it thoroughly with positive and negative test scenarios.

7.2 Integration Testing

Integration testing is a type of software testing where two or more modules of an application are logically grouped together and tested as a whole. The focus of this type of testing is to find the defect on interface, communication, and data flow among modules. Top-down or Bottom-up approach is used while integrating modules into the whole system.

This type of testing is done on integrating modules of a system or between systems. **For example**, a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems. Integration testing should be done while integrating of airline website and payment processing system.

a) Gray box testing

As the name suggests, gray box testing is a combination of white-box testing and black-box testing. Testers have partial knowledge of the internal structure or code of an application.

7.3 System Testing

System testing is a type of testing where tester evaluates the whole system against the specified requirements.

a) End to End Testing

It involves testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

For example, a tester is testing a pet insurance website. End to End testing involves testing of buying an insurance policy, LPM, tag, adding another pet, updating credit card information on users' accounts, updating user address information, receiving order confirmation emails and policy documents.

b) Black Box Testing

Blackbox testing is a software testing technique in which testing is performed without knowing the internal structure, design, or code of a system under test. Testers should focus only on the input and output of test objects.

c) Smoke Testing

Smoke testing is performed to verify that basic and critical functionality of the system under test is working fine at a very high level.

Whenever a new build is provided by the development team, then the Software Testing team validates the build and ensures that no major issue exists. The testing team will ensure that the build is stable, and a detailed level of testing will be carried out further.

For example, tester is testing pet insurance website. Buying an insurance policy, adding another pet, providing quotes are all basic and critical functionality of the application. Smoke testing for this website verifies that all these functionalities are working fine before doing any in-depth testing.

d) Sanity Testing

Sanity testing is performed on a system to verify that newly added functionality or bug fixes are working fine. Sanity testing is done on stable build. It is a subset of the regression test. **For example**, a tester is testing a pet insurance website. There is a change in the discount for buying a policy for second pet. Then sanity testing is only performed on buying insurance policy module.

e) Happy path Testing

The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on valid and positive inputs through which the application generates the expected output.

f) Monkey Testing

Monkey Testing is carried out by a tester, assuming that if the monkey uses the application, then random input and values will be entered by the Monkey without any knowledge or understanding of the application. The objective of Monkey Testing is to check if an application or system gets crashed by providing random input values/data. Monkey Testing is performed randomly, no test cases are scripted, and it is not necessary to be aware of the full functionality of the system.

7.4 Acceptance Testing

Acceptance testing is a type of testing where client/business/customer test the software with real time business scenarios.

The client accepts the software only when all the features and functionalities work as expected. This is the last phase of testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

a) Alpha Testing

Alpha testing is a type of acceptance testing performed by the team in an organization to find as many defects as possible before releasing software to customers.

For example, the pet insurance website is under UAT. The UAT team will run real-time scenarios like buying an insurance policy, buying annual membership, changing the address, ownership transfer of the pet in a same way the user uses the real website. The team can use test credit card information to process payment-related scenarios.

b) Beta Testing

Beta Testing is a type of software testing which is carried out by the clients/customers. It is performed in the **Real Environment** before releasing the product to the market for the actual end-users.

Beta Testing is carried out to ensure that there are no major failures in the software or product, and it satisfies the business requirements from an end-user perspective. Beta Testing is successful when the customer accepts the software. Usually, this testing is typically done by the end-users. This is the final testing done before releasing the application for commercial purposes. Usually, the Beta version of the

software or product released is limited to a certain number of users in a specific area. So, the end-user uses the software and shares the feedback with the company. The company then takes necessary action before releasing the software worldwide.

c) Operational acceptance testing (OAT)

Operational acceptance testing of the system is performed by operations or system administration staff in the production environment. The purpose of operational acceptance testing is to make sure that the system administrators can keep the system working properly for the users in a real-time environment.

The focus of the OAT is on the following points:

- Testing of backup and restore.
- Installing, uninstalling, upgrading software.
- The recovery process in case of natural disaster.

- User management.
- Maintenance of the software.

7.5 Security Testing

It is a type of testing performed by a special team. Any hacking method can penetrate the system. Security Testing is done to check how the software, application, or website is secure from internal and/or external threats. This testing includes how much software is secure from malicious programs, viruses and how secure & strong the authorization and authentication processes are. It also checks how software behaves for any hacker's attack & malicious programs and how software is maintained for data security after such a hacker attack.

a) Penetration Testing

Penetration Testing or Pen testing is the type of security testing performed as an authorized cyberattack on the system to find out the weak points of the system in terms of security.

Pen testing is performed by outside contractors, generally known as ethical hackers. That is why it is also known as ethical hacking. Contractors perform different operations like SQL injection, URL manipulation, Privilege Elevation, session expiry, and provide reports to the organization. **Notes:** Do not perform the Pen testing on your laptop/computer. Always get written permission to do pen tests.

7.6 Performance Testing

Performance testing is testing of an application's stability and response time by applying load.

The word stability means the ability of the application to withstand in the presence of load.

Response time is how quickly an application is available to users. Performance testing is done with the help of tools. Loader.IO, JMeter, LoadRunner, etc. are good tools available in the market.

a) Load testing

Load testing is testing of an application's stability and response time by applying load, which is equal to or less than the designed number of users for an application.

For example, your application handles 100 users at a time with a response time of 3 seconds, then load testing can be done by applying a load of the maximum of 100 or less than 100 users. The goal is to verify that the application is responding within 3 seconds for all the users.

b) Stress Testing

Stress testing is testing an application's stability and response time by applying load, which is more than the designed number of users for an application.

For example, your application handles 1000 users at a time with a response time of 4 seconds, then stress testing can be done by applying a load of more than 1000 users. Test the application with 1100,1200,1300 users and notice the response time. The goal is to verify the stability of an application under stress.

c) Scalability Testing

Scalability testing is testing an application's stability and response time by applying load, which is more than the designed number of users for an application.

For example, your application handles 1000 users at a time with a response time of 2 seconds, then scalability testing can be done by applying a load of more than 1000 users and gradually increasing the number of users to find out where exactly my application is crashing.

Let's say my application is giving response time as follows:

- 1000 users -2 sec
- 1400 users -2 sec
- 4000 users -3 sec
- 5000 users -45 sec
- 5150 users- crash – This is the point that needs to identify in scalability testing

d) Volume testing (flood testing)

Volume testing is testing an application's stability and response time by transferring a large volume of data to the database. Basically, it tests the capacity of the database to handle the data.

e) Endurance Testing (Soak Testing)

Endurance testing is testing an application's stability and response time by applying load continuously for a longer period to verify that the application is working fine. **For example**, car companies soak testing to verify that users can drive cars continuously for hours without any problem.

7.7 Usability Testing

Usability testing is testing an application from the user's perspective to check the look and feel and user-friendliness. **For example**, there is a mobile app for stock trading, and a tester is performing usability testing. Testers can check the scenario like if the mobile app is easy to operate with one hand or not, scroll bar should be vertical, background colour of the app should be black and price of and stock is displayed in red or green colour.

The main idea of usability testing of this kind of app is that as soon as the user opens the app, the user should get a glance at the market.

a) Exploratory testing

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Testers use the knowledge of the business domain to test the application. Test charters are used to guide the exploratory testing.

b) Cross browser testing

Cross browser testing is testing an application on different browsers, operating systems, mobile devices to see look and feel and performance. Why do we need cross-browser testing? The answer is different users use different operating systems, different browsers, and different mobile devices. The goal of the company is to get a good user experience regardless of those devices. Browser stack provides all the versions of all the browsers and all mobile devices to test the application. For learning purposes, it is good to take the free trial given by browser stack for a few days.

c) Accessibility Testing

The aim of Accessibility Testing is to determine whether the software or application is accessible for disabled people or not. Here, disability means deafness, colour blindness, mentally disabled, blind, old age, and other disabled groups. Various checks are performed, such as font size for visually disabled, colour and contrast for colour blindness, etc.

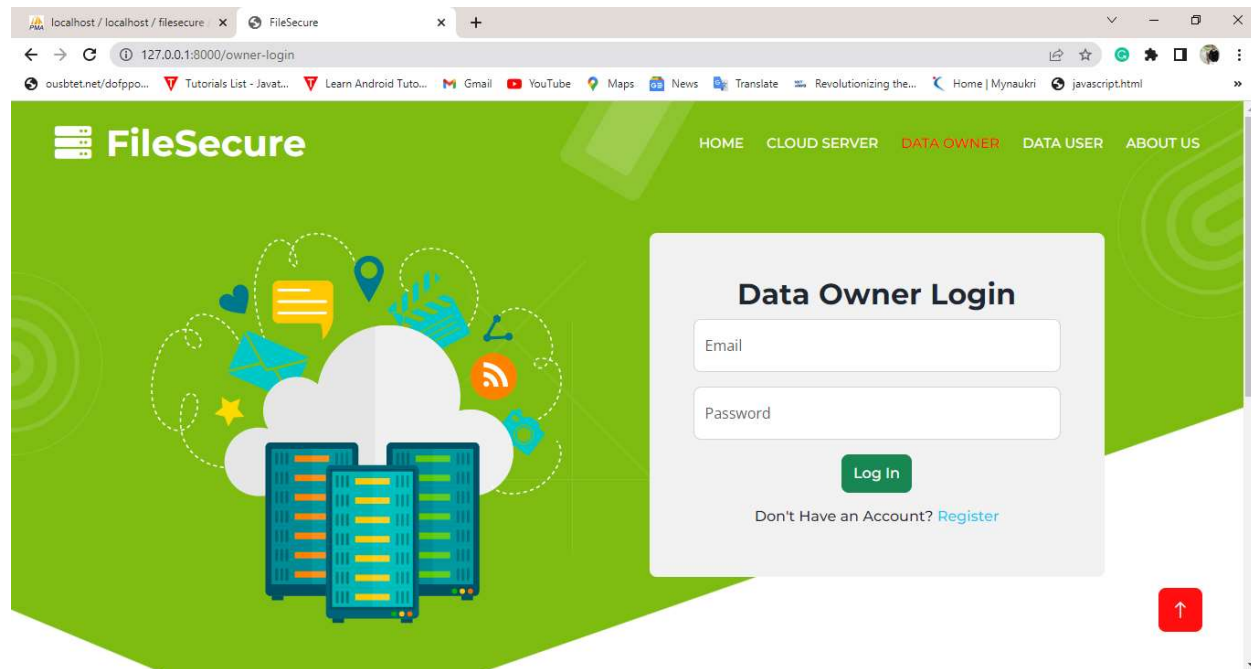
Compatibility testing

This is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment.

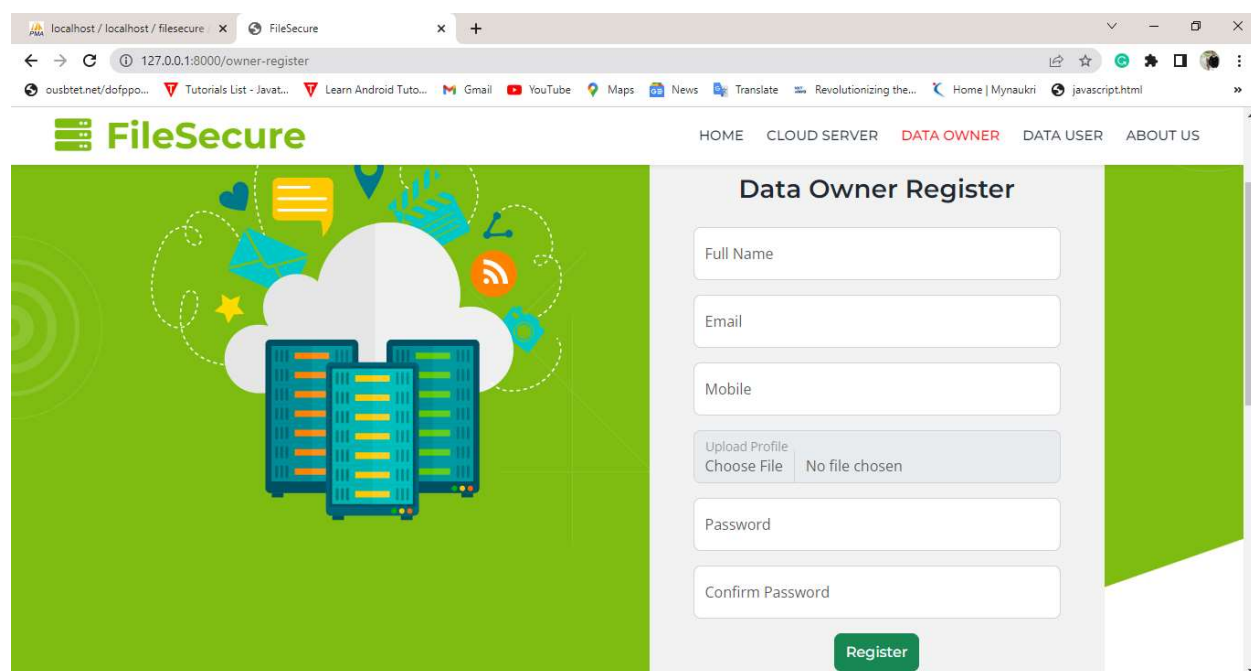
8. SCREENSHOTS

This section will familiarize you with the overall interface of the website including all the process.

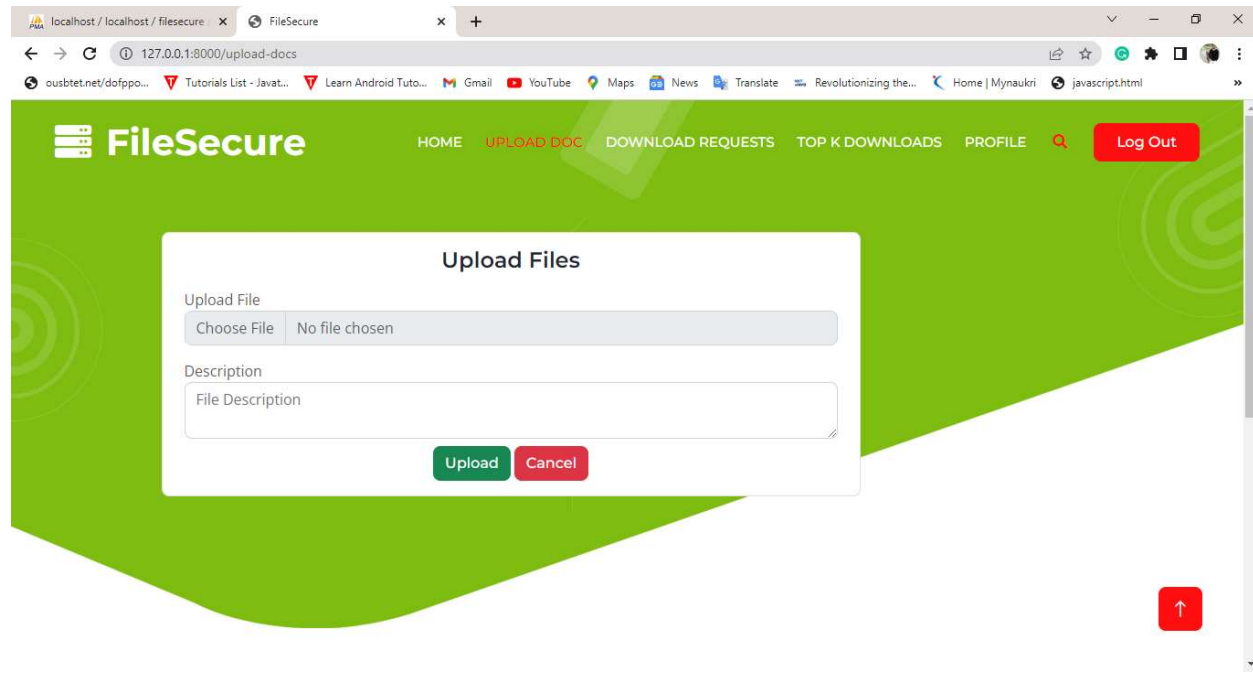
OUTPUT:



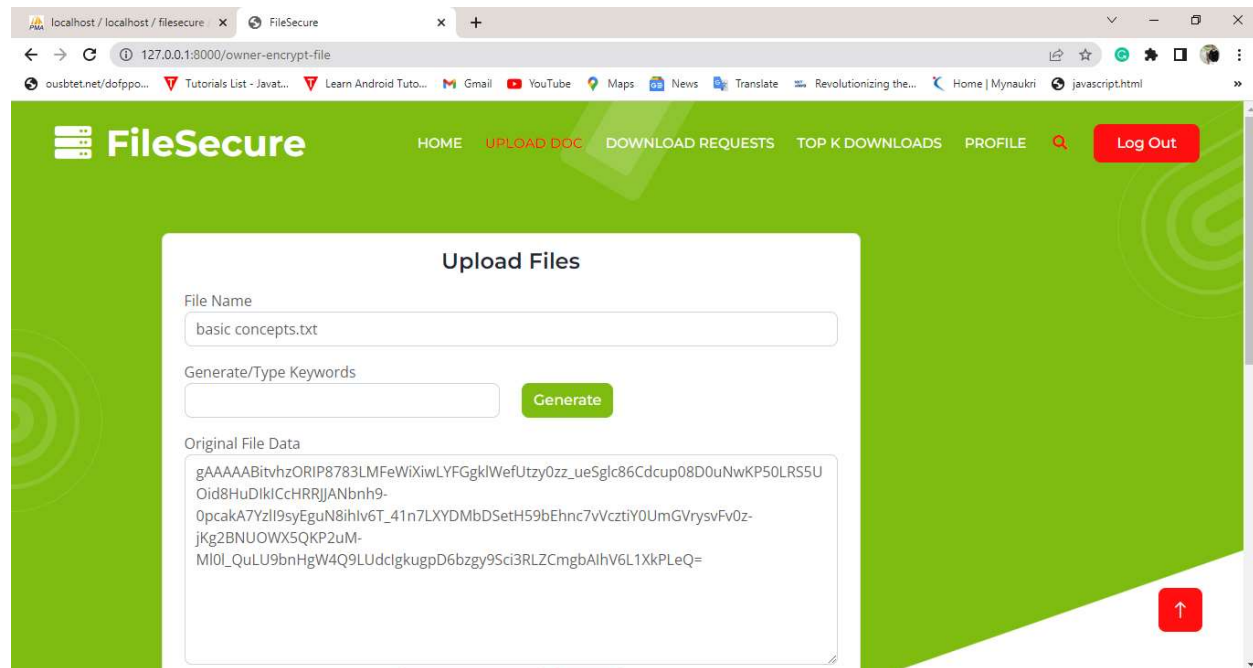
8.1 Data owner login page



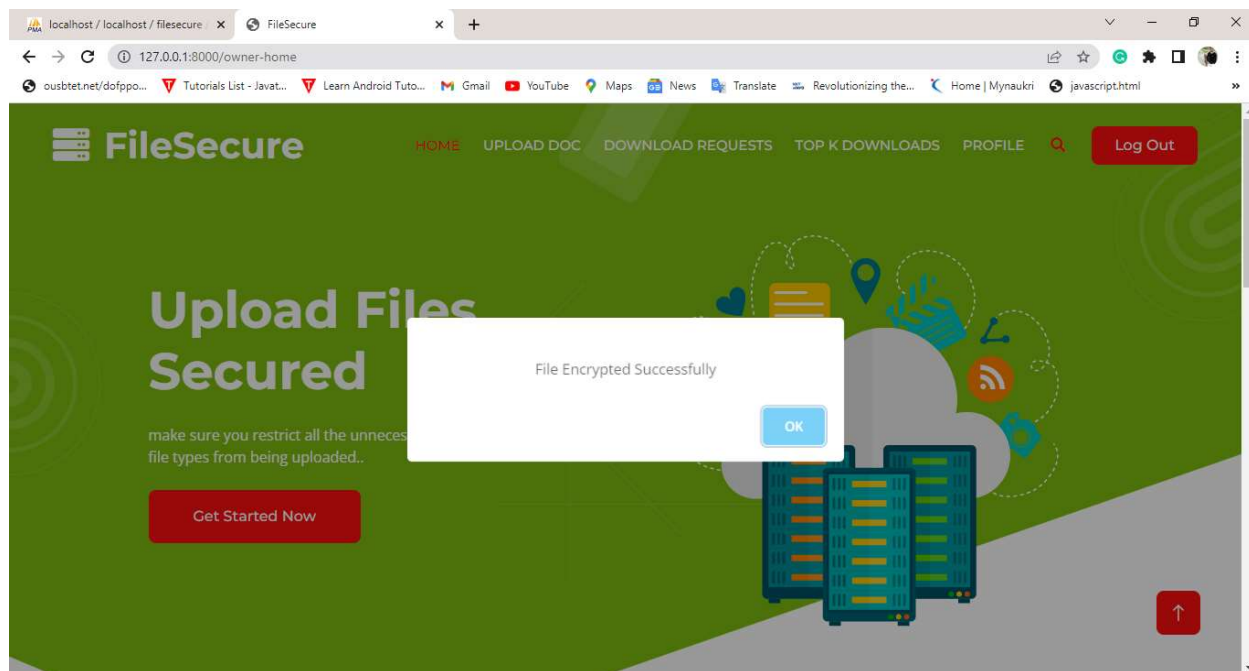
8.2 Data owner registration



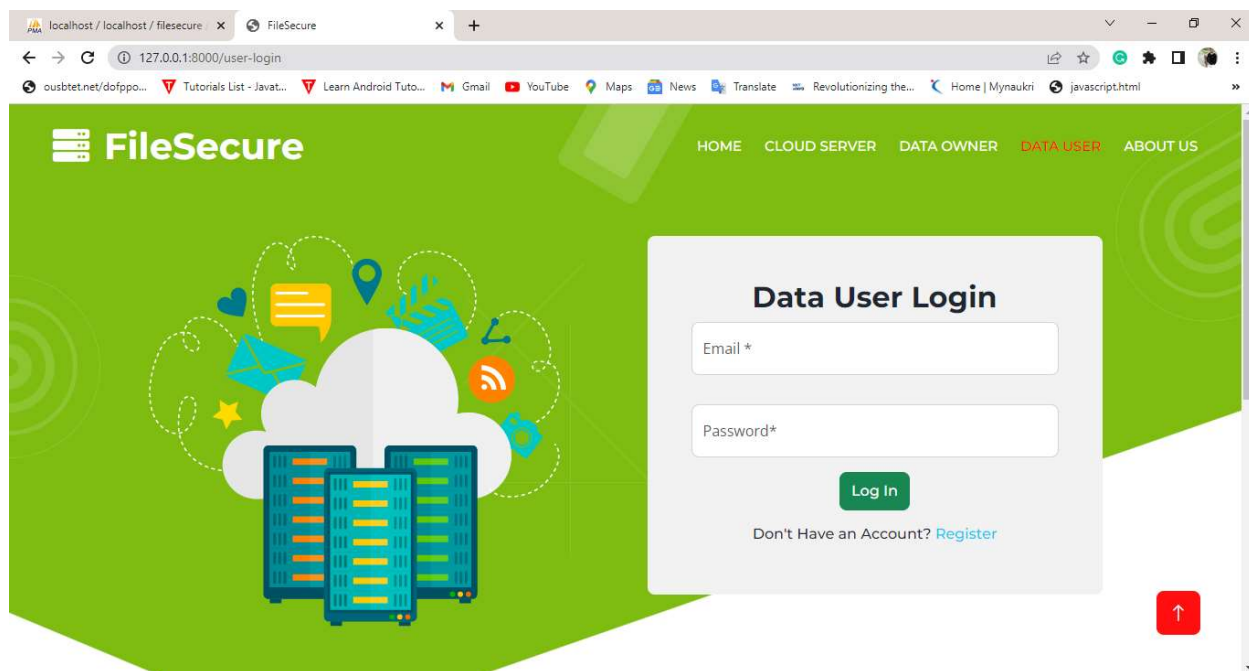
8.3 Data owner upload files



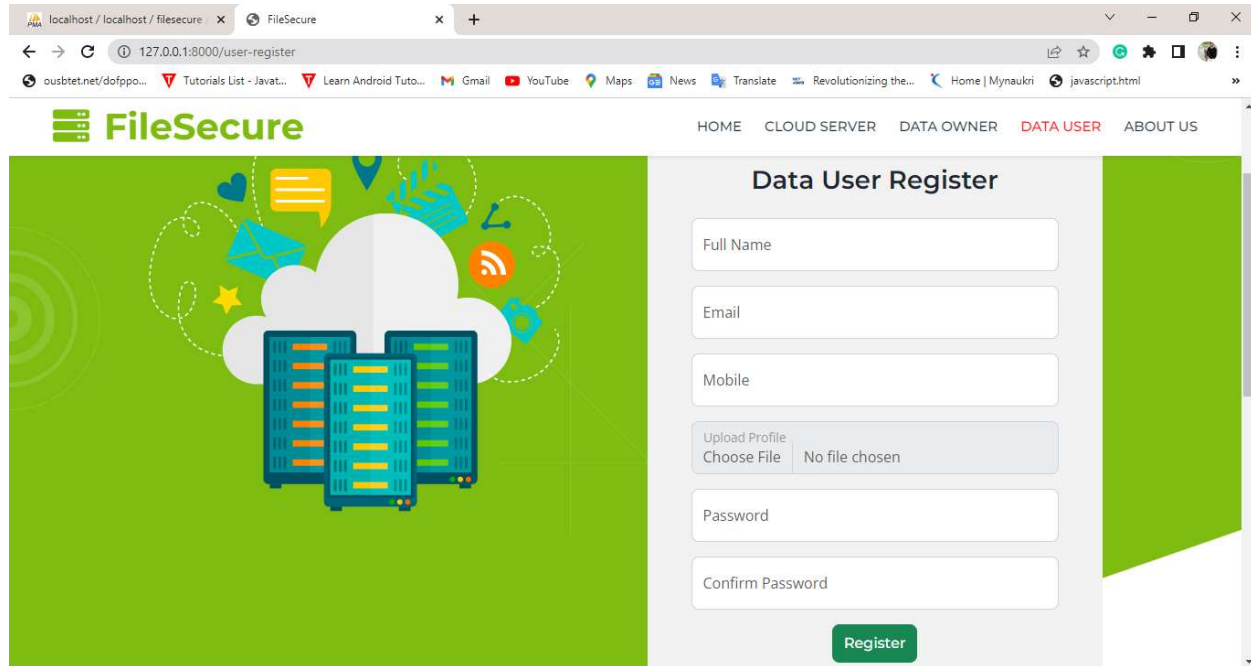
8.4 Generates Joint Keywords



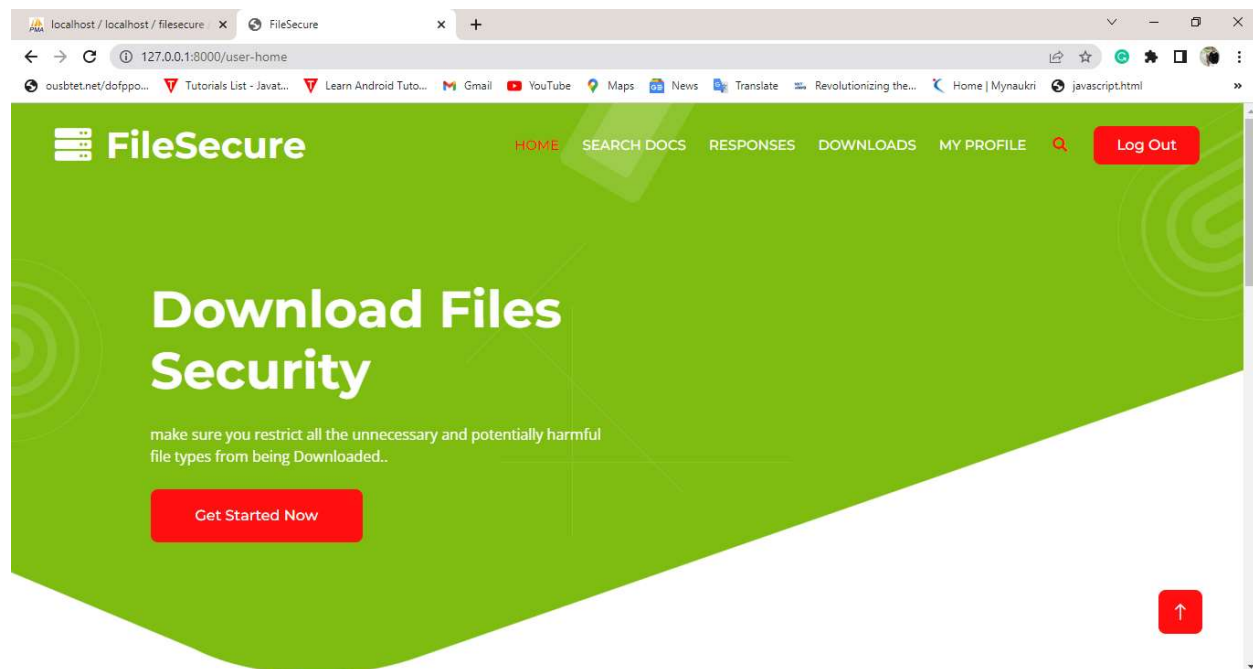
8.5 File Encryption



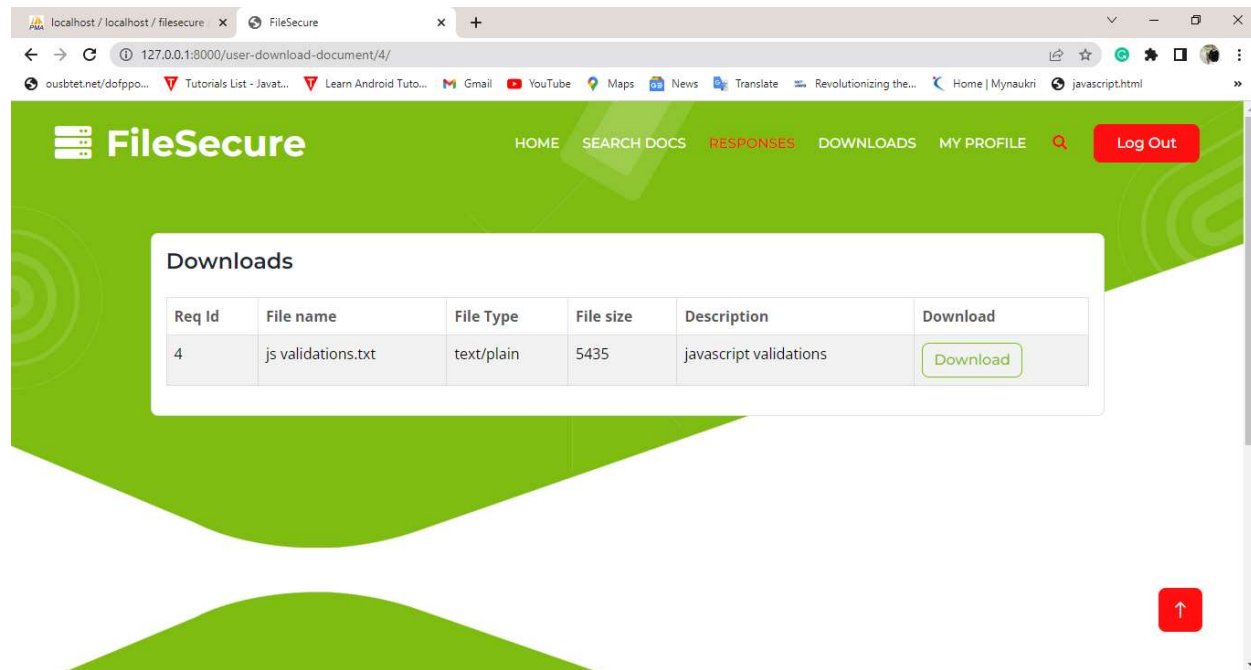
8.6 Data User Login



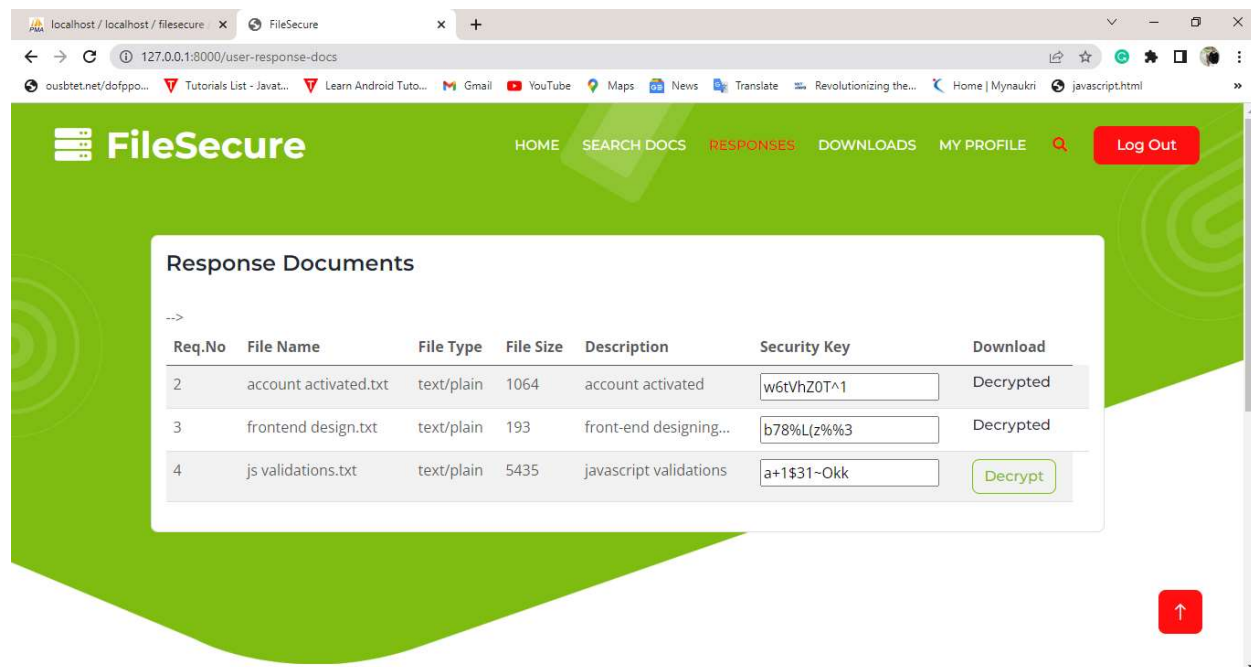
8.7 Data User Registration



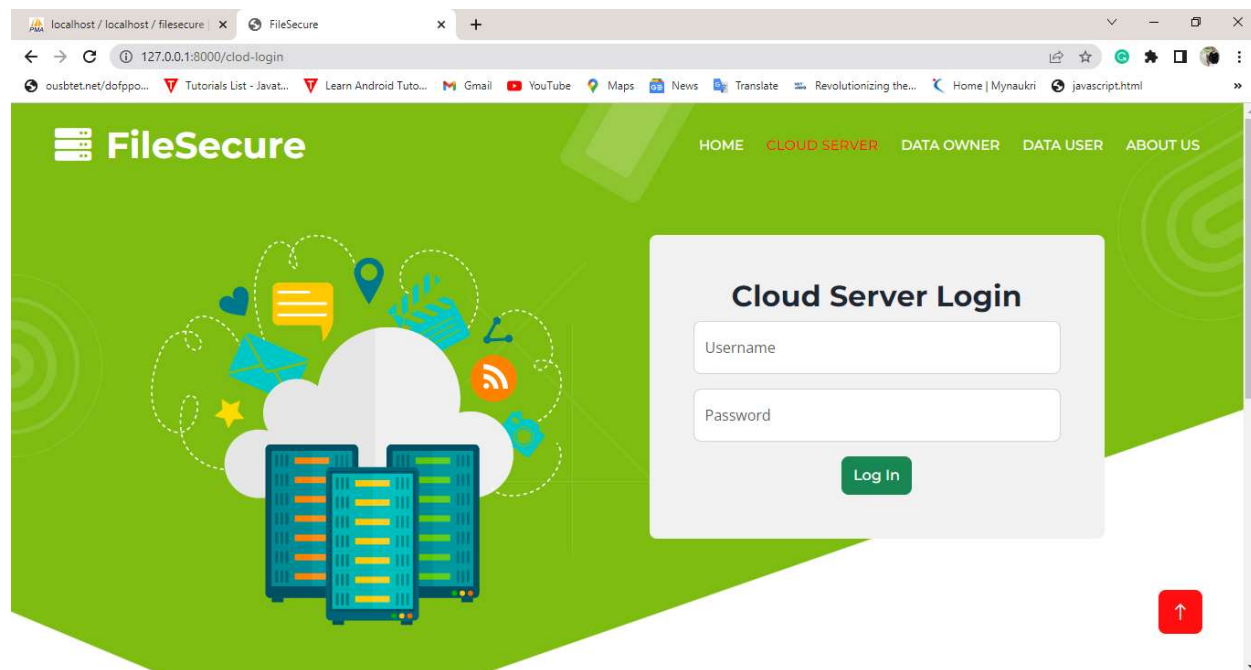
8.8 Downloading files



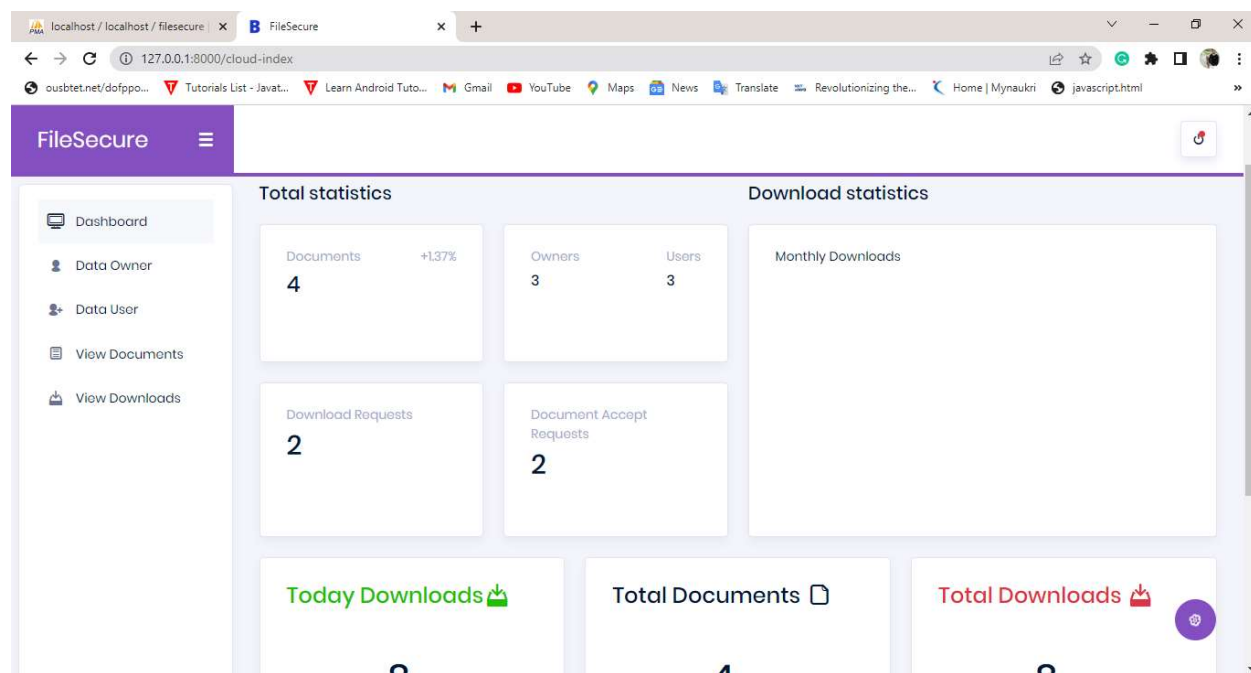
8.9 Downloading request.



8.10 Decryption using security key.



8.11 Cloud server login



8.12 Dashboard

The screenshot shows the FileSecure web application interface. The left sidebar contains a menu with options: Dashboard, Data Owner (selected), Data User, View Documents, and View Downloads. The main content area displays the 'Data Owner Details Table' with the following data:

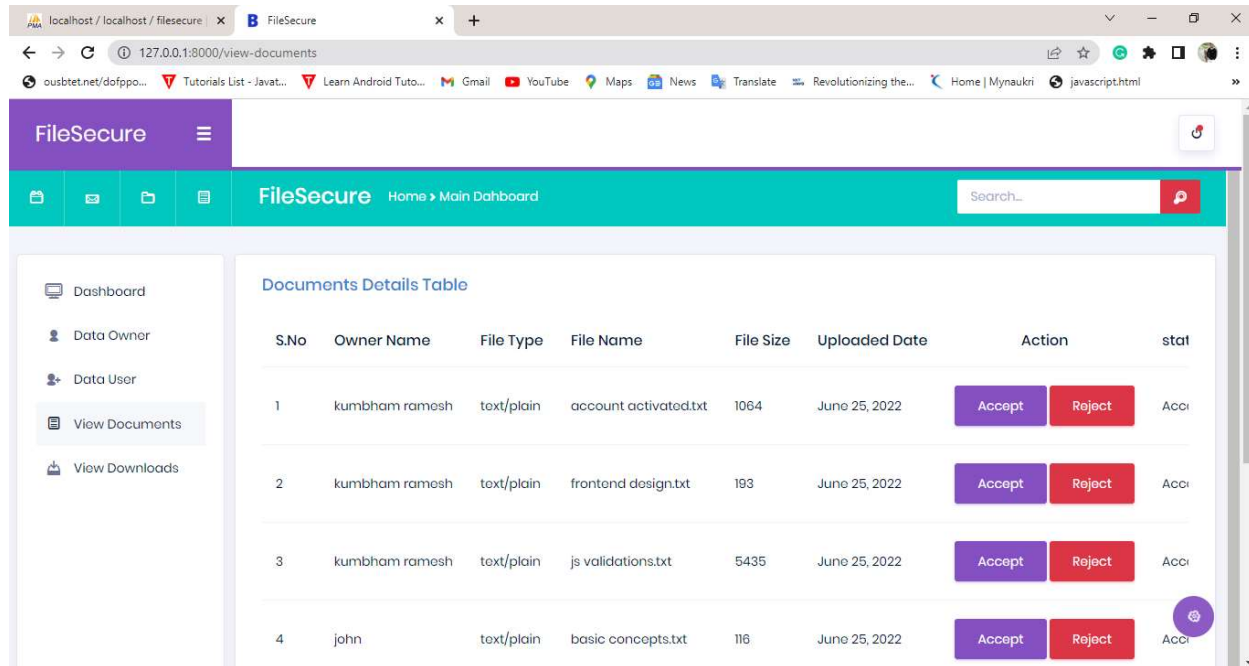
S.No	Owner Name	Mobile	Email	Status	Action
1	kumbham ramesh	7981751867	krq4585@gmail.com	Accepted	Accept Reject
2	tharun sai	8788831212	saitnarun581@gmail.com	Accepted	Accept Reject
3	mark	9999999999	mark@gmail.com	ponding	Accept Reject
7	john	8888888888	fazalsirprojects@gmail.com	Accepted	Accept Reject

8.13 Data owner Details

The screenshot shows the FileSecure web application interface. The left sidebar contains a menu with options: Dashboard, Data Owner, Data User (selected), View Documents, and View Downloads. The main content area displays the 'Data User Details Table' with the following data:

S.No	Profile	Name	Mobile	Email	Status
1		tharun	9876543210	saitnarun581@gmail.com	verified
2		ramesh k	7981751867	krq4585@gmail.com	verified
4		sam	9999999999000	fazalsirprojects@gmail.com	verified

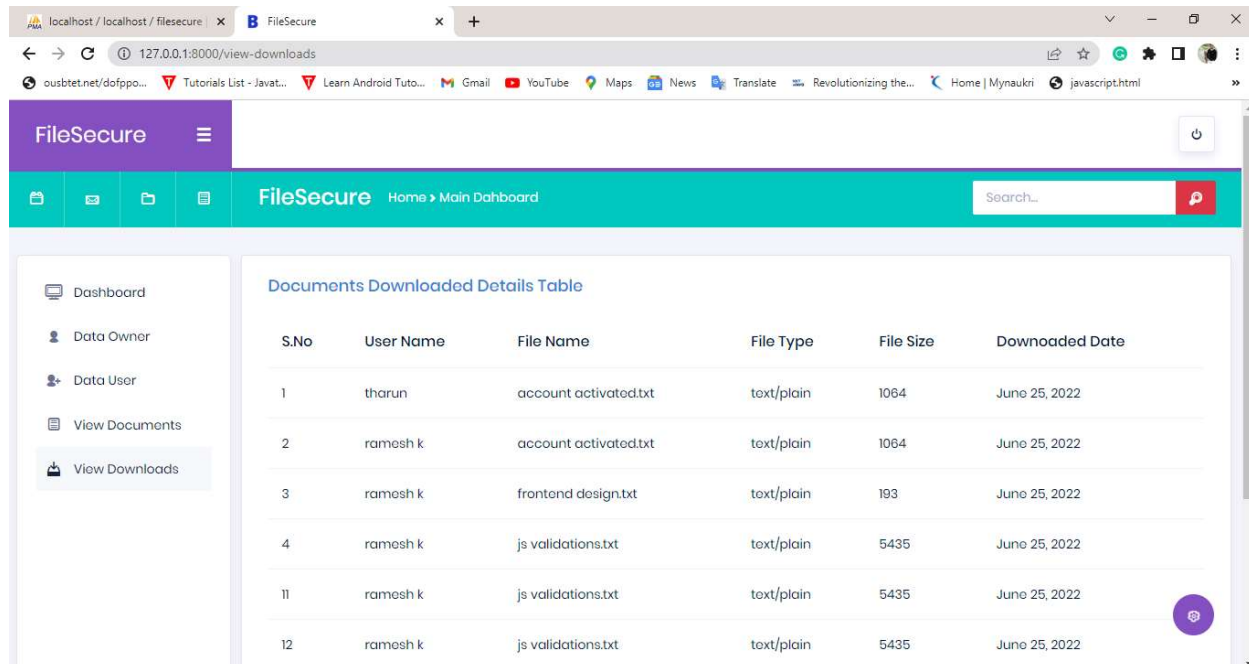
8.14 Data User details



The screenshot shows the FileSecure web application interface. The top navigation bar includes the FileSecure logo and a search bar. The left sidebar contains links to Dashboard, Data Owner, Data User, View Documents, and View Downloads. The main content area displays the 'Documents Details Table' with the following data:

S.No	Owner Name	File Type	File Name	File Size	Uploaded Date	Action	stat
1	kumbham ramesh	text/plain	account activated.txt	1064	June 25, 2022	Accept Reject	Acci
2	kumbham ramesh	text/plain	frontend design.txt	193	June 25, 2022	Accept Reject	Acci
3	kumbham ramesh	text/plain	js validations.txt	5435	June 25, 2022	Accept Reject	Acci
4	john	text/plain	basic concepts.txt	116	June 25, 2022	Accept Reject	Acci

8.15 Documents uploaded.



The screenshot shows the FileSecure web application interface. The top navigation bar includes the FileSecure logo and a search bar. The left sidebar contains links to Dashboard, Data Owner, Data User, View Documents, and View Downloads. The main content area displays the 'Documents Downloaded Details Table' with the following data:

S.No	User Name	File Name	File Type	File Size	Downloaded Date
1	tharun	account activated.txt	text/plain	1064	June 25, 2022
2	ramesh k	account activated.txt	text/plain	1064	June 25, 2022
3	ramesh k	frontend design.txt	text/plain	193	June 25, 2022
4	ramesh k	js validations.txt	text/plain	5435	June 25, 2022
11	ramesh k	js validations.txt	text/plain	5435	June 25, 2022
12	ramesh k	js validations.txt	text/plain	5435	June 25, 2022

8.16 Documents downloaded.

9. CONCLUSION

We propose an efficient search method using features to match joint keywords (FMJK) on encrypted cloud data. This method proposes that each d keywords are randomly selected from the non-duplicated keywords, which are extracted from the documents of the data owner, to generate a joint keyword, and all joint keywords form a keyword dictionary, which greatly reduces the dimension of the keywords dictionary. Because the dimension of creating indexes and trapdoors is related to the dimension of keyword dictionary, it also reduces the dimension of the key, the indexes and the trapdoors, which improves the search efficiency. In the process of creating each dimension of the indexes and the trapdoors, the document features and query keywords are accurately matched with the joint keywords in the keywords dictionary to get a weighted score, which ensures the accuracy of the query. And the reduction of dimension also reduces the overhead storage space occupied by the indexes and the trapdoors. The theoretical analysis and experimental results show that the proposed method is more feasible and more effective than the compared schemes.

10. FUTURE ENHANCEMENT

future enhancements for the efficient search method using feature matching on encrypted cloud data:

1. **Improved Encryption Techniques:** Explore advanced encryption techniques to enhance the security of the data while still allowing for efficient search functionality. Techniques such as homomorphic encryption or searchable symmetric encryption could be investigated.
2. **Optimized Indexing:** Develop more efficient indexing methods to speed up the search process. This could involve techniques such as distributed indexing, indexing on encrypted data, or utilizing machine learning algorithms to predict search patterns and optimize the index accordingly.
3. **Scalability:** Enhance the scalability of the system to handle larger volumes of data and more concurrent searches. This could involve redesigning the architecture to be more distributed or utilizing cloud-based solutions for scalability.
4. **Query Optimization:** Implement techniques to optimize search queries for better performance. This could involve query rewriting, query caching, or dynamic query optimization based on usage patterns.
5. **Privacy-Preserving Techniques:** Investigate additional privacy-preserving techniques to further protect user data. This could involve differential privacy mechanisms or data anonymization techniques.
6. **Integration with Other Services:** Explore integration with other cloud services or applications to provide additional functionality or enhance the user experience. For example, integrating with file sharing platforms or collaboration tools.
7. **User Authentication and Access Control:** Enhance user authentication and access control mechanisms to ensure that only authorized users can access the encrypted data and perform searches.
8. **Support for Multimedia Data:** Extend the search functionality to support searching encrypted multimedia data such as images, videos, or audio files. This could involve developing specialized feature extraction techniques for multimedia data.
9. **Cross-Platform Compatibility:** Ensure compatibility with a wide range of devices and platforms to maximize accessibility for users. This could involve developing dedicated client applications for different operating systems and devices.
10. **Continuous Monitoring and Improvement:** Continuously monitor system performance and user

feedback to identify areas for improvement and implement enhancements accordingly. This could involve conducting regular performance evaluations, collecting user feedback, and iterating on the design-based findings

11. REFERENCES

- [1] Z. Wan and R. H. Deng, "VPSearch: Achieving verifiability for privacy preserving multi-keyword search over encrypted cloud data," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 6, pp. 1083–1095, Nov./Dec. 2016.
- [2] Y. Yang, H. Lin, X. Liu, W. Guo, X. Zheng, and Z. Liu, "Blockchain based verifiable multi-keyword ranked search on encrypted cloud with fair payment," *IEEE Access*, vol. 7, pp. 140818–140832, 2019.
- [3] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multikeyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 829–837.
- [4] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2009, pp. 139–152.
- [5] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Trans. Commun.*, vol. E98.B, no. 1, pp. 190–200, 2015.
- [6] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Jan. 2016.
- [7] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3025–3035, Nov. 2014.
- [8] L. Liu and Z. Liu, "A novel fast dimension-reducing ranked query method with high security for encrypted cloud data," *Chin. J. Electron.*, vol. 29, no. 2, pp. 344–350, Mar. 2020.
- [9] W. Zhang, Y. Lin, and G. Qi, "Catch you if you misbehave: Ranked keyword search results verification in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 74–86, Mar. 2015.
- [10] Z. Guan, X. Liu, L. Wu, J. Wu, R. Xu, J. Zhang, and Y. Li, "Cross-lingual multi-keyword rank search with semantic extension over encrypted data," *Inf. Sci.*, vol. 514, pp. 523–540, Apr. 2020.
- [11] M. Murata, H. Nagano, R. Mukai, K. Kashino, and S. Satoh, "BM25 with exponential IDF for instance search," *IEEE Trans. Multimedia*, vol. 16, no. 6, pp. 1690–1699, Oct. 2014.
- [12] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2000, pp. 44–55.
- [13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934,

Jan. 2011, doi: 10.3233/JCS-2011- 0426.

[14] R. Li, Z. Xu, W. Kang, K. C. Yow, and C.-Z. Xu, “Efficient multi-keyword ranked query over encrypted data in cloud computing,” *Future Gener. Comput. Syst.*, vol. 30, no. 1, pp. 179–190, Jan. 2014, doi: 10.1016/j. future.2013.06.029.

[15] J. Wang, H. Ma, T. Qiang, L. Jin, H. Zhu, S. Ma, and X. Chen, “A new efficient verifiable fuzzy keyword search scheme,” *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 3, no. 4, pp. 61–71, Dec. 2012.

[16] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, “Toward efficient multikeyword fuzzy search over encrypted outsourced data with accuracy improvement,” *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Jul. 2016, doi: 10.1109/TIFS.2016.2596138.

[17] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, “Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data,” *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1874–1884, Aug. 2017, doi: 10.1109/TIFS.2017.2692728.

[18] R. Zhao, H. Li, Y. Yang, and Y. Liang, “Privacy-preserving personalized search over encrypted cloud data supporting multi-keyword ranking,” in *Proc. 6th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2014, pp. 1–6.

[19] C. Chen, X. Zhu, P. Shen, J. Hu, S. Guo, Z. Tari, and A. Y. Zomaya, “An efficient privacy-preserving ranked keyword search method,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.

[20] RFC Index. Accessed: May 25, 2020. [Online]. Available: <https://www.rfc-editor.org/rfc-index-100a.html>

12. PROJECT WORK MAPPING WITH PROGRAMME OUTCOMES

PROGRAMME OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

1. Organize, maintain and protect IT Infrastructural resources.
2. Design and Develop web, mobile, and smart apps based software solutions to the real world problems.

PROJECT PROFORMANCE

Classification of Project	Application	Product	Research	Review
	✓			

Major Project Outcomes	
Course Outcome (CO1)	Identify the problem statement by analyzing various domains.
Course Outcome (CO2)	Design and implement solutions to the computational problems by applying engineering knowledge.
Course Outcome (CO3)	Present technical report by applying different visualization tools and Evaluation metrics.
Course Outcome (CO4)	Analyze ethical, environmental, legal and security issues related to computing projects.

12.1 Mapping table

IT3523: MAJOR PROJECT														
Course Outcomes	Program Outcomes and Program Specific Outcomes													
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12		PSO1 PSO2
CO1	3	3		2		3			3	3	3			
CO2	3		3	3	3	3	2		3	3	3	3		3 3
CO3					2				3	3	3	3		3 3
CO4						3	3	3				3		3 3
Average	3	3	3	2	2	3	2	3	3	3	3	3		3 3

1- Slightly(Low) mapped

2- Moderately (Medium) mapped

3-Substantially (High) mapped

PROGRAMME OUTCOMES	Mapping HIGH/MEDIUM/LOW	JUSTIFICATION
1	2	apply the knowledge of mathematics.
2	3	considering a problems of data storage and analyze those problems and developing Algorithms.
3	3	is project meets the desired specification of the society
4	1	we have created this user interface by using Django .
5	3	using the python technology Access we have created databases for storing data and user friendly interface.

6	3	this developing process we were able to meet the local challenges as well as global challenges.
7	3	is interface does not provide benefits to all types of users which helps for the society.
8	3	will provide some ethical, social behavior in some aspects.
9	3	The work is done by team to function on multi-disciplinary team.
10	3	our project is done in all aspects like communicating and documenting effectively.
11	1	our project is developed by python technology languages and it will engage in lifelong learning.
12	3	We find a solution to our problem by developing an application, which is effective for financial management.

PROGRAM SPECIFIC OUTCOMES

PSOs	1	2
PROJECT	H	H

PROGRAM SPECIFIC OUTCOMES	Mapping HIGH/MEDIUM/LOW	JUSTIFICATION
1	HIGH	Project involves maintain the

		data in cloud server and provide security to the data , a robust data management system is essential.
2	HIGH	Creating intuitive and user-friendly interfaces for web, is crucial for engaging users.

ISSN: 0253-7214

**Journal of Advanced
Zoology**

Acceptance Letter

Date: 12-02-2024

Title: " A ROBUST SEARCH METHOD USING FEATURES TO DETERMINE COMBINED KEYWORDS ON CLOUD ENCRYPTED DATA".

Y.K. Viswanadham¹ , G Naga Lakshmi², G Dinesh Kumar³, B Archana⁴,B Sravanthi⁵

Author/s,

We are pleased to inform you that above-mentioned manuscript has been reviewed and accepted for publication in the upcoming issue (2024) of the **Journal of Advanced Zoology**, ISSN 0253-7214. This letter of acceptance is to be considered as the official acceptance of your manuscript with no further amendments required. Author/s are requested to follow ethical and privacy guidelines mentioned in the journal homepage (<https://jazindia.com/>).

Thank you for your contribution to the Journal.

After receiving the publication fee, we will email you the galley proof for your final confirmation.

Regards,



Dr. S.P. Tripathi
Chief Editor

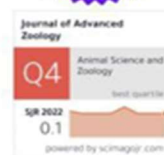
jazindiagkp@gmail.com

<https://jazindia.com/>

<https://www.scopus.com/sourceid/22525>

<https://mjl.clarivate.com/search-results>

<https://www.scimagojr.com/journalsearch.php?q=22525&tip=sid&clean=PO>



JOURNAL OF ADVANCED ZOOLOGY: (AN INTERNATIONAL JOURNAL OF ZOOLOGICAL RESEARCH), 305 -C, Azad Nagar Colony, Rustampur,

P.O-Shivpuri, Gorakhpur- 273 016, India

[A ROBUST SEARCH METHOD USING FEATURES TO DETERMINE COMBINED KEYWORDS ON CLOUD ENCRYPTED DATA

Y.K. Viswanadham¹ Associate Professor, G Naga Lakshmi², G Dinesh Kumar³, B Archana⁴,

B Sravanthi⁵

Department of Information Technology
Seshadri Rao Gudlavalluru Engineering College, Gudlavalluru
Andhra Pradesh-521356, India

ykvnath@gmail.com

guralanagalakshmi17@gmail.com

dineshkumargudiputi777@gmail.com

archana.bolla2210@gmail.com

sravanthisravanthi4091@gmail.com

ABSTRACT: Users are more comfortable trusting their sensitive information to the cloud as its security continues to improve. However, when there are several encrypted files, each with its own set of keywords for indexing, the storage overhead grows exponentially, and search efficiency suffers. Therefore, this work provides a technique for searching encrypted cloud data that makes use of features to match joint keywords (FMJK). Joint keywords are generated by randomly selecting a subset of the data owner's non-duplicated keywords choice among the documents' extracted keywords; together, these keywords form a keyword dictionary. Every combined keyword matches with a document's feature as well as a query keyword, making the former's result considered a dimension of a document's index with the latter's result considered a dimension about the query trapdoor. Its BM25 method is then utilized for arranging the top k results by the inner product between the document index and the trapdoor.

Key words: *Encrypted cloud data, feature matching, searchable encryption, dimensionality reduction, joint keywords.*

1. INTRODUCTION

With the fast growth of science and technology, organizations or individuals increasingly depend on keeping a huge number of data documents on cloud servers for the purpose to transfer data swiftly and remotely. But as use of cloud services grows, storage costs rise, search efficiency declines, and privacy safeguards become an area of study. KNN (K Nearest Neighbor) technology is utilized to construct indexes allowing cipher-text retrieval in the majority of current cipher-text sorting retrieval techniques. Most search encryption systems need a lot of space and time to implement since they are directly tied to the encrypted key, the document's index, plus the query request dimensions involved in searching encrypted material. Encrypting less data in fewer dimensions is one way to speed up the search process. The retrieval needs of a huge quantity of data are still beyond the capabilities of current research, and neither can academics sort and filter valuable data only authorized users. Since there are several types of users, it is critical to develop a system that can maintain privacy, boost retrieval efficiency, and improve query accuracy.

In this research, we offer the MRSE (Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data) technique, which is a search method that uses features to match joint keywords (FMJK) on encrypted cloud data. Initially each document's topic has to be expressed by the characteristics extracted from it. Then, d random keywords are generated from the data owner's documents' non-duplicated keywords to create a joint keyword, and the entire joint keywords constitute a keyword dictionary, and finally, the features of each document are combined with the joint keywords to generate an index. The search query keywords are entered by the allowed user, matched against the joint keywords to generate a trapdoor, and then the safe inner product between the trapdoor and index is computed to yield the top k results. The article in question makes the following contributions: (1) Each set of d randomly chosen keywords forms a joint keyword that corresponds to a characteristic of the document that is to be mapped with a particular index dimension, thus decreasing the overall dimension size of the key, the index, as well as the trapdoor, streamlining the matrix operation throughout