STORING DATA IN ONLINE WEBSITE

A Real Time Research Project Report Submitted to

Jawaharlal Nehru Technological University Hyderabad

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

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING – (CYBERSECURITY)

By

V.Vinay kumar	**(23E11A6252)**
S. vamshi	**(23E11A6240)**
K.keerthan prasad	**(23E11A6215)**
B. Charan	**(23E11A6205)**

Under the guidance of

Ms Shivani
Assistant Professor
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(CYBER SECURITY)
BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution)
Accredited by NAAC 'A' Grade, Accredited by NBA (UG Programmes: CSE, ECE, EEE & Mechanical) Approved by AICTE, Affiliated to JNTUH Hyderabad
Mangalpally, ibrahimpatanam -501 510, Hyderabad, Telangana

JUNE 2025

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING: BIET
B.Tech.(CSE) (Cybersecurity)

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution)

Accredited by NAAC 'A' Grade, Accredited by NBA (UG Programmes: CSE, ECE, EEE & Mechanical) Approved by AICTE, Affiliated to JNTUH Hyderabad
Mangalpally , ibrahimpatanam-501 510, Hyderabad, Telangana

Certificate

This is to certify that the Real Time Research Project work entitled STORING DATA IN ONLINE WEBSITE is the bonafide work done

By

Vinay kumar	**(23E11A6252)**
S. Vamshi	**(23E11A6240)**
K.Keerthan prasad	**(23E11A6215)**
B. Charan	**(23E11A6205)**

in the Department of Computer Science and Engineering, ***BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY,*** *Ibrahimpatnam is submitted to* ***Jawaharlal Nehru Technological University, Hyderabad*** *in partial fulfillment of the requirements for the award of* ***B.Tech*** *degree in* ***Computer Science and Engineering*** *during* ***2024- 2025.***

Supervisor:
Ms Shivani
Assistant Professor
Dept of: Computer Science and Engineering,
Bharat Institute of Engineering and Technology
Ibrahimpatnam–501 510, Hyderabad

Department I/C
Mrs Nazneen Fathima
Professor
Dept of: Computer science and engineering - cyber security,
Bharat Institute of Engineering and Technology
Ibrahimpatnam–501 510, Hyderabad

Viva-Voce held on……

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We avail this opportunity to express our deep sense of gratitude and hearty thanks to

Sri CH. Venugopal Reddy, Chairman of BIET, for providing congenial atmosphere and encouragement.

We would like to thank **Prof. G. Kumaraswamy Rao**, Former Director & O.S. of DLRL Ministry of Defence, Sr. Director R&D, BIET, and **Dr. V Srinivasa Rao**, Dean of CSE, for having provided all the facilities and support.

We would like to thank our Department Incharge / HOD **Mrs. Nazneen Fathima** , for encouragement at various levels of our Project.

We are thankful to our Project Coordinator **Ms Sangeetha C,** Assistant Professor, Computer Science and Engineering for their support and cooperation throughout the process of this project.

We are thankful to our guide **Ms Shivani**, Assistant Professor, ECE for her sustained inspiring Guidance and cooperation throughout the process of this project. Her wise counsel and suggestions were invaluable.

We express our deep sense of gratitude and thanks to all the Teaching and Non-Teaching Staff of our college who stood with us during the project and helped us to make it a successful venture.

We place highest regards to our Parent, our Friends and Well-wishers who helped a lot in making the report of this project

V.Vinay kumar	**(23E11A6252)**
S. vamshi	**(23E11A6240)**
K.keerthan prasad	**(23E11A6215)**
B. Charan	**(23E11A6205)**

Declaration

We hereby declare that this Real Time Research Project is titled "**STORING DATA IN ONLINE WEBSITE"** is a genuine Real Time Research Project work carried out by us, in **B.Tech (Cyber security)** degree course of **Jawaharlal Nehru Technology University Hyderabad, Hyderabad** and has not been submitted to any other course or university for the award of my degree by me.

Signatures of the Project team members

1.

2.

3.

4.

ABSTRACT

Our project focuses on developing a web-based platform for online data storage using HTML, JavaScript, and CSS. This website enables users to upload, manage, and access their files from any device with an internet connection, eliminating the need for local storage. The platform leverages browser-based storage solutions and client-side scripting to facilitate seamless file management. Users can organize their data efficiently while ensuring accessibility from anywhere. With a simple and user-friendly interface, the system enhances ease of use, making it suitable for both personal and collaborative purposes.

Although cloud storage typically involves complex server-side infrastructure, our approach focuses on front-end technologies, utilizing local storage and third-party APIs where necessary. Security measures such as encryption and authentication mechanisms will be incorporated to protect user data. Despite certain limitations, such as storage capacity constraints and dependency on client-side technologies, this project serves as a fundamental model for lightweight and accessible online storage solutions. Future enhancements could include integrating server-side technologies to expand functionality and scalability.

Keywords:

HTML, CSS, Javascript, Uploadcare API , Processor INTEL(3,5,7,9)/RYZEN(5,7,9).

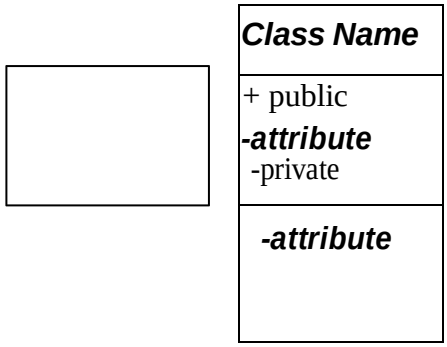
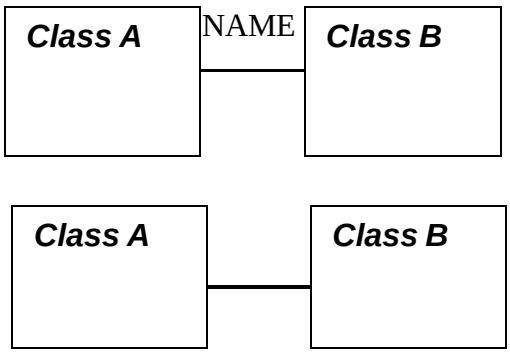
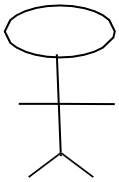
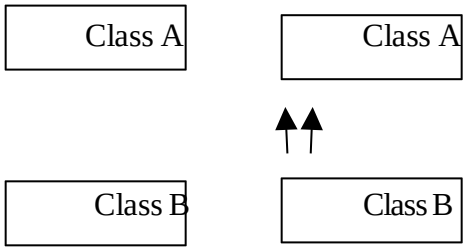
LIST OF CONTENTS

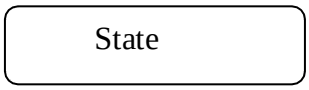
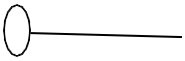
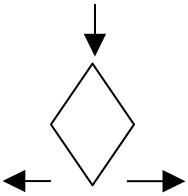
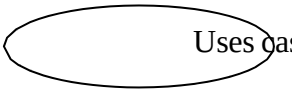
LIST OF FIGURES

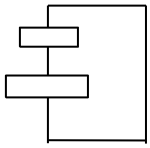
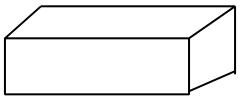
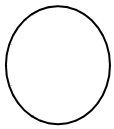
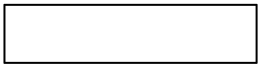
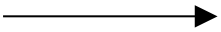
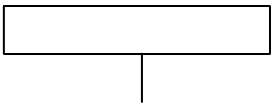
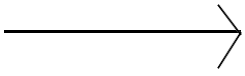
ABBREVIATIONS

S. No.	Short form	Abbreviation
1.	APP	Application
2.	HTML	Hyper text Markup Language
3.	CSS	Cascading Styling sheets
4.	JS	JavaScript
5.	API	Application programming interface
6.	UML	Unified Modeling Language
7.	DFD	Data Flow Diagram
8.	E-mail	Electronic mail
9.	ID	Identity
10.	DB	Data Base

LIST OF SYMBOLS

S.No	Name	Notation	Description
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external environment

5.	Relation (uses)	Uses	Used for additional process communication.
6.	Relation (extends)	Extends 	Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Usecase		Interact ion between the system and external environment.

14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which area collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard,sensors,etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message	Message 	Represents the message exchanged.

1. INTRODUCTION

Cloud file storage offers convenient, scalable, and secure data management. Users can access files anywhere, benefiting from reliable redundancy and disaster recovery measures. Security features like encryption and access controls protect against unauthorized access. It's cost-effective with pay-as-you-go models, integrating seamlessly with other cloud services for enhanced collaboration and automation. Compliance certifications ensure adherence to regulatory standards. Cloud storage also supports efficient backup and recovery, making it ideal for individuals and businesses seeking flexible, reliable data storage solutions.

1.1 Objective :

To access their files from any location with internet connectivity, promoting seamless collaboration and remote work capabilities. To Provide scalable storage solutions that can accommodate varying amounts of data, allowing users to expand or reduce storage capacity based on their needs without upfront infrastructure investments. To Implement robust security measures such as encryption, access controls, and data redundancy to protect stored files from unauthorized access, data breaches, and ensure high availability and reliability of data access.

1.2 Scope of project :

A project on file storage in the cloud encompasses planning, implementation, and management of scalable, secure storage solutions. It involves assessing storage needs, selecting suitable cloud providers, and configuring storage architecture. Security measures like encryption, access controls, and compliance with regulations are crucial. Integration with existing systems and applications ensures seamless workflows. Ultimately, the project aims to leverage cloud technologies for reliable, accessible, and cost-effective file storage solutions.

1.3 Existing system:

The existing system of file storage in the cloud encompasses a wide array of providers offering scalable storage solutions accessible via internet connectivity. These services include robust options for data redundancy across multiple servers and data centers to ensure high availability and minimize the risk of data loss.

2. RELATED WORK

Data security and privacy protection for cloud storage. Authors are P.Yang, N. Xiong and J. Ren. new development trends including Internet of Things (IoT), smart city, enterprises digital transformation and world's digital economy are at the top of the tide. The continuous growth of data storage pressure drives the rapid development of the entire storage market on account of massive data generated. By providing data storage and management, cloud storage system becomes an indispensable part of the new era. Currently, the governments, enterprises and individual users are actively migrating their data to the cloud. Such a huge amount of data can create magnanimous wealth. However, this increases the possible risk, for instance, unauthorized access, data leakage, sensitive information disclosure and privacy disclosure. Although there are some studies on data security and privacy protection, there is still a lack of systematic surveys on the subject in cloud storage system. In this paper, we make a comprehensive review of the literatures on data security and privacy issues, data encryption technology, and applicable countermeasures in cloud storage system. Specifically, we first make an overview of cloud storage, including definition, classification, architecture and applications. Secondly, we give a detailed analysis on challenges and requirements of data security and privacy protection in cloud storage system. Thirdly, data encryption technologies and protection methods are summarized. Finally, we discuss several open research topics of data security for cloud storage. **[1].**

Efficient revocable multi-authority attribute-based encryption for cloud storage. Authors are Y. Ming, B. He and C. Wang Pu. is known, attribute-based encryption (ABE) is usually adopted for cloud storage, both for its achievement of fine-grained access control over data, and for its guarantee of data confidentiality. Nevertheless, single-authority attribute-based encryption (SA-ABE) has its obvious drawback in that only one attribute authority can assign the users' attributes, enabling the data to be shared only within the management domain of the attribute authority, while rendering multiple attribute authorities unable to share the data. On the other hand, multi-authority attribute-based encryption (MA-ABE) has its advantages over SA-ABE. It can not only satisfy the need for the fine-grained access control and confidentiality of data, but also make the data shared among different multiple attribute authorities. However, existing MA-ABE schemes are unsuitable for the devices with resources-constraint, because these schemes

are all based on expensive bilinear pairing. Moreover, the major challenge of MA-ABE scheme is attribute revocation. So far, many solutions in this respect are not efficient enough. In this paper, on the basis of the elliptic curves cryptography, we propose an efficient revocable multi-authority attribute-based encryption (RMA-ABE) scheme for cloud storage. The security analysis indicates that the proposed scheme satisfies indistinguishable under adaptive chosen plaintext attack assuming hardness of the decisional Diffie-Hellman problem. Compared with the other schemes, the proposed scheme gets its advantages in that it is more economical in computation and storage **[2].**

Blockchain based deduplication and integrity auditing over encrypted cloud storage. Authors are M. Song, Z. Hua, Y. Zeng, H.Huang and X.Jia. Cloud computing promises great advantages in handling the exponential data growth. Secure deduplication can greatly improve cloud storage efficiency while protecting data confidentiality. In the meantime, when data are outsourced to the remote cloud, there is an imperative need to audit the integrity. Most existing works only consider the support for either secure deduplication or integrity auditing. Recently, there have been some research efforts aiming to integrate secure deduplication with integrity auditing. However, prior works are unsatisfactory in that they suffer from the leakage of ownership privacy and forgeability of auditing results for low-entropy data. In this paper, we propose a new scheme that delicately bridges secure deduplication and integrity auditing in encrypted cloud storage. In contrast with prior works, our scheme protects the ownership privacy and prevents the cloud service provider from forging the auditing results for low-entropy data. Furthermore, we propose a blockchain-based mechanism that helps to ensure key recoverability and reduce local storage cost of keys. Formal analysis is provided to justify the security guarantees. Experiment results demonstrate the modest performance overhead of our scheme. **[3].**

Web Cloud: Web-Based cloud storage for secure data sharing across platforms Authors are S.Sun, H.Ma, Z.Song and. With more and more data moving to the cloud, privacy of user data have raised great concerns. Client-side encryption/decryption seems to be an attractive solution to protect data security, however, the existing solutions encountered three major challenges: low security due to encryption with low-entropy PIN, inconvenient data sharing with traditional encryption algorithms, and poor usability with dedicated software/plugins that require certain types of terminals. This work designs and

implements WebCloud, a practical browser-side encryption solution, leveraging modern Web technologies. It solves all the above three problems while achieves several additional remarkable features: robust and immediate user revocation, fast data processing with offline encryption and outsourced decryption. Notably, our solution works on any device equipped with a Web user agent, including Web browsers, mobile and PC applications. We implement WebCloud based on ownCloud for basic file management utility, and utilize WebAssembly and Web Cryptography API for complex cryptographic operations integration. Finally, comprehensive experiments are conducted with many well-known browsers, Android and PC applications, which indicates that WebCloud is cross-platform and efficient. As an interesting by-product, the design of WebCloud naturally embodies a dedicated and practical ciphertext-policy attribute-based key encapsulation mechanism (CP-AB-KEM) scheme, which can be useful in other applications **[4].**

Sanitizable access control system for secure cloud storage against malicious data publishers. Authors are W.Susilo, P.Jiang, J.lai,G.yang and R.H.Deng. Cloud computing is considered as one of the most prominent paradigms in the information technology industry, since it can significantly reduce the costs of hardware and software resources in computing infrastructure. This convenience has enabled corporations to efficiently use the cloud storage as a mechanism to share data among their employees. At the first sight, by merely storing the shared data as plaintext in the cloud storage and protect them using an appropriate access control would be a nice solution. This is assuming that the cloud is fully trusted for not leaking any information, which is impractical as the cloud is owned by a third party. Therefore, encryption is mandatory, and the shared data will need to be stored as a ciphertext using an appropriate access control. However, in practice, some of these employees may be malicious and may want to deviate from the required sharing policy. The existing protection in the literature has been explored to allow only legitimate recipients to decrypt the contents stored in the cloud storage, but unfortunately, no existing work deals with issues raised due to the presence of malicious data publishers. Malicious data publishers construct data following the given policy, but the ciphertexts can actually be decrypted by unauthorized users without valid keys, or simply, anyone else who is unauthorized. The impact of the involvement of malicious data publishers is detrimental, as it may damage intellectual properties from the corporations. Therefore, it

remains an elusive research problem on how to enable a sound approach to resolve the issue when malicious data publishers are involved in the system, which is a very practical question. In this work, we present a new direction of research that can cope with the presence of malicious data publishers. We resolve the aforementioned problem by proposing the notion of Sanitizable Access Control System (SACS), which is designed for a secure cloud storage that can also resist against malicious data publishers. We define the threat model and its formal security model, as well as its design and scheme which is based on q-Parallel Bilinear Diffie-Hellman Exponent Assumption. We provide the security proof of our construction as well as its performance analysis. We believe that this work has opened a new area of research which has never been explored before, even though it is very practical. Therefore, this work will enhance the adoption of secure cloud storage in practice. Show Less **[5].**

A secure and efficient data deduplication schema with dynamic ownership management in cloud computing. Authors are X. ma,W. Yang, Y. Zhu and Z. Bai. Encrypted data deduplication is an important technique for saving storage space and network bandwidth, which has been widely used in cloud storage. Recently, a number of schemes that solve the problem of data deduplication with dynamic ownership management have been proposed. However, these schemes suffer from low efficiency when the dynamic ownership changes a lot. To this end, in this paper, we propose a novel server-side deduplication scheme for encrypted data in a hybrid cloud architecture, where a public cloud (Pub-CSP) manages the storage and a private cloud (Pri-CSP) plays a role as the data owner to perform deduplication and dynamic ownership management. Further, to reduce the communication overhead we use an initial uploader check mechanism to ensure only the first uploader needs to perform encryption, and adopt an access control technique that verifies the validity of the data users before they download data. Our security analysis and performance evaluation demonstrate that our proposed server-side deduplication scheme has better performance in terms of security, effectiveness, and practicability compared with previous schemes. Meanwhile, our method can efficiently resist collusion attacks and duplicate faking attacks **[6].**

Cloud storage monitoring system analyzing through file access authors are A.Augustus Devarajan, T.Sudalaimuthu. Cloud computing is an important technology on current

demanding business requirements and it has been emerged as unavoidable technology. The usage of IaaS Service storage for Cloud Computing is being expanding exponential every year. The cloud storages are used by the cloud user due to its economy compared with other storage methods. The replications of files helps user for easy access with high availability which reduces the overall access time of the files, but at the same time it occupies more storage space and result in high storage cost. The cloud user holds multiple times of the storage than what he is actually needed. It is a dire need of system to find unwanted files in the cloud and also optimize the storage space by evaluating through file access frequency.This paper propose Cloud Storage Monitoring (CSM) system, which monitor the IaaS storage usage and analyze the file access patterns by various parameters to identify the frequency of access, size, future access prediction, replication of files in the cloud storage. This allocates a ranking for each file which also predicts future access pattern. This generates a recommendation dashboard to the user who can decide on the operations such as reorganize, delete or archive the files and eliminate duplicate files in the cloud storage which can increase the space for future use. This system is experimented in the CloudSim environment and validate through multiple simulations testing, by using comparison techniques related to file attributes, delta version-hashing, Data de-duplication. The ranking algorithm technique applied on frequency distribution shows that increase in the storage space upto 10.91% higher than the normal system. It also helps to forecast towards future files usability prediction and prevents the duplicate entries. **[7].**

Encoding web based data for efficient storage in machine learning applications authors are animikh.A , Akshay.K, V. Akhilesh, Chetana.H. With the advent of the information era, we have seen a huge boom in the amount of data produced over the years, which is primarily the result of the Internet and its billions of users worldwide.. There is a need for the processing of the data prior to being applied to various learning algorithms. Deep learning algorithms using neural networks require efficient and proper datasets to yield better results for predictive analysis. As we need to deal with big data over the internet, efficient storage is a challenge. Encoding the dataset in a convenient and efficient form and then storing it is of immense importance. Here, we compare various encoding algorithms to store pre-processed text data. Using Huffman Encoding, the simulation results, for a random sample of 8000 English words, have indicated that the storage space

(memory) requirement dropped to just 0.1% in comparison with the more traditional One-Hot encoding technique. **[8].**

Data storage in web authors are B. gong, chong.G, Q. wang. As the next-generation internet paradigm, Web 3.0 aims to build a free, equal and decentralized Internet. Different from the data monopoly of the Internet oligarchy in Web 2.0, users in Web 3.0 are fairly endowed with the right to store, share, access and manage data. However, as data leakage, tampering and loss may result in a forfeiture of users' data control, the security of data storage has become a key and essential prerequisite to safeguard users' data rights. Blockchain has gained widespread acknowledgment as a prospective technology for solving data security. Nonetheless, its primary suitability for large-scale datasets is limited, owing to the substantial communication and storage overheads. Fortunately, a distributed data storage network (DDSN) can compensate for accommodating large-scale datasets by providing sizeable and inexpensive storage space. In this article, we first summarize the data storage advantages of blockchain and DDSN, and analyze their security challenges in terms of access control, authenticity and confidentiality. Furthermore, to reduce storage costs while protecting data authenticity, we propose a ciphertext split storage model for CP-ABE. In this model, the CP-ABE ciphertext is divided into the attribute component and the data component based on functional structure to realize on-chain/off-chain split storage. In addition, by embedding the ambiguity factor required for decryption and the attributes in the access policy into the index, we design an efficient attribute-based keyword search mechanism to resolve the conflict between confidentiality and availability caused by encryption. Finally, we demonstrate the effective performance of the proposed framework through numerical results. **[9].**

Persistent browser storage extractor authors are F.Aakanksha, A. Kumar, C. Varcol, H. Cho. With the recent development in the World Wide Web, browsers have become more sophisticated, and they can store data locally to provide a faster and user-friendly browsing experience. The locally stored data can be the treasure of information for the digital investigators because using this information can help in solving a crime by discovering any suspect-related information. Browsers can store data in various storage areas such as History, Local Storage, IndexedDB, Cookies, etc. As a digital forensic

investigator, it can become tedious to go to every storage mechanism and look for the stored information for a particular website visit. Therefore, this research is going to provide a mechanism, a Proof of Concept user-friendly GUI-based application, where digital investigators can extract all related data about a particular website visit. This work will incorporate three different browsers, Google Chrome, Opera, and Mozilla Firefox. Not only showing the aggregated view of the extracted data, but digital investigators will also have an option to search data by using a specific domain or event and get all the information related to that domain or event in a unified format. **[10].**

3. PROPOSED SYSTEM

A proposed system for file storage in the cloud aims to address the challenges and leverage the advantages of existing cloud storage solutions. Here's an outline of key components and features that could be incorporated into such a system

Enhanced Security Measures:

End-to-End Encryption: Implement strong encryption protocols to ensure data confidentiality both at rest and in transit. Use techniques like AES-256 encryption and TLS/SSL for secure data transfer.

Access Controls: Implement robust access control mechanisms, including role-based access control (RBAC) and multi-factor authentication (MFA), to enforce granular permissions and prevent unauthorized access.

Data Integrity Assurance: Utilize cryptographic hash functions and digital signatures to verify data integrity and detect any unauthorized modifications to stored files.

Scalability and Performance Optimization:

Elastic Scalability: Design the system to scale dynamically based on workload demands, allowing for automatic provisioning and deprovisioning of storage resources to optimize performance and cost-efficiency.

Content Delivery Networks (CDNs): Integrate with CDNs to optimize data delivery by caching content closer to end-users, reducing latency and enhancing overall performance for global access.

Cost-Effective Storage Options:

Storage Tiers: Offer multiple storage classes (e.g., hot, cool, archive) with varying performance and cost characteristics. Enable automatic tiering based on data access patterns to optimize storage costs without compromising access speed.

4. PROJECT DESCRIPTION

File storage in the cloud offers convenient, scalable, and secure data management solutions. It allows users to store, access, and share files from anywhere with an internet connection. Leading providers ensure high levels of data redundancy and availability across geographically distributed data centers, reducing the risk of data loss. Advanced security measures, including encryption and access controls, protect sensitive information from unauthorized access and cyber threats. Cloud storage services typically offer flexible pricing models, enabling businesses to adjust storage capacity based on demand. Integration with other cloud services enhances productivity and collaboration, making cloud file storage an essential tool for modern digital workflows.

4.1 Problem definition:

The problem of file storage in the cloud revolves around ensuring secure, reliable, and efficient management of data. Challenges include choosing suitable cloud providers that meet specific storage needs while maintaining data integrity and availability. Security concerns, such as unauthorized access and data breaches, must be addressed through robust encryption and access control mechanisms. Scalability and cost-effectiveness are crucial factors, requiring careful planning to accommodate growing storage demands without incurring excessive costs. Integration with existing systems and compliance with regulatory requirements add complexity. Addressing these challenges effectively is essential to deploying a successful and sustainable cloud file storage solution.

4.2 Methodologies:

4.2.1 Module names :

1. User Authentication and Authorization
2. File Upload and Download
3. File Management
4. Data Encryption
5. Access Control
6. Backup and Recovery
7. Scalability and Performance Optimization
8. Integration with Other Services
9. Monitoring and Logging

10. Compliance and Security Audits

4.2.2 Modules explanation:

1. User Authentication and Authorization: Verifies user identity and controls access.
2. File Upload and Download: Manages secure file transfers.
3. File Management: Organizes, renames, and deletes files.
4. Data Encryption: Secures data with encryption methods.
5. Access Control: Manages permissions for file access.
6. Backup and Recovery: Ensures data resilience and restoration.
7. Scalability and Performance Optimization: Enhances system efficiency and responsiveness.
8. Integration with Other Services: Facilitates connectivity with external tools.
9. Monitoring and Logging: Tracks system performance and user activities.
10. Compliance and Security Audits: Ensures adherence to regulations and security standards

4.2.3 Module diagram :

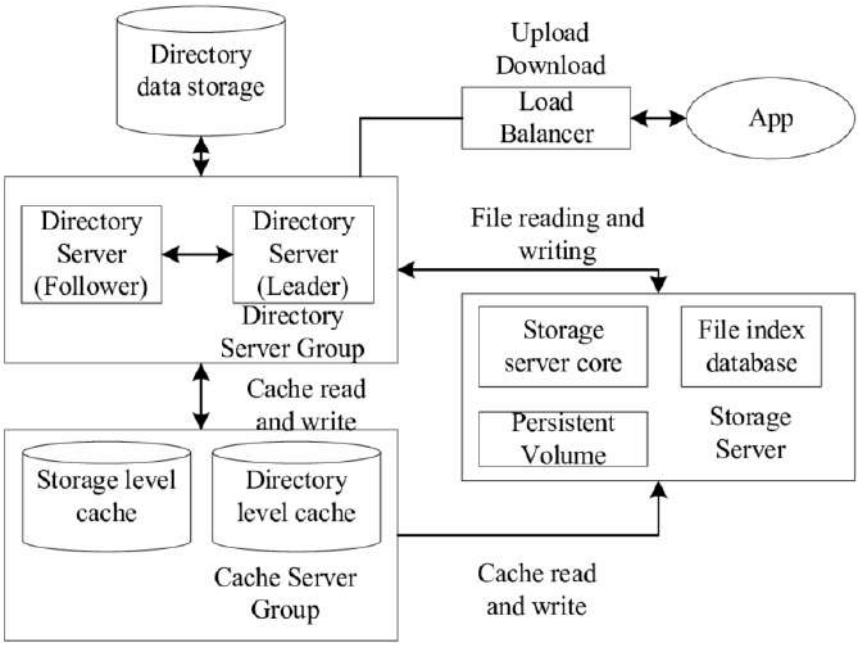


Fig no. 4.2.3. Module diagram

5. MOTIVATION

File storage in the cloud motivates by offering unparalleled flexibility, scalability, and accessibility. It eliminates the constraints of physical storage infrastructure, allowing businesses and individuals to efficiently store, manage, and access data from anywhere with internet connectivity. The cloud's pay-as-you-go model reduces upfront costs, while robust security measures protect against data breaches and ensure compliance with regulatory standards. Seamless integration with other cloud services enhances collaboration and productivity. Moreover, continuous advancements in cloud technology promise ongoing innovation and improvements, making cloud file storage an indispensable solution for modern organizations seeking agility, efficiency, and competitive advantage in today's digital landscape.

5.1 Contribution :

File storage in the cloud contributes significantly to modern computing by revolutionizing data management practices. It enables organizations to scale storage resources dynamically, minimizing operational costs associated with traditional hardware infrastructure. Cloud storage facilitates global accessibility to data, fostering collaboration and remote work capabilities. Enhanced data security measures, including encryption and access controls, ensure robust protection against cyber threats. Furthermore, cloud providers' continuous innovation in storage technologies leads to improved reliability, performance, and scalability. Overall, cloud file storage empowers businesses to streamline operations, innovate faster, and focus resources on core competencies, driving digital transformation and competitiveness in today's interconnected world.

5.2 Technique or algorithm

5.2.1 Existing technique :

Existing techniques in cloud file storage include distributed storage systems like Amazon S3, Google Cloud Storage, and Microsoft Azure Blob Storage. These platforms leverage redundancy across multiple data centers to ensure high availability and durability. They employ data encryption at rest and in transit to secure files, along with robust access control mechanisms to manage permissions effectively. Techniques also include automated backup and recovery solutions, snapshotting for data versioning, and scalable

storage architectures that accommodate varying workloads. Integration with other cloud services enhances functionality, supporting diverse applications from data analytics to machine learning. Overall, these techniques optimize reliability, security, and efficiency in cloud file storage solutions.

5.2.2 Proposed technique :

A proposed technique for advancing cloud file storage involves integrating blockchain technology to enhance data security and integrity. Blockchain's decentralized and immutable ledger capabilities can provide transparent and tamper-proof audit trails for file access and modifications. Smart contracts can automate and enforce access control policies, ensuring only authorized parties can interact with specific files. Additionally, leveraging edge computing for storage and processing closer to users can improve latency and responsiveness. Implementing hybrid storage solutions combining public cloud with on-premises infrastructure offers flexibility and data sovereignty benefits. Continuous innovation in encryption methods and data deduplication techniques further optimize storage efficiency and reduce costs, fostering a more resilient and scalable cloud file storage ecosystem.

6. REQUIREMENTS ENGINEERING

The interpretation of the handwriting character by developing techniques and methods such as improvement of character classification techniques. The accurate and rapid classification for accurate information retrieval, sound classification, stock price forecasting.

6.1 Hardware requirements:

The hardware are requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system.they are used by software engineers as the starting point for the system design. It shouls what the system do and not how it should beimplemented.

PROCESSOR : INTEL(3,5,7,9)/RYZEN(5,7,9)

RAM : 4 GB RAM

HARD DISK : 250 GB

6.2 Software requirements :

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

OPERATING SYSTEM : WINDOWS 7/8/10/11

PLATFORM : GOS

PROGRAMMING LANGUAGE : WEB DEVELOPMENT

6.3 Functional requirements :

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, In this system we are implementing effective kidney disease prediction system using decision tree support vector machine, random forest algorithm. We can give the input as in CSV file or manual entry to the system.

6.4 Non-Functional requirements:

The major non-functional requirements of the system are as follows

Usability

The system is designed with completely automated process hence there is no or less user intervention.

Reliability

The system is more reliable because of the qualities that are inherited from the chosen platform java.The code is built by using java is more reliable .

performance

This system is developing in the high level language and using the advanced frontend and back end technologies it will give response to the end user on client system with in very less time.

supportability

The system is designed to be the cross platform supportable. The system is supported on a widerange of hardware and any software platform, which is having JVM, built into the system.

Implementation

The system is implemented in web environment using struts framework. The apache tomcat isused as the web server and windows xp professional is used as the platform. Interface the userinterface is based on Struts provides HTML

7. DESIGN ENGINEERING

Design engineering for cloud file storage involves integrating robust architectures and methodologies to optimize reliability, scalability, and security. It begins with assessing storage requirements and selecting appropriate cloud providers based on performance and compliance needs. Architectural design focuses on fault-tolerant distributed systems, ensuring data redundancy and high availability across geographically dispersed servers. Security design incorporates encryption, access controls, and compliance with regulatory standards to protect sensitive data. Continuous monitoring and performance tuning enhance operational efficiency. Collaboration with cross-functional teams ensures alignment with business objectives and seamless integration with existing IT infrastructure, enabling agile and resilient cloud file storage solutions.

7.1 UML Diagram:

7.1.1 Use case diagram:

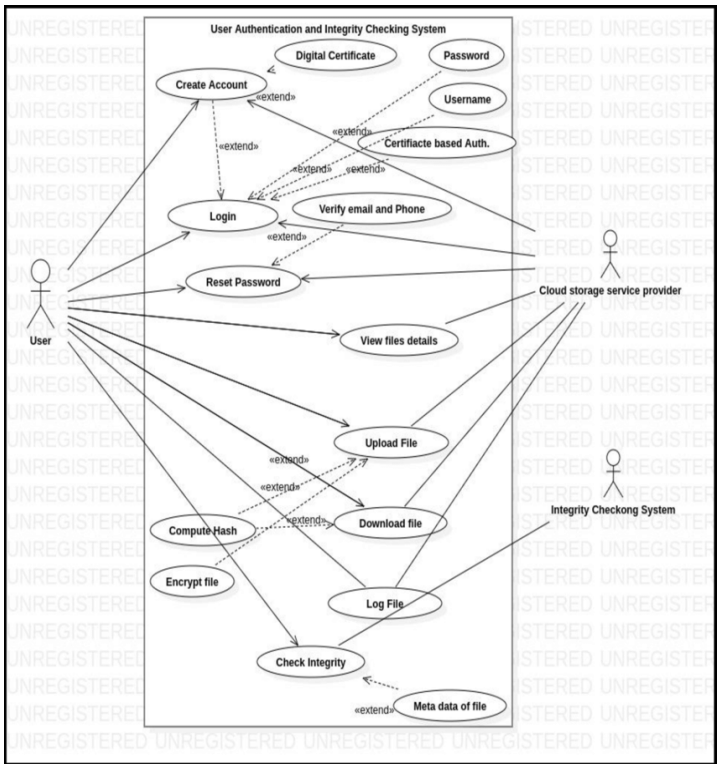


Fig no.7.1.1.Use case diagram

Explanation:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is topresent 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

7.1.2 Class diagram:

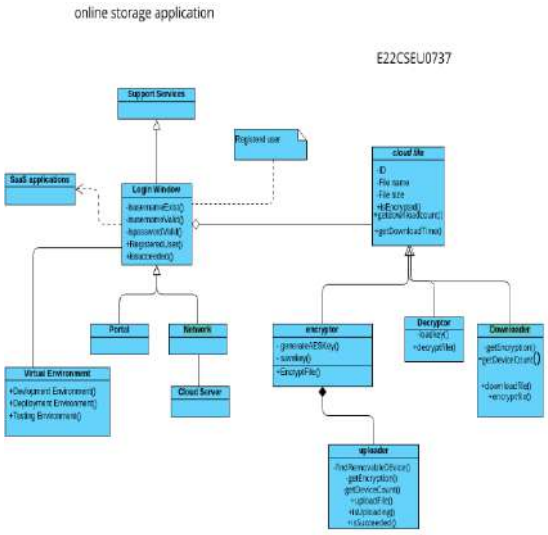


Fig no..7.1..2.Class diagram

Explanation:

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project

7.1.3.Objective diagram:

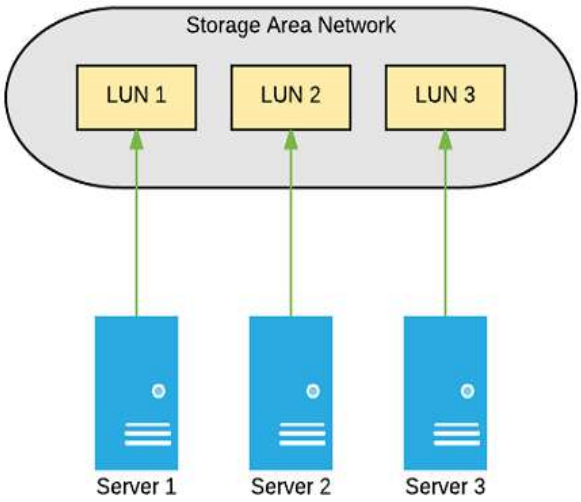


Fig no..7.1.3.Objective diagram

Explanation:

In the above diagram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

7.1.4.Component diagram:

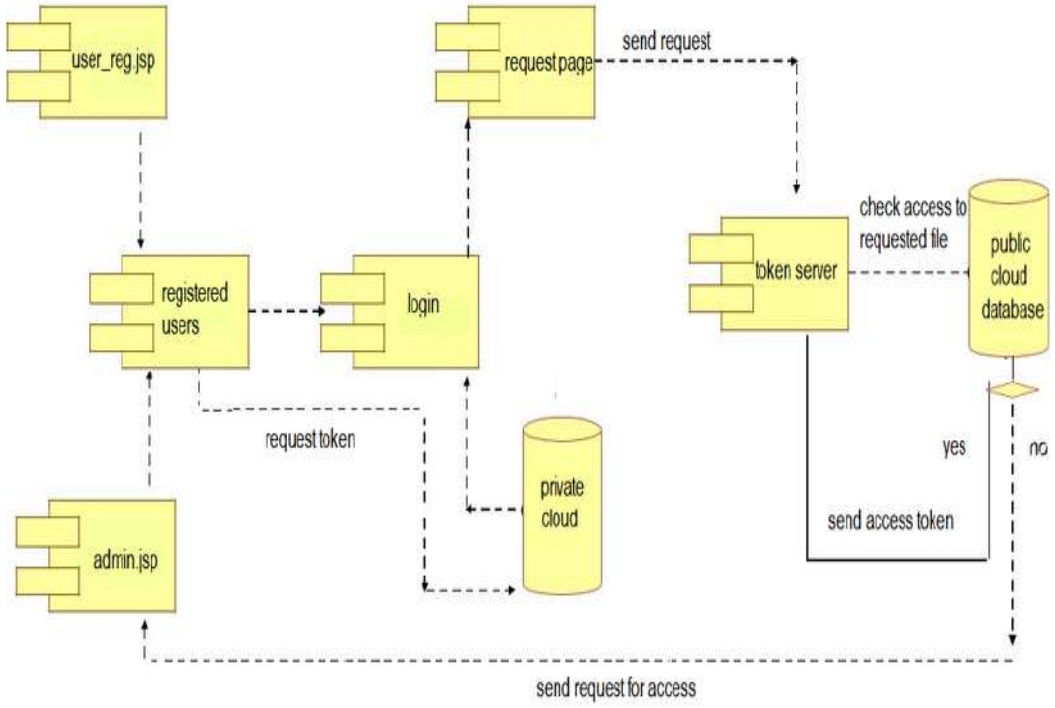


Fig no. 7.1.4 Component diagram

Explanation:

In the Unified Modeling Language, a component diagram depicts how componentsare wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

7.1.5.Deployment diagram :

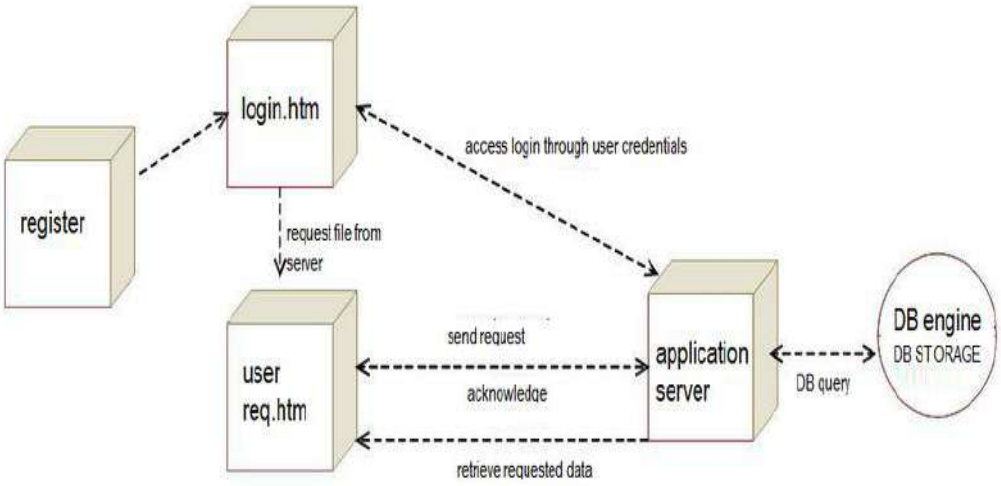


Fig no.7.1.5.Deployment diagram

Explanation:

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

7.1.6. Sequence diagram:

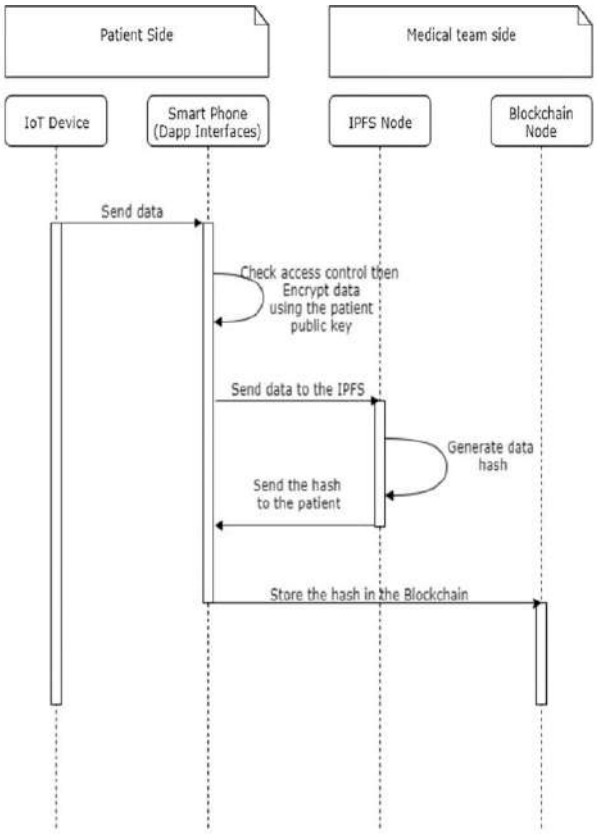


Fig no..7.1.6 Sequence diagram

Explanation:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved inthe scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

7.1.7.Collabaration diagram:

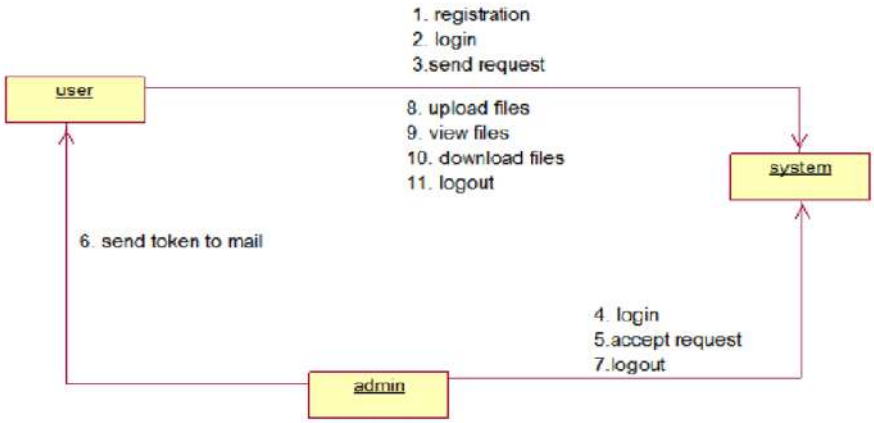


Fig no..7.1.7g. collaboration diagram

Explanation:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old althoughit has been refined as modeling paradigms have evolved

7.1.8.State diagram:

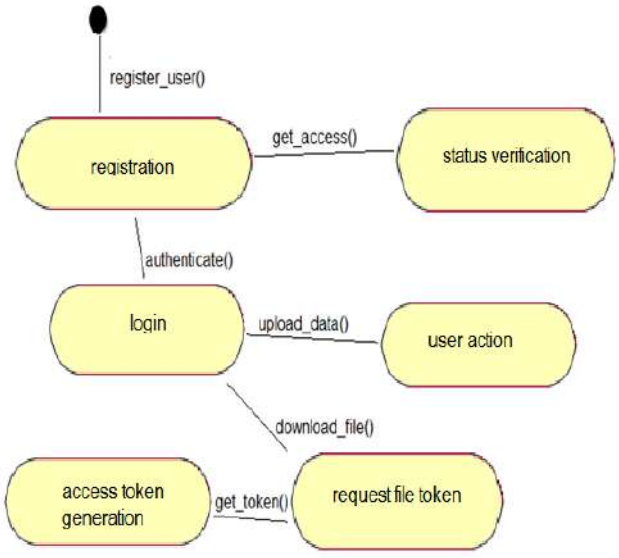


Fig no. 7.1.8. State diagram

Explanation:

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. UML activity diagrams could potentially model the internal logic of a complex operation. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structural development.

7.1.9. Activity diagram:

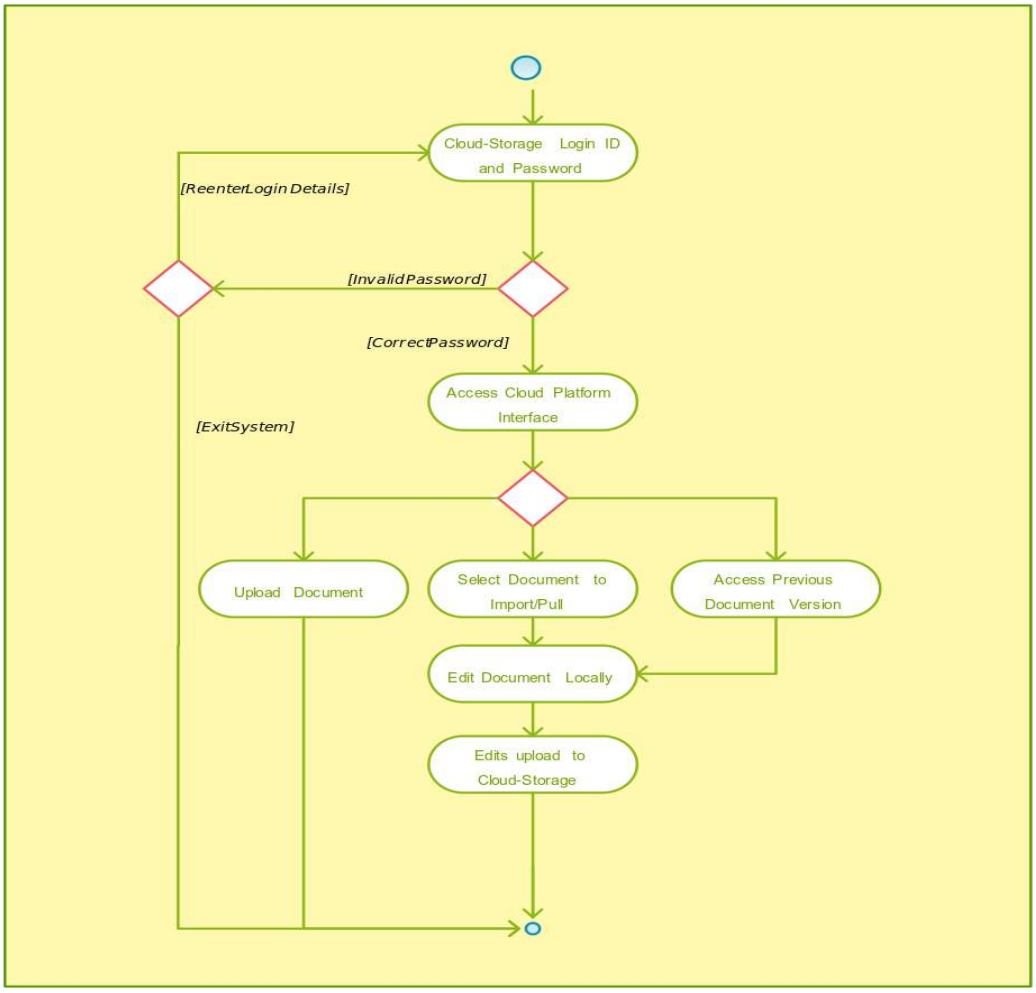


Fig no.7.1.9. activity diagram

Explanation:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system.

7.2 Data flow diagram:

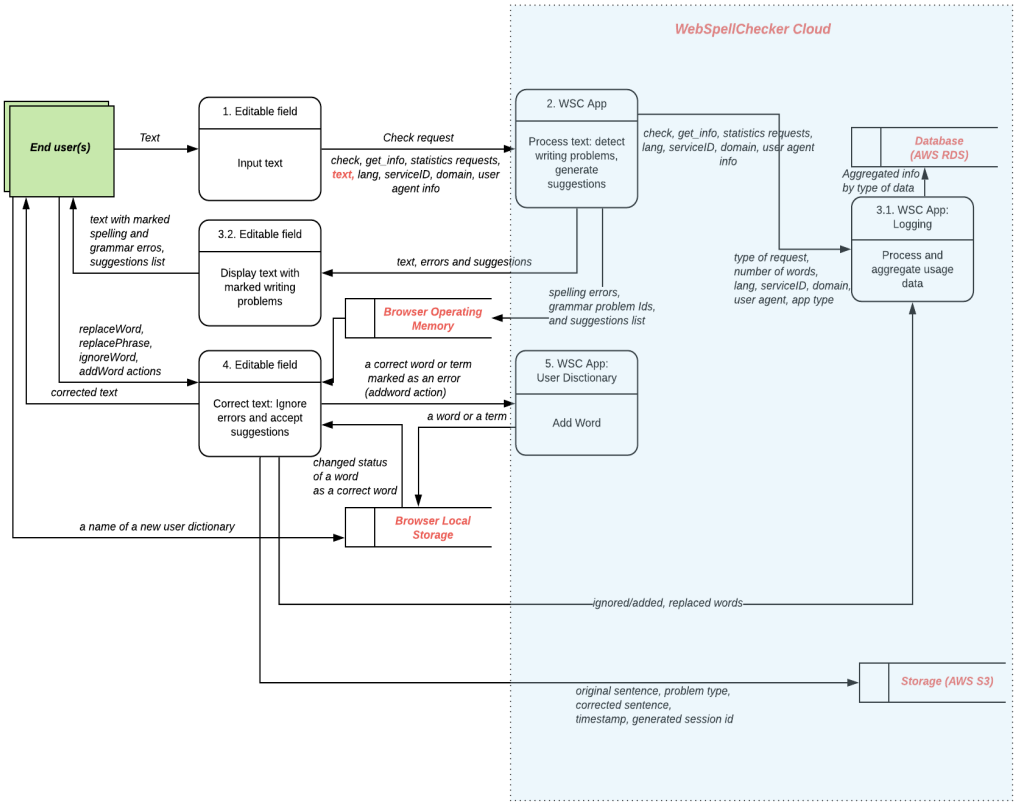


Fig no..7.2.Data flow diagram

Explanation:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).A DFD shows whatkinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel

7.3 System architecture:

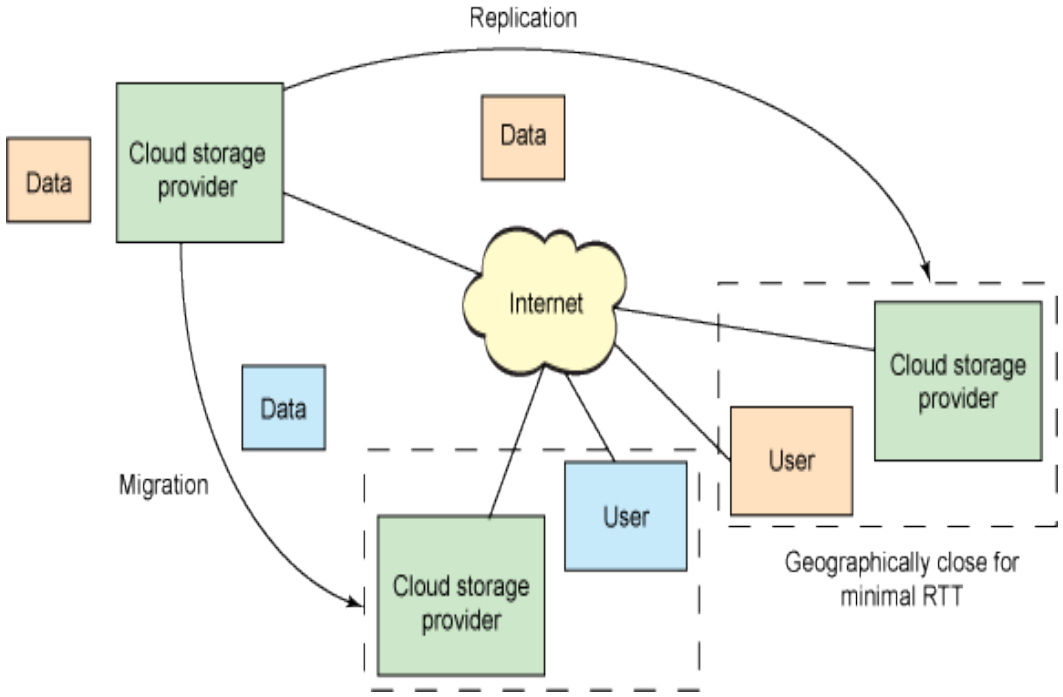


Fig no.7.3.System architecture

Explanation:

This diagram shows how data is managed between users and cloud storage providers over the Internet. Data can be uploaded, stored, and accessed from different cloud storage providers. Users connect to the cloud through the Internet to store or retrieve their data. Cloud storage providers can duplicate data to improve reliability and availability. Data migration between providers ensures better performance or backup. Some users connect to geographically closer providers for minimal latency .

8. SOFTWARE SPECIFICATION

8.1 Development tools:

HTML :

HTML (Hypertext Markup Language) structures web content using tags like `<html>`, `<head>`, `<body>`, and semantic elements like `<header>`, `<article>`, and `<footer>`. It defines text formatting, links (`<a>`), images (`<img>`), forms (`<form>`), and supports accessibility. HTML integrates with CSS for styling and JavaScript for interactivity, essential for modern web development.

History of HTML :

HTML, or Hypertext Markup Language, was conceived by Tim Berners-Lee at CERN in the late 1980s to share documents over the internet. The first standard, HTML 2.0, was formalized in 1995. HTML 3.2 (1997) introduced tables and forms. HTML 4.01 (1999) added style sheets and scripting support. XHTML in the early 2000s aimed for stricter syntax. HTML5, finalized in 2014, revolutionized web development with semantic elements, multimedia support, canvas for graphics, and APIs for interactivity and offline applications. It's continually updated by the W3C and WHATWG, marking HTML's evolution from a simple markup language to a robust framework powering the modern web.

CSS CODE :

CSS (Cascading Style Sheets) is used to style HTML elements, defining their appearance, layout, and presentation on web pages. It uses selectors to target elements and properties like color, font-size, margin, and more to control design aspects. CSS improves accessibility, user experience, and consistency in web design.

History of CSS :

CSS (Cascading Style Sheets) emerged in the late 1990s as a solution to separate content from presentation on the web. It aimed to enhance HTML's limited styling capabilities. CSS1 was standardized in 1996, followed by CSS2 in 1998, introducing advanced layout techniques and media queries. CSS3, started in 1999 and still evolving, expanded with modules for animations, gradients, and responsive design. Its adoption transformed web design, offering flexibility, consistency, and improved user experience.

Java Script:

JavaScript is a high-level, interpreted programming language primarily used to make web pages interactive and dynamic. It enables features like form validation, animations,

interactive maps, and real-time content updates without reloading the page. Running directly in the browser, JavaScript works alongside HTML and CSS to build fully functional and responsive websites. With the introduction of Node.js, JavaScript also became a powerful language for server-side development, allowing developers to use it across the full web development stack.

History of Java Script:

JavaScript was created in 1995 by Brendan Eich while working at Netscape, initially called Mocha, later renamed to Live Script, and finally JavaScript to align with the popularity of Java. To standardize the language, ECMAScript was introduced in 1997. Over the years, JavaScript evolved significantly, especially with the ES6 (ECMAScript 2015) update, which brought modern features like classes, modules, and arrow functions. Since then, JavaScript has continued to receive annual updates, making it one of the most widely used and influential programming languages in the world today.

9. IMPLEMENTATION

Complete source code:

// contactus.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Contact Us</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <div class="container">
      <div>
<h2>Contact Us</h2>
      </div>
      <div class="nav-menu">
         <button type="submit" class="btn" id="about" name="about" onclick="about()">
ABOUT</button>
        <button type="submit" class="btn" id="login" name="login"
onclick="login()">LOGIN </button>
         <button type="submit" class="btn" id="signup" name="signup"
onclick="register()"> SIGNUP</button>
         <button type="submit" class="btn" id="index" name="index" onclick="home()">
HOME</button>
      </div>
   </div>
   <div class="container1">
      <form class="contact-form">

         <div>
            <h2>Contact us</h2>
         </div>
```

```
        <div class="ig">

<label for="Name" >Name:</label>
            <input type="text" id="Name" >
        </div>
        <div class="ig">
            <label for="email" >Email:</label>
            <input type="text" id="email" >
        </div>
        <div class="ig">
            <label for="subject">subject:</label>
            <input type="text" id="subject" >
        </div>
        <div class="ig">
            <label for="message" >Message:</label>
      <textarea name="message" id="message" rows="4"></textarea>
        </div>
        <div>
            <button type="submit" id="submit" name="submit" >submit</button>
        </div>
      </form>
        </div>
        <script>
            function about(){
                window.location.href = "about.html";
            }
            function login(){
                window.location.href = "login.html";
            }
            function register(){
                window.location.href = "register.html";
            }
            function home(){
                window.location.href = "index.html";
            }
```

```
        </script>

<script>
            document.addEventListener("DOMContentLoaded", function() {
   document.querySelector(".contact-form").addEventListener("submit", function(event)
{
      event.preventDefault(); // Prevent the default form submission
      // Get form values
      let name = document.getElementById("name").value.trim();
      let email = document.getElementById("email").value.trim();
      let subject = document.getElementById("subject").value.trim();
      let message = document.getElementById("message").value.trim();
      // Simple validation
      if (name === "" || email === "" || subject === "" || message === "") {
         alert("Please fill in all fields.");
         return;
      }
      // Show thank you message
      alert("Thank you for contacting us, " + name + "!");
      // Redirect to index.html after a short delay
      setTimeout(function() {
         window.location.href = "index.html";
      }, 1000);
   });
});
        </script>
        <script>
            document.addEventListener("DOMContentLoaded", function() {
   // Add animation to contact form
   let contactForm = document.querySelector(".contact-form");
   contactForm.style.opacity = "0";
   contactForm.style.transform = "translateY(50px)";
   contactForm.style.transition = "opacity 0.5s ease-out, transform 0.5s ease-out";
   setTimeout(() => {
      contactForm.style.opacity = "1";
```

```
        contactForm.style.transform = "translateY(0)";

    }, 100);
});
            </script>
</body>
</html>
```

// feedback.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Us</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container1">
        <form class="contact-form">
            <div>
                <h2>Feedback</h2>
            </div>
            <div class="ig">
                <label for="name" >Name:</label>
                <input type="text" id="name" >
            </div>
            <div class="ig">
                <label for="email" >Email:</label>
                <input type="text" id="email" >
            </div>
            <div class="ig">
                <label for="feedback">Feedback:</label>
        <textarea name="feedback" id="feedback" rows="4"></textarea>
            </div>
            <div>
```

```
            <button type="submit" id="submit" name="submit" >submit</button>

  </div>
      </form>
         </div>
         <script>
            document.addEventListener("DOMContentLoaded", function() {
   document.querySelector(".contact-form").addEventListener("submit", function(event)
{
      event.preventDefault(); // Prevent the default form submission
      // Get form values
      let name = document.getElementById("name").value.trim();
      let email = document.getElementById("email").value.trim();
      let feedback = document.getElementById("feedback").value.trim();
      // Simple validation
      if (name === "" || email === "" ||  feedback === "") {
         alert("Please fill in all fields.");
         return;
      }
      // Show thank you message
      alert("Thank you for giving a feedback, " + name + "!");
      // Redirect to index.html after a short delay
      setTimeout(function() {
         window.location.href = "home.html";
      }, 1000);
   });
});
         </script>
         <script>
            document.addEventListener("DOMContentLoaded", function() {
   // Add animation to contact form
   let contactForm = document.querySelector(".contact-form");
   contactForm.style.opacity = "0";
   contactForm.style.transform = "translateY(50px)";
   contactForm.style.transition = "opacity 0.5s ease-out, transform 0.5s ease-out";
```

```
    setTimeout(() => {

   contactForm.style.opacity = "1";
       contactForm.style.transform = "translateY(0)";
    }, 100);
});
          </script>
</body>
</html>
```

// guest.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Guest</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <div>
<h2>Login As a Guest</h2>
        </div>
        <div class="nav-menu">
        <button type="submit" class="btn" id="about" name="about" onclick="about()">
ABOUT</button>
            <button type="submit" class="btn" id="signup" name="signup"
onclick="register()"> SIGNUP</button>
            <button type="submit" class="btn" id="index" name="index" onclick="home()">
HOME</button>
        </div>
    </div>
    <div class="con2">
        <div class="container2">
            <h2>Login As a Guest</h2>
```

```
        <form>

 <div class="ig">
            <label for="name" >Name:</label>
            <input type="text" name="name" id="name">
          </div>
          <div class="ig">
           <label for="email" >E-mail:</label>
           <input type="email" name="email" id="email">
         </div>
         <div>
           <button type="submit" >LogIn As Guest</button>
         </div>
         <b>If you want full access then,</b><a href="register.html">Register here</a>
        </form>
     </div>
   </div>
   <script>
    function about(){
       window.location.href = "about.html";
    }
    function register(){
       window.location.href = "register.html";
    }
    function home(){
       window.location.href = "index.html";
    }
</script>
   <script>
    document.addEventListener("DOMContentLoaded", function() {
    document.querySelector("form").addEventListener("submit", function(event) {
       event.preventDefault(); // Prevent form submission
       let name = document.getElementById("name").value.trim();
       let email = document.getElementById("email").value.trim();
       if (name === "" || email === "") {
```

```
            alert("Please fill in all fields.");

 return;
        }
        // Store session details in localStorage (for temporary tracking)
        localStorage.setItem("guestName", name);
        localStorage.setItem("guestEmail", email);
        localStorage.setItem("fileUploaded", "false"); // Track upload limit
        localStorage.setItem("fileViewed", "false"); // Track view limit
        alert("Welcome, " + name + "! You are logged in as a guest.");
        // Redirect to upload.html
        window.location.href = "manage.html";
    });
});
  </script>
  <script>
    document.addEventListener("DOMContentLoaded", function() {
    // Add animation to guest form
    let guestForm = document.querySelector(".container2");
    guestForm.style.opacity = "0";
    guestForm.style.transform = "translateY(50px)";
    guestForm.style.transition = "opacity 0.5s ease-out, transform 0.5s ease-out";
    setTimeout(() => {
        guestForm.style.opacity = "1";
        guestForm.style.transform = "translateY(0)";
    }, 100);
});
  </script>
</body>
</html>
```

// manage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>File Manager</title>
    <link rel="stylesheet" href="style.css">
    <script src="https://ucarecdn.com/libs/widget/3.x/uploadcare.full.min.js"></script>
    <style>
        body{
            height: 1000vh;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>File Manager</h1>
        <div class="nav-menu">
            <button class="btn" onclick="home()">Home</button>
            <button class="btn" onclick="profile()">Profile</button>
        </div>
    </div>
    <div class="con1">
        <div class="ufile">
            <main>
                <h2>Upload a File</h2>
                <input type="hidden" id="uploadcare-widget" role="uploadcare-uploader"
data-public-key="27c4bc0538338ed619c9">
                <button onclick="uploadFile()">Upload</button>
            </main>
        </div>
        <div class="mfile">
            <h2>File manager</h2>
            <main>
            <h2>Stored Files</h2>
            <ul id="fileList"></ul>
            <div id="fileContainer" class="file-container"></div>
        </main>
```

```
        </div>

</div>
    <script>
        function home() {
            window.location.href = "home.html";      }
        function profile() {
            window.location.href = "profile.html";    }
        let widget = uploadcare.Widget("#uploadcare-widget");
        widget.onChange(function(file) {
            if (file) {
                file.done(function(info) {
                    let fileName = info.name;
                    let fileUrl = info.cdnUrl;
                    localStorage.setItem(fileName, fileUrl);
                    alert("File uploaded successfully!");
                    loadFiles();   });   }  });
        function loadFiles() {
            let fileList = document.getElementById("fileList");
            fileList.innerHTML = "";
            for (let i = 0; i < localStorage.length; i++) {
                let fileName = localStorage.key(i);
                let fileUrl = localStorage.getItem(fileName);
                if (!fileUrl) continue;
                let li = document.createElement("li");
                li.innerHTML = `<strong>${fileName}</strong>`;
                let viewBtn = document.createElement("button");
                viewBtn.textContent = "View";
                viewBtn.onclick = function () { viewFile(fileName, fileUrl); };
                let deleteBtn = document.createElement("button");
                deleteBtn.textContent = "Delete";
                deleteBtn.onclick = function () {
                    localStorage.removeItem(fileName);
                    loadFiles();
                    document.getElementById("fileContainer").innerHTML = "";   };
```

```
            let downloadBtn = document.createElement("button");

downloadBtn.textContent = "Download";
            downloadBtn.onclick = function () {
                let link = document.createElement("a");
                link.href = fileUrl;
                link.download = fileName;
                document.body.appendChild(link);
                link.click();
                document.body.removeChild(link);};
            li.appendChild(viewBtn);
            li.appendChild(deleteBtn);
            li.appendChild(downloadBtn);
            fileList.appendChild(li);     }}
        function viewFile(fileName, fileUrl) {
            let fileContainer = document.getElementById("fileContainer");
            fileContainer.innerHTML = `<h2>Preview: ${fileName}</h2>`;
            let fileType = fileName.split('.').pop().toLowerCase();
            let element;
            if (["png", "jpg", "jpeg", "gif", "bmp"].includes(fileType)) {
                element = document.createElement("img");
                element.src = fileUrl;
            } else if (["mp4", "webm", "ogg"].includes(fileType)) {
                element = document.createElement("video");
                element.controls = true;
                element.src = fileUrl;
            } else if (["mp3", "wav", "ogg"].includes(fileType)) {
                element = document.createElement("audio");
                element.controls = true;
                element.src = fileUrl;
            } else if (fileType === "pdf") {
                element = document.createElement("iframe");
                element.style.width = "100%";
                element.style.height = "500px";
                element.src = fileUrl;
```

```
            } else {

  element = document.createElement("a");
                element.href = fileUrl;
                element.textContent = "Download File";
                element.download = fileName;   }
              fileContainer.appendChild(element);
          }
          window.onload = loadFiles;
      </script>
</body>
</html>
```

10.RESULTS AND OUTPUTS

Here are some of the snapshots of our output screen of FILE STORAGE IN CLOUD .

10.1 Various snapshots of derived outputs:

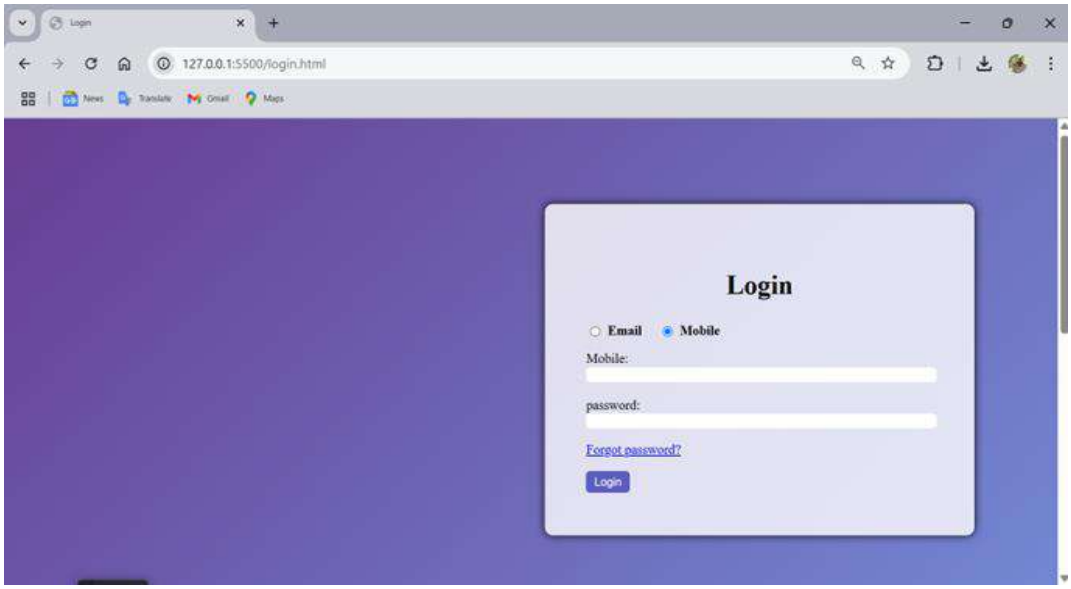


Fig no.10.1.1. login through mobile

Explanation:

Existing user can login by mobile number through this page by entering details like mobile number and password

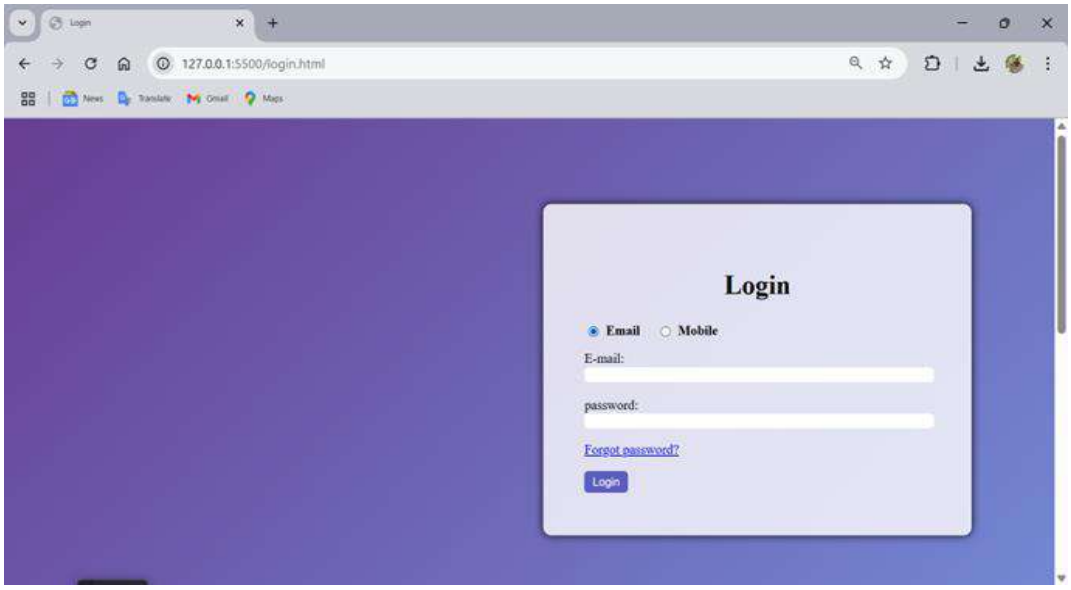


Fig no. 10.1.2. login through email id

Explanation:

Existing user can login by E-mail ID through this page by entering details like E-mail

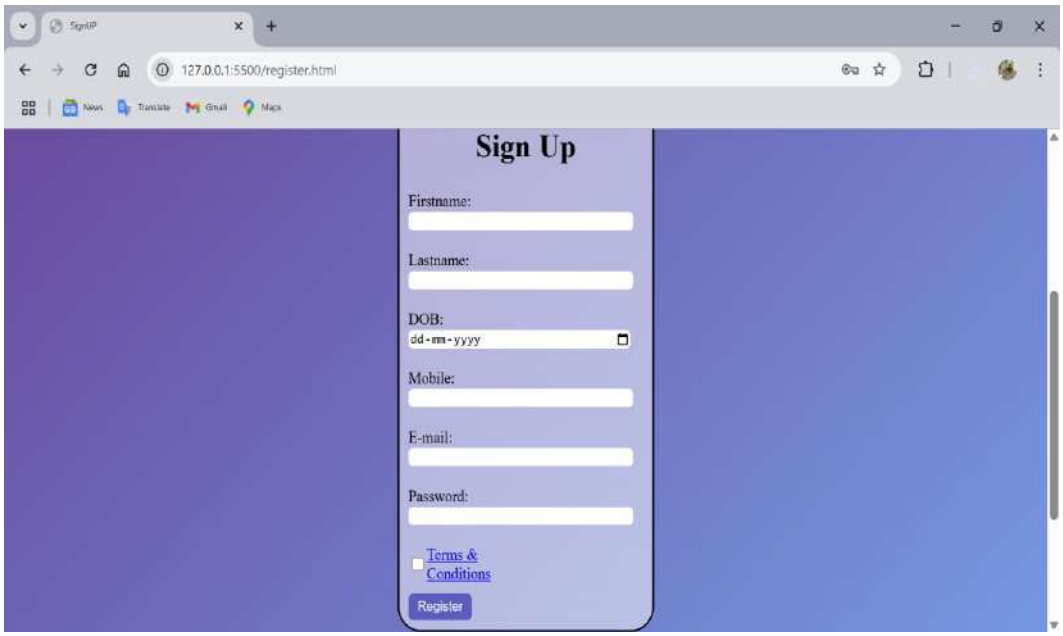


Fig no.10.1.3. register page

Explanation:

In this new users can register to access file storage in cloud by registering their first and last name and email id and password and accepting terms and conditions new user can get registered

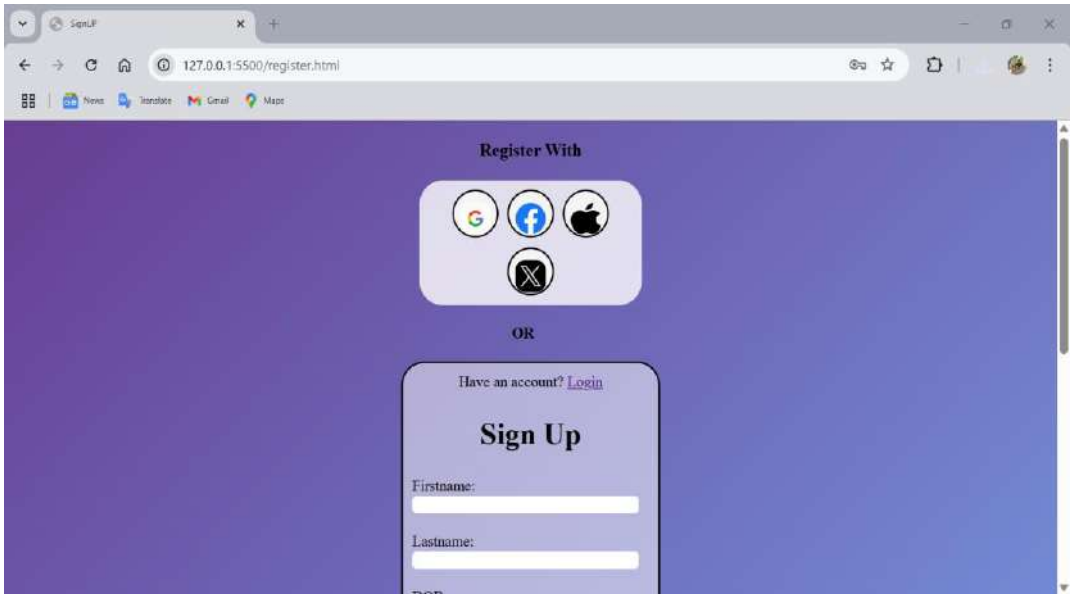


Fig no.10.1.4. register through social accounts

Explanation:

In this new users can register to access file storage in cloud by registering their social accounts like Facebook , Instagram , Apple id etc.

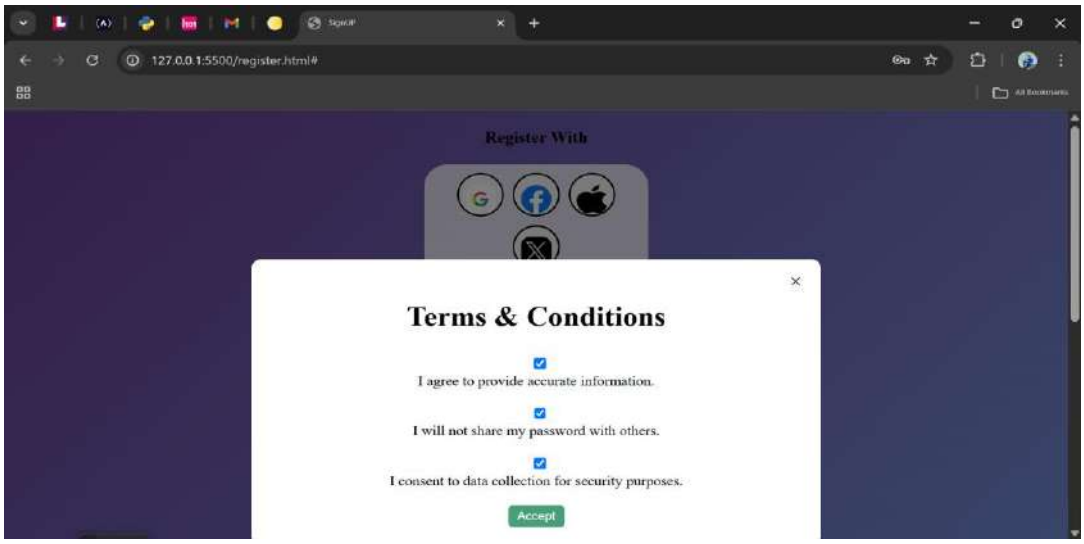


Fig no. 10.1.5. terms and conditions

Explanation:

In this the user has to agree to the terms and conditions for further access.

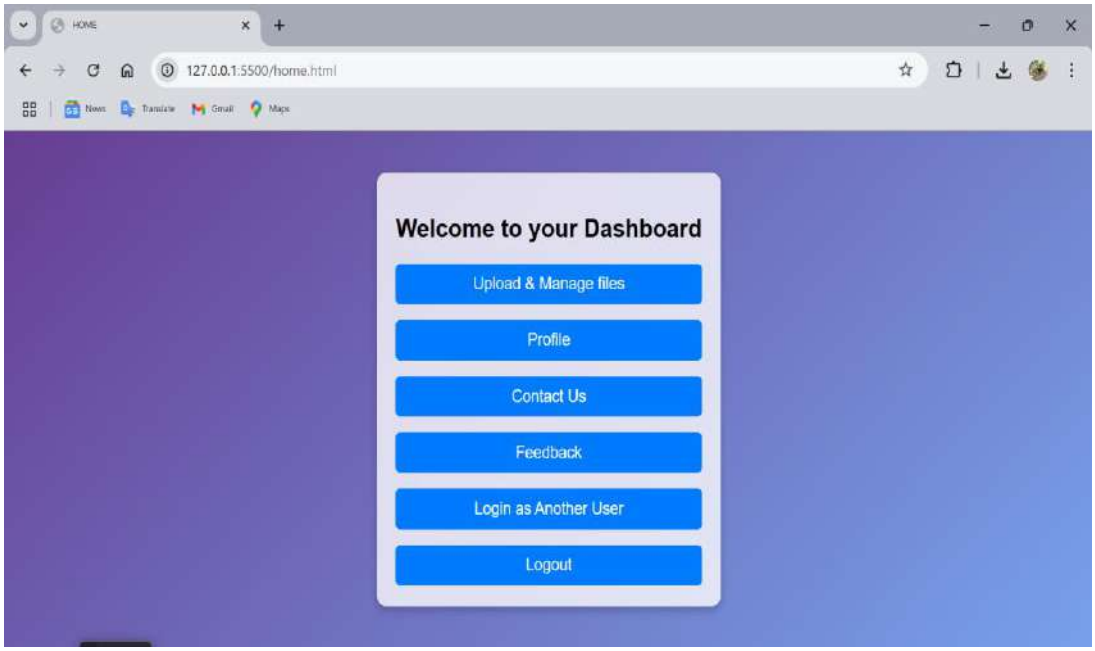


Fig no.10.1.6. home page

Explanation:

After user logins or registers it is the main site which they see firstly and here user can see theit files and can download them also

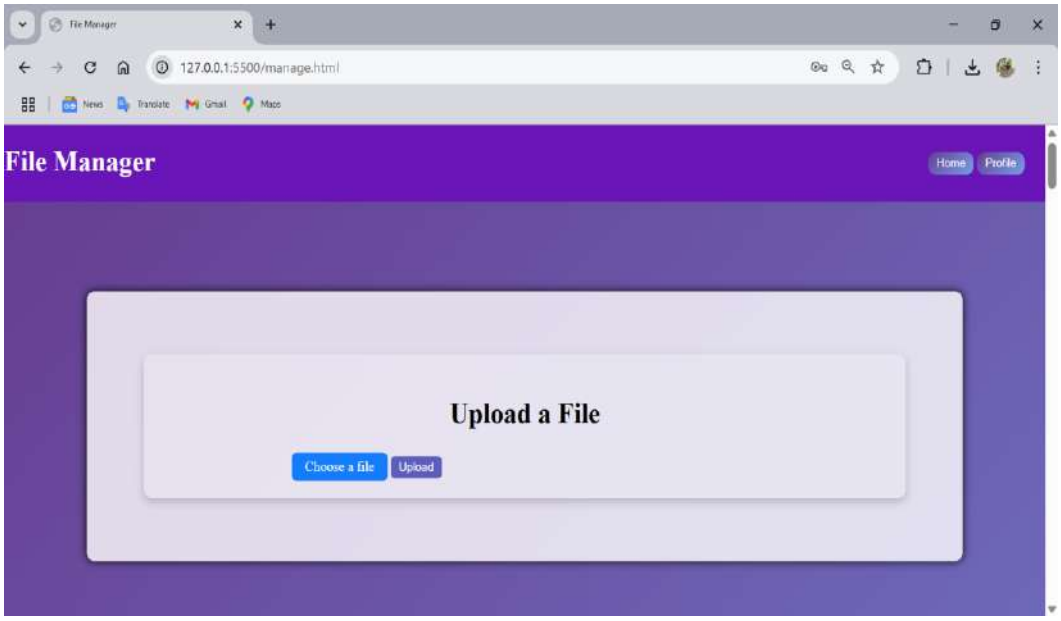


Fig no. 10.1.7. upload page

Explanation:

In this page user can upload the required file of any format,by choosing file by clicking choose file button and selecting the file and they can upload that file by clicking upload button

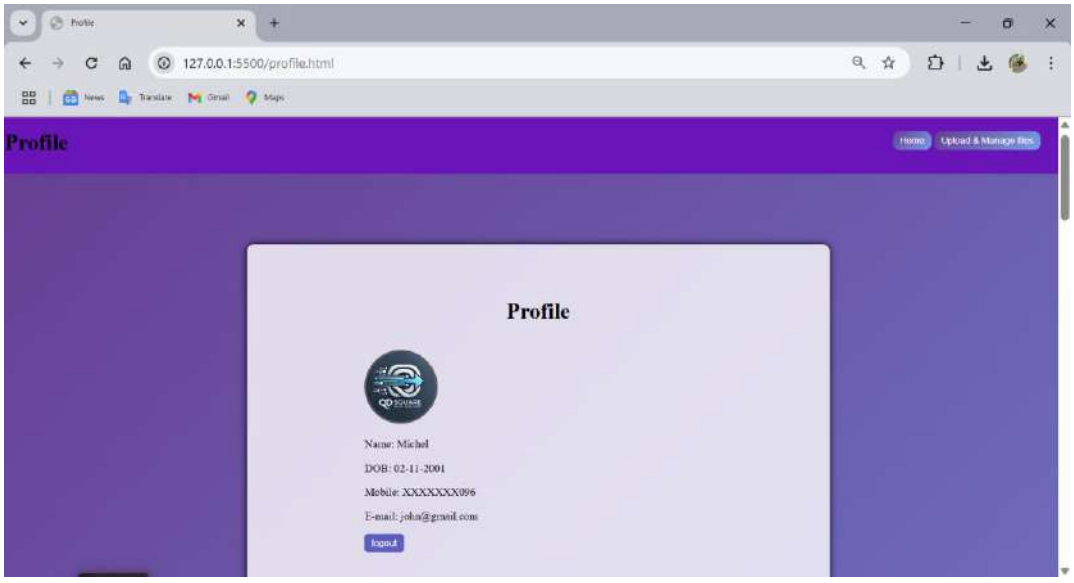


Fig no. 10.1.8. profile page

Explanation:

In this page user can see their details by which details they have logined or registered

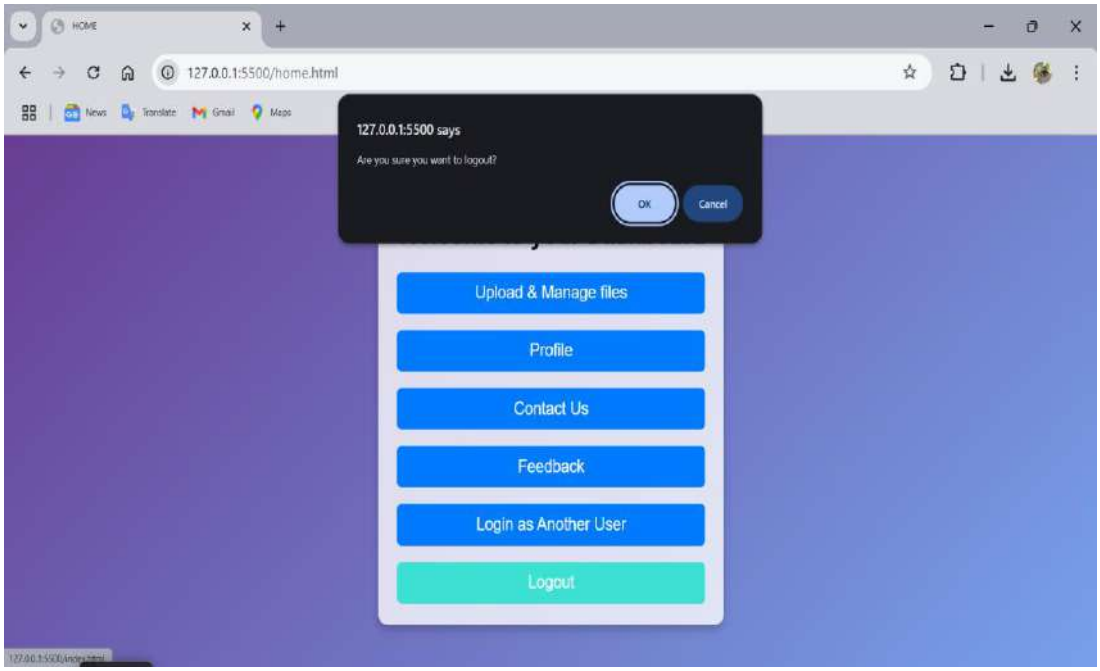


Fig no.10.1.9. logout

Explanation:

User can exit the site by clicking logout button

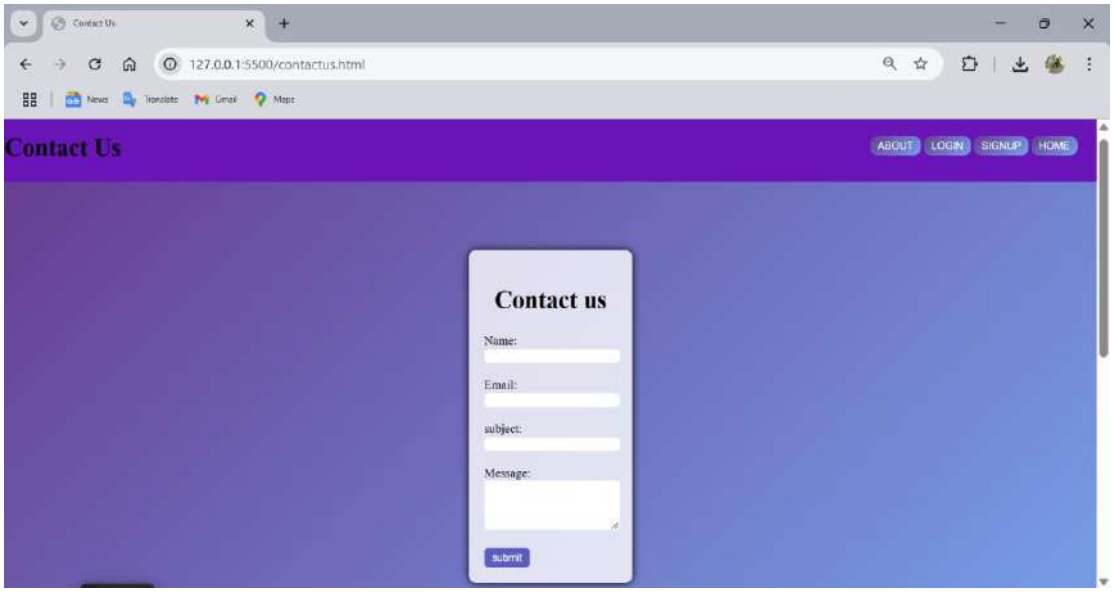


Fig no.10.1.10. contact us page

Explanation:

Our Contact page provides an easy way to reach us for inquiries, support, or feedback. Whether you have questions, need assistance, or want to connect, we're here to help. Fill out the form, email us, or call we'll respond promptly to ensure your needs are met.

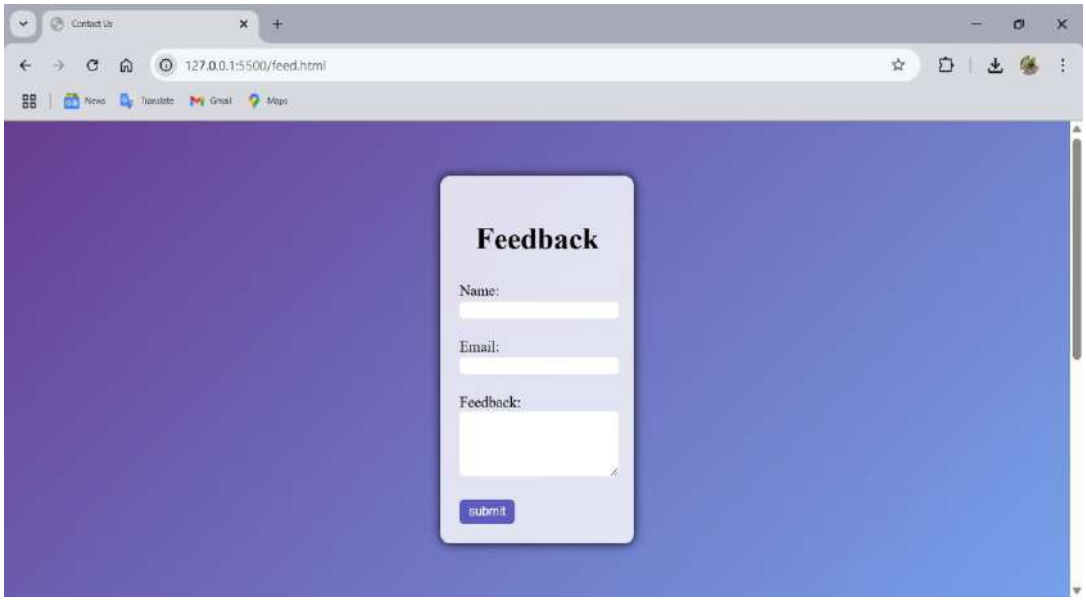


Fig no.10.1.11. feed back page

Explanation:

User can use this Feedback page to share their experience, ideas, or concerns. Their input helps us improve and serve them better

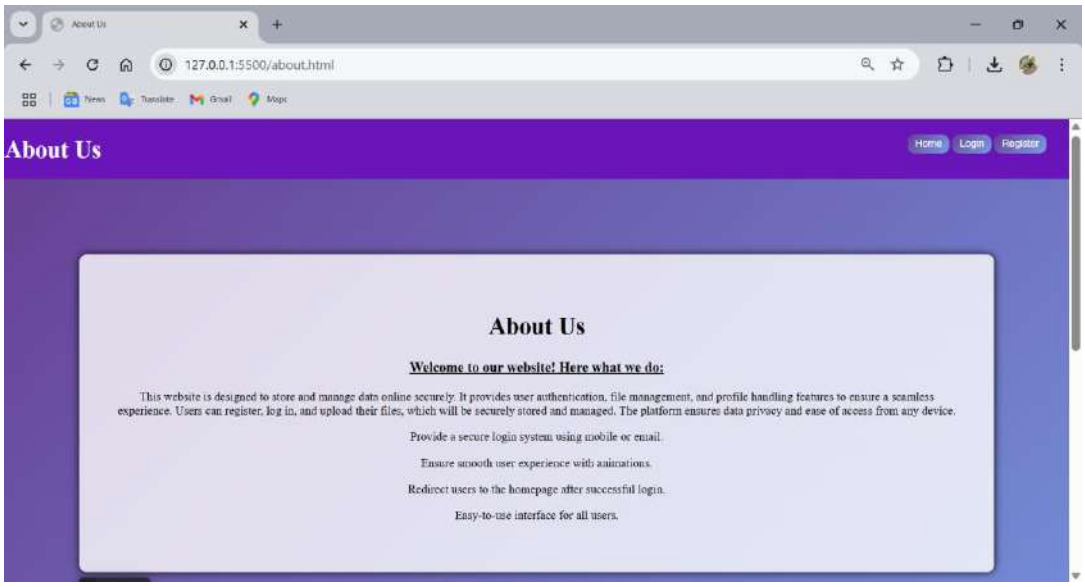


Fig no.10.1.12. about us page

Explanation:

Learn more about who we are, what we do, and why we do it. Our About page shares our mission, values, and the story behind our journey.

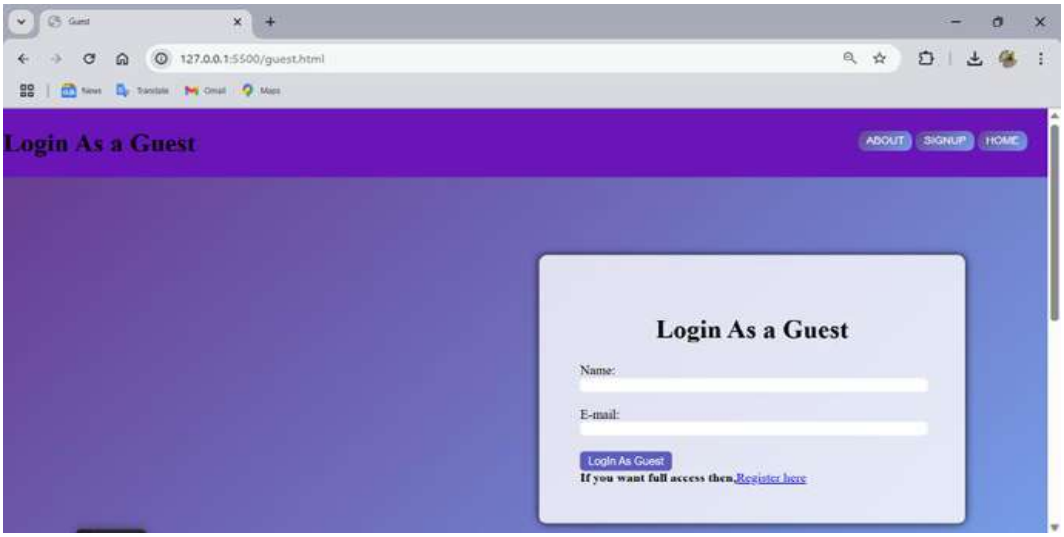


Fig no.10.1.13. guest login

Explanation:

User can experience the website as a guest how it will works and to know about the performance of the website

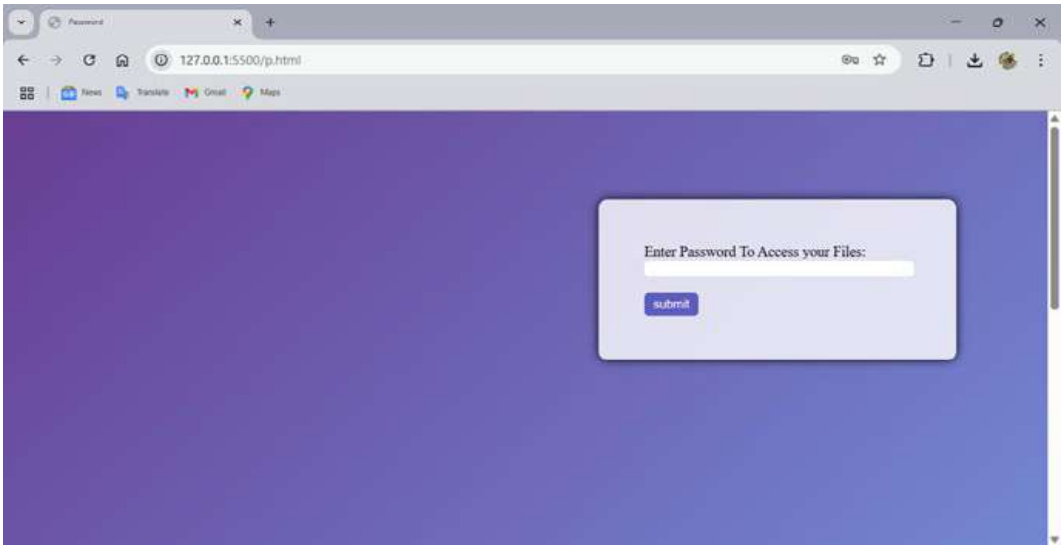


Fig no.10.1.14. password to access files

Explanation:

User have give the password to access the stored files, if the password is valid then only the user can redirect to the file manager page

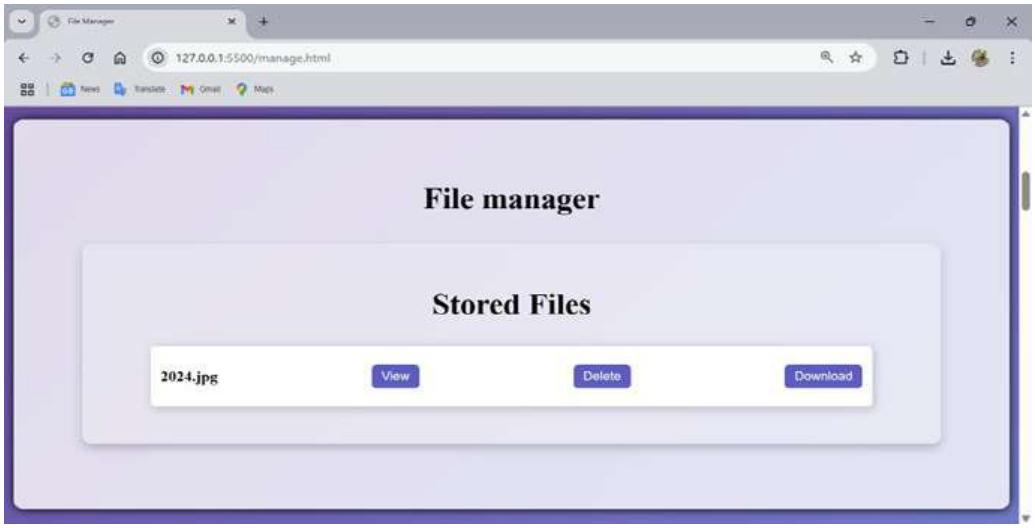


Fig no.10.1.15. file manager page

Explanation:

In this file manager page it shows all the files which are stored by the user and user can can view, download and delete the files by clicking on the buttons.

11.SOFTWARE TESTING

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

11.1 Developing methodologies:

Developing methodologies for testing a food delivery application involves first understandingthe application's requirements thoroughly. Next, create a comprehensive test plan that outlines the testing objectives, scope, resources, and timelines. Design detailed test cases covering all functionalities, focusing on order placement, payment processing, order tracking, user interface, performance under load, and security vulnerabilities. Set up a test environment that replicates real-world conditions to ensure accurate testing across different devices, operating systems, and network environments. Execute the test cases systematically, documenting and reporting any issues encountered. Finally, review and refine the testing methodologies based on feedback and continuous improvement to ensure the application meets high standards of functionality, usability, security, and performance.

11.2 Types of testing:

11.2.1 Unit testing:

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

11.2.2 Functional testing:

Functional testing for a food delivery application involves validating its features and functionalities to ensure they align with the specified requirements. This testing focuses on assessing the application's ability to perform tasks such as order placement, payment processing, and order tracking accurately and efficiently. Test cases cover various scenarios, including normal operations, edge cases, and error handling, aiming to uncover defects related to functionality. By conducting comprehensive functional testing, teams can ensure the application meets user expectations, functions reliably across different devices and platforms, and delivers a seamless user experience.

11.2.3 System testing:

System testing for a food delivery application involves evaluating the application as a whole to ensure it meets its intended objectives and functions correctly in its operational environment. System tests validate end-to-end scenarios, including ordering food, processing payments, tracking orders, and handling various user interactions across different devices and platforms. It also includes testing non-functional aspects such as performance, scalability, and reliability under realistic usage conditions. By conducting thorough system testing, teams can identify and address any issues related to functionality, usability, performance, and integration, thereby ensuring the application performs effectively and meets the needs of its users and stakeholders.

11.2.4 Performance testing:

Performance testing for a food delivery application involves assessing its responsiveness, stability, scalability, and speed under various conditions. Key aspects include load testing to simulate concurrent user interactions, stress testing to determine the application's breaking point, and endurance testing to assess performance over extended periods. Performance tests measure metrics such as response times, throughput, resource utilization (CPU, memory), and error rates to identify bottlenecks and areas for optimization.

11.2.5 Integration testing:

Integration testing for a food delivery application involves verifying that individual components or modules work together as expected when integrated into larger subsystems or the entire system. This testing phase focuses on

detecting interface defects between integrated components, ensuring data communication and interactions function correctly. Integration tests are designed to validate the flow of data between various modules, APIs, databases, and external services used by theapplication. Test cases simulate real-world scenarios to confirm that integrated components collaborate seamlessly, handle data correctly, and maintain system integrity. By conducting comprehensive integration testing, teams can identify and rectify issues early in the development lifecycle, ensuring smooth interactions between different parts of the application and enhancing overall reliability and performance.

11.2.6 Acceptance testing:

Acceptance testing for a food delivery application involves validating whether theapplication meets specified business requirements and user expectations. This testing phase is conducted by stakeholders or end-users to determine if the application is ready for deployment and use in its intended environment. Acceptance tests focus on verifying that the application functions as intended from a user's perspective, including its usability, functionality, performance, and compliance with business requirements. By successfully passing acceptance testing, the application demonstrates its readiness for deployment, ensuring it meets the needs and satisfaction of its intended users.

11.2.7 Build the test plan:

The test plan for the food delivery application aims to systematically validate its functionality, usability, performance, security, and compatibility. It begins with anoverview outlining the testing objectives and scope, focusing on features like order placement, payment processing, and order tracking. Testing types include functional testing to verify user interactions, non-functional testing for performance under load and stress, security testing for data protection, and compatibility testing across devices and platforms. The plan details the test environment requirements, specifying hardware, software, and tools necessary for testing execution and management. Test cases are meticulously designed to cover various scenarios, ensuring comprehensive coverage of both expected and edge-case behaviors. the application meets quality standards before deployment.

12.CONCLUSION AND FUTURE SCOPE

The QD Square (Quick Data Drop) project serves as an innovative solution for secure, efficient, and user-friendly online file storage and management. Designed with the objective of addressing the growing need for simplified data exchange in academic and organizational environments, the platform brings together essential features such as user authentication, guest access, file uploading, file management, and profile customization. These features are cohesively integrated through a clean and responsive user interface. Throughout the development process, fundamental front-end technologies like HTML, CSS, and JavaScript were utilized to build a structured and interactive user experience. Each page in the project—from index.html to manage.html and profile.html—performs a dedicated function within the system. For instance, the login and registration modules ensure that only authenticated users can access and manage files, while the guest access feature provides limited functionality to non-registered visitors. The homepage and about sections introduce the application and guide users through its usage.

The future of file storage in the cloud is poised for several enhancements driven by technological advancements and evolving user needs. Increased Security Measures: As cyber threats continue to evolve, cloud storage providers will focus on enhancing security measures. This may include advancements in encryption technologies, stronger authentication methods, and more sophisticated threat detection and response capabilities. Artificial Intelligence and Machine Learning Integration: AI and ML will play a crucial role in optimizing cloud storage operations. These technologies can automate data management tasks such as categorization, indexing, and predictive analytics to improve storage efficiency and data accessibility. Edge Computing Integration With the rise of IoT devices and applications requiring low-latency data processing, edge computing will integrate with cloud storage. This integration will enable data to be stored and processed closer to where it is generated, improving responsiveness and reducing bandwidth usage.

13.REFERENCES

[1.] F. Aakanksha, A.Kumar, C. Varol, H.Cho and A. Rasheed, "**Presistent Browser Storage Data Extractor**," 2024 **12th International Symposium Data Forensics and Security (ISDFS)**, San Antonio, TX,USA,2024,pp. 1-5, doi: 10.1109/ISDF60797.2024.10527245.

[2.] B.Gong, C.Guo, Y.J. Liu and Q. Wang, "**Toward Secure Data Storage in Web 3.0:Ciphertext-Policy Attribute-Based Encryption**," in **IEEE Network**, vol. 37, no.6, pp.42-49, Nov.2023,doi:10.1109/MNET.2023.3317109.

[3.] M.Song, Z. Hua, Y.Zheeng,H,Huang and X.Jia, "**Blockchain-Based Deduplication and Integrity Auditing Over Encrypted Cloud Storage**", in **IEEE Transactions on Dependable and Secure Computing**, vol.20, no.6, pp.4928-4945, Nov.-Dec. 2023, doi:10.1109/TDSC.2023.3237221.

[4.] S.Sun, H.Ma, Z.Song and R. Zhang, "**Webcloud: web-based cloud storage for secure data sharing across platforms**," in **IEEE Transactions on Dependable and Secure Computing**, vol. 19, no.3, pp. 1871-1884, 1 May-June 2022, doi:10.1109/TDSC.2020.3040784.

[5.] X.Ma, W.Yang, Y.Zhu and Z.Bai, "**A Secure and Efficient Data Deduplication Schema with Dynamic Ownership Management in Cloud Computing**" 2022 **IEEE International Performance, Computing and Communications Conference (IPCCC)**, Austin, TX, USA,2022, pp.194-201,doi:0.1109/IPCCC55026.2022.9894331.

[6.] W.Susilo, P.Jiang, J.Lai, F.Guo, G.Yang and R.H.Deng, "**Sanitizable Access Control System for Secure Cloud Storage Against Malicious Data Publishers**", in **IEEE Transactions on Dependable and Secure Computing**, vol.19 no.3,pp. 2138-2148, 1 May-June 2022, doi: 10.1109/TDSC.2021.3058132.

[7.] Y.Ing, B. He and C.Wang, "**Efficient Revocable Multi-Authority Attribute-Based Encryption for Cloud Storage**" in **IEEE Access**, vol.9,pp.42593-42603,2021,doi:10.1109/ACCESS.2021.3066212.

[8.] P. Yang. N.Xiong and J.Ren, "**Data security and privacy protection for cloud storage : A Survey**," in **IEEE Access**, vol.8, pp. 131723-131740,2020, doi:10.1109/ACCESS.2020.3009876.

[9.] A.Augustus Devarajan and T. SudalaiMuthu, "**Cloud Storage Monitoring system analyzing through File Access Pattern**," 2019 **International Conference on Computational Intelligence in Data Science (ICCIDS)**, Chennai, India, 2019,pp.1-6,

doi:10.1109/ICCIDS.2019.8862113.
[10.] A.Aich, A.Krishna,V. Akhilesh and C.Hegde, "**Encoding Web-based Data for Efficient Storage in Machine Learning Applications**," 2019 **Fifteenth International Conference on Information Processing (ICINPRO)**, Bengaluru, India, 2019, pp. 1-6, doi:10.1109/ICInPro47689 .2019.9092264.