DESIGNING CLOAKING WALL MODEL WITH LEAKAGE-SUPPRESSED AND LIGHTWEIGHT ACCESS CONTROL FOR CLOUD DATA STORAGE

Submitted by

VIGNESH A P

Register No: 910622301057

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION & COMMUNICATION ENGINEERING

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

MASTER OF COMPUTER APPLICATIONS (MCA)

in

K.L.N COLLEGE OF ENGINEERING

(An Autonomous Institution)

POTTAPALAYAM, SIVAGANGAI – 630 612.

ANNA UNIVERSITY, CHENNAI – 600 025.

JULY 2024

DESIGNING CLOAKING WALL MODEL WITH LEAKAGE-SUPPRESSED AND LIGHTWEIGHT ACCESS CONTROL FOR CLOUD DATA STORAGE

Submitted by

VIGNESH A P

Register No: 910622301057

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION & COMMUNICATION ENGINEERING

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

MASTER OF COMPUTER APPLICATIONS (MCA)

in

K.L.N COLLEGE OF ENGINEERING

(An Autonomous Institution)

POTTAPALAYAM, SIVAGANGAI – 630 612.

ANNA UNIVERSITY, CHENNAI – 600 025.

JULY 2024

K.L.N COLLEGE OF ENGINEERING

(An Autonomous Institution)

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS(MCA)

BONAFIDE CERTIFICATE

Certified that this project report titled **"DESIGNING CLOAKING WALL MODEL WITH LEAKAGE-SUPPRESSED AND LIGHTWEIGHT ACCESS CONTROL FOR CLOUD DATA STORAGE"** is the bonafide work of **Mr. VIGNESH A P (Reg No: 910622301057)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basics of which a degree or award was conferred on an earlier occasion on his or any other candidate.

Dr. MR. ILANGO MCA, M.Phil., Ph.D.,	**Mr. K. TAMIL SELVAM M.C.A, M.Phil., Ph. D.,**
DIRECTOR / MCA	**INTERNAL GUIDE,**
K.L.N College of Engineering	**K.L.N College of Engineering.**

Certified that the candidate was examined by us in the viva-voce examination held at K.L.N college of Engineering, Sivagangai on …………………………….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

The Mind
building the future with mind...

Date: 14/06/2024

Place: Trichy

Project Completion Certificate

This is to certify that **Mr.A.P.VIGNESH (Reg.No:910622301057)** Student of **MCA.,** (Master of Computer Applications) in **K.L.N College of Engineering, Madurai**, has successfully completed the **"Designing Cloaking Wall Model With Leakage-Suppressed and Lightweight Access Control for Cloud Data Storage"** in **Python** Platform from **January 2024 to June 2024** in our project title company. During the period, he had been exposed to different processes and was found to be Punctual, Hard Working, and Inquisitive. We wish him every success in life and career.

Sincerely

THE MIND IT SOLUTION ★ TRICHY-2 ★

[illegible]

Intern Coordinator

The Mind IT

The Mind IT
No.1, Ground Floor, Christ Church Complex, Chatram Bus Stand, Trichy, Tamil Nadu-620002.
www.themind.co.in, hr@themind.co.in, 0431-3559737.

ACKNOWLEDGEMENT

I would like to express my sincere thanks to our honorable Management and our Respected **PRINCIPAL, Dr. A.V. RAM PRASAD M.E., Ph.D.,** for his support and wonderful opportunity for doing the project in our college.

I would like to express my sincere thanks to **DIRECTOR Dr. MR. ILANGO MCA., M.Phil., Ph.D.,** Director of MCA Department, for his permission, guidance and support in preceding the project.

I would like to express my sincere inertness and gratitude to **MR. K. TAMIL SELVAM M.C.A, M.Phil., Ph.D.,** Internal guide of MCA Department, for his immense guidance and valuable suggestion.

I would like to express my sincere thanks to my Project team **THE MIND IT SOLUTION, TRICHY**, for their immense guidance and valuable suggestion in completing my project successfully.

I thank my parents, faculty members of MCA Department and my friends for their moral support to complete the project in a successful manner.

A project is never the outcome of a single person's efforts. It is a confluence of a varied thought process harmoniously integrated into a resourceful product. It is, but natural that I feel indebted to several people for having made this project possible.

ABSTRACT

Cloud computing has revolutionized organizational operations by providing convenient, on-demand access to resources. Cross-organisation data sharing is challenging because all the involved organisations must agree on ‘how’ and ‘why’ the data is processed. Data protection is the primary concern in the area of information security and cloud computing. Due to a lack of transparency, the organisations need to trust that others comply with the agreements and regulations. The data owned by a cloud are considered strictly confidential and require strong protection.

In this project, we focus on developing the Cloaking Wall Model and apply it to the cloud storage while protecting the access patterns of stored data. The model incorporates four methods: Long-Term Cloaking, Multi-Region based Cloaking, Time-based Cloaking, and Geolocation-based Cloaking. A typical cloaking scheme identifies whether a data user is a bot or not and returns benign content for data users and disguise content for bots. The Camouflage Data Disguise technique integrating Chaffing and Winnowing with the robust ChaCha20 encryption algorithm that provides disguise data for unauthorised access permissioned users or Bot. This enables the distribution of malicious content only to the right target.

These methods collectively fortify data security, ensuring persistent confidentiality, global consistency, timed access control, and location-sensitive protection. Furthermore, the proposed scheme optimizes certificate and key management, reducing workloads and enhancing system simplicity. This project addresses critical challenges in cross-organizational data sharing, offering a robust solution for secure and privacy-preserving cloud data access. Additionally, the proposed scheme can reduce the workloads of certificate management and simplify key management.

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

1.1. ORGRANIZATION PROFILE

The Mind IT Solution is one of the few IT system integration, professional service and software development companies in Macedonia that works with Enterprise systems and companies which has sister concern in Trichy. As a privately owned company, The Mind IT Solution provides IT Consultancy, software design and development as well as professional services and hardware deployment and maintenance to the following verticals: Government (Local and Central), Financial Services (insurance, banking and clearing house), Telecommunications, Energy and Utilities, Health Care and Education.

Mission

The Mind IT Solution' mission is to enhance the business operation of its clients by developing and/or implementing premium IT products and services.

- Providing high quality software development services, professional consulting and development outsourcing that would improve our customers' operations
- Making access to information easier and securer (Enterprise Business)
- Improving communication and data exchange (Business to Business)
- Providing our customers with a Value for Money and

Vision

The Mind IT Solution is a leading IT company for Consulting Services and Deployment of best of breed Business Solutions to top tier domestic and international customers.

Contact us

The Mind IT

No.1, Ground Floor, Christ Church Complex,

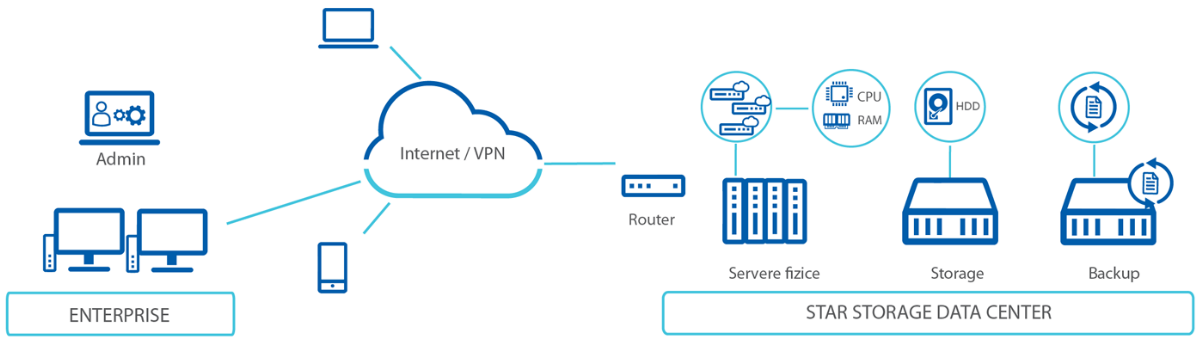
Chatram bus stand, Trichy, Tamil Nadu-620002.

www.themind.co.in, hr@themind.co.in,

0431-3562302.

1.2. PROJECT DESCRIPTION

An enterprise cloud brings together private, public, and distributed clouds in a unified IT environment. It offers a centralized control point. From there, businesses can manage enterprise cloud applications and infrastructure in any cloud. An enterprise cloud provides businesses with a seamless, consistent, and high-performance experience. Enterprise cloud computing is the process of using virtualized IT resources such as external servers, processing power, data storage capacity, databases, developer tools, and networking infrastructure by companies and organizations. Enterprise cloud solutions help organizations optimize their operations and cut costs.



The cloud computing framework provides an optimal environment for faster, safer and cheaper delivery of IT services within an enterprise. The enterprise architecture and cloud computing model form the skeleton and blueprint that gives form to the digital side of your organization.

Key Components

Enterprise cloud architecture isn’t a monolith, it’s a carefully organized combination of several different elements that work in tandem. While the specific architecture can vary depending on the service provider and the requirements of the business, the following are generally the key components:

- **Compute Resources**

This is where all the processing happens. Compute resources can range from virtual machines to containers that execute applications and services.

Storage Services

Think of this as the virtual equivalent of file cabinets and storage rooms. Enterprise cloud storage is far more advanced than your regular hard drives, offering high-speed access and superior data redundancy.

- **Networking**

This forms the backbone of the cloud architecture, interconnecting various services and components. Enterprise cloud networking often includes load balancing, VPNs, and private subnets for enhanced security and efficiency.

- **Database Services**

These are specialized storage services optimized for handling structured data. They serve as the backend for applications that require data retrieval and storage capabilities, often in real-time.

- **Content Delivery Network (CDN)**

A CDN helps in distributing the flow of network traffic across multiple servers, ensuring high availability and reliability. It's particularly crucial for businesses that serve a global audience.

- **Monitoring and Analytics Tools**

These tools keep tabs on performance metrics, usage statistics, and security incidents. They are essential for maintaining optimal performance and security.

- **Security Services**

Enterprise cloud security includes features such as Identity and Access Management (IAM), encryption, and intrusion detection systems, in addition to basic firewalls.

- **APIs and SDKs**

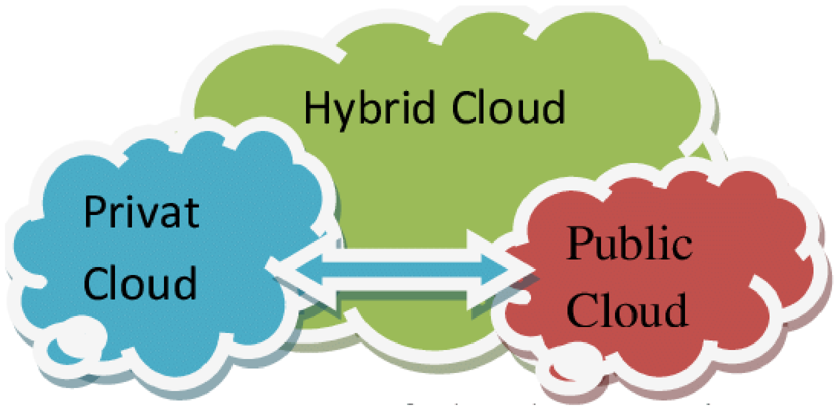
These are the building blocks that allow for customization and integration with other services and applications. They offer the means to not only extract more value from your cloud services but also to integrate them seamlessly into existing workflows.

- **Orchestration and Automation**

These components allow businesses to automate repetitive tasks and orchestrate complex workflows, enhancing efficiency exponentially. Start-ups can test out new business ideas risk-free and at low cost, due to enormous scalability. Since there is no upfront capital expense involved, in case a new project takes off, it can be scaled up instantly and vice versa. Enterprise cloud computing allows a company to create a shared workspace in order to collaborate with its trading partners and work together as a 'virtual enterprise network'.

Types of enterprise cloud architecture

In the world of computing and more precisely the **enterprise cloud**, cloud solutions have become essential. Businesses need a solution that will allow them to access their data and applications anytime, anywhere. There are four common models for enterprise cloud, each with its own advantages and use cases.



- **Public cloud**

Public cloud is a computing model in which cloud infrastructure and services are provided over a network that is accessible by the general public. All customers can use the services offered by the vendor. This means that the cloud resources are owned, managed, and operated by a third-party cloud computing service provider. Examples of such cloud solutions are Google Cloud Platform (GCP), Amazon Web Services (AWS), and Microsoft Azure.

- **Private cloud**

Private Cloud is exclusively dedicated to a single organization. Only verified users can access the cloud, as it's not available to the public. It's a type of cloud computing that involves companies creating their own infrastructure, either in-house, or through a third-party provider who hosts and secures it.

- **Hybrid cloud**

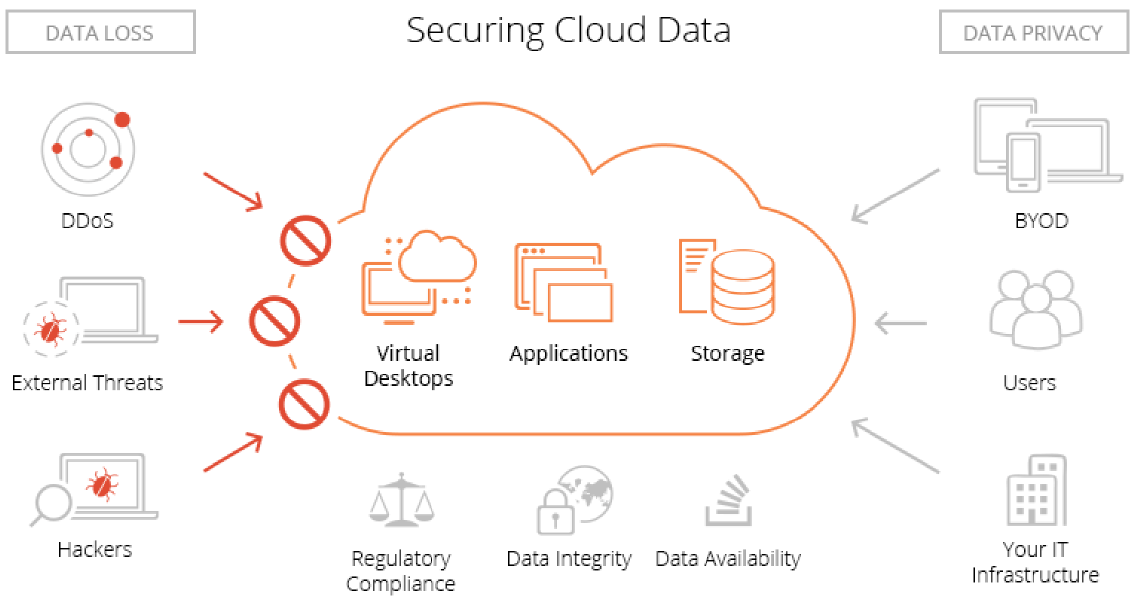
Hybrid cloud is a model that combines the use of public and private clouds. Both environments can communicate and exchange data freely. A hybrid cloud allows organizations to take advantage of the scalability and cost-effectiveness of the public cloud while maintaining the security and compliance of sensitive data in a private infrastructure.

- **Multi-cloud**

The multi-cloud model is a more advanced and expanded variant of the hybrid cloud. In this structure, a company can use more than one provider and freely combine private and public clouds according to its needs. This is a good solution for enterprise-class businesses that have large specialized departments. In this way, the data of the finance or legal department can be managed separately from that of the marketing or HR departments.

1.2.1. PROBLEMS DEFINITION

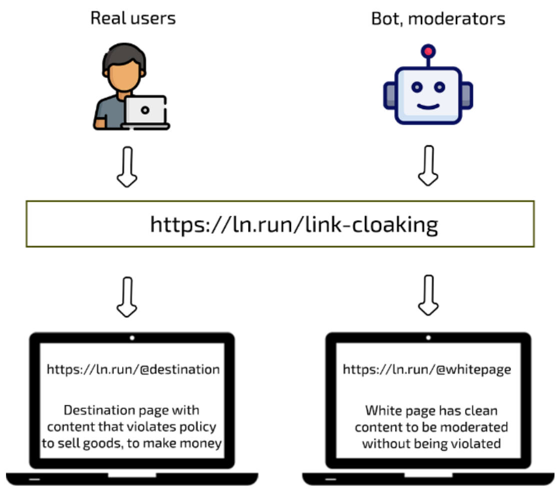
Rapid globalization of technology and the ever-expanding interconnectedness of the 'Internet of Things' will continue to demand our constant attention to considerations around information security; all channels, all devices, all the time. Cloud security considerations span a range of concerns; resource connectivity, user entitlements, data loss prevention, transitory/stationary data handling and encryption policies, data security classification restrictions, cross-border information flow...the list goes on. It is not uncommon for information security considerations to run counter to cloud solution patterns. Without clear information security guidelines articulated in the Cloud Strategy, it is unlikely that the organization will be well protected from security risks in the cloud.



The discipline of cyber security responsible for securing the cloud computing environment is known as cloud security. Essentially cloud security has the same goals as cybersecurity. Cloud security is aimed at protecting data, infrastructure, and applications on the cloud. It requires the administrator to protect data on a third party's infrastructure. Today, enterprises are transitioning to cloud-based environments at unimaginable speeds. As per estimates, the worldwide public cloud services market is set to grow by 20.4% in 2022. The migration of data and business content on the cloud has IT professionals concerned because of the security and governance of the cloud environment. Protecting customer data is crucial for building trust for the service providers. Tradition network security and cloud security differ in certain aspects. The access in traditional network security is controlled using a perimeter model. Whereas the cloud environment is highly connected and makes it much easier to pass the security measures at the perimeter. In the case of cloud security, everything is software and the cloud-based security solutions need to respond to environmental changes through cloud-based management systems or application processing interfaces (APIs) to cater to the dynamic environment. Due to the large scale and flexibility of the cloud environment, implementing security solutions in the cloud environment face numerous challenges. Since the data on the cloud is accessed outside the corporate networks on numerous occasions maintaining a record of the data access is difficult. In essence, the problem revolves around the necessity for an advanced and comprehensive approach to cloud data storage security. The proposed Cloaking Wall Model aims to provide a holistic solution by addressing these concerns, offering persistent confidentiality, global consistency, timed access controls, and location-sensitive protection. Moreover, the model strives to optimize certificate and key management, contributing to a more streamlined and secure cloud data storage environment.

1.2.3. CLOAKING

The “cyber cloaking” initiative leverages emerging technology that can actually hide (make invisible) or “cloak” any IP device, server(s) or secure cloud services rendering them invisible to internal searches, external cyber-hackers, and internet bots. Cloaking prevents leakage of information or service that is vulnerable to web attacks. HTTP headers and return codes are concealed before sending a response to a client.


Real users
Bot, moderators
https://ln.run/link-cloaking
https://ln.run/@destination
Destination page with content that violates policy to sell goods, to make money
https://ln.run/@whitepage
White page has clean content to be moderated without being violated

Cloaking provides strong protection against breaches, unauthorized insider access, injection attacks, and cloud misconfigurations. Cloaked environment is continuously monitored (real-time) for threats and for compliance. Cloaking the device’s address gives a hacker nothing to see, and it can be done on systems ranging from government networks to medical electronics implanted inside human beings. The result goes beyond shielding. Network nodes simply cease to exist for outsiders trying to engage in malicious acts. The same security process also speeds up data transfer among diverse networks sharing similarly cloaked data and devices.

CAMOUFLAGE DATA DISGUISE

Today, ongoing data provisioning creates significant risk within less secure organizations. A greater number of data copies in non-production environments, coupled with widespread access to that data, increase the risk of data breaches from both external attacks and insider threats. While external attacks are often sensationalized, the real threat to sensitive data is from insiders. The harsh reality is that in most cases users had authorized access to the data they stole. On the other hand, companies are using data to support regular and on-going activities, such as application development, research and analysis, testing, and outsourcing.

With the growth of data copies and of access to sensitive data, the biggest challenge remains to be securing these non-production environments while enabling various data users to complete mission critical work. Camouflage Data Disguise enables organizations to safely use data for critical business processes without exposing sensitive information. It mitigates the risk of data breach and non-compliance by de-identifying sensitive data in non-production environments. Camouflage Data Masking replaces sensitive data with fictional but realistic values that maintain referential integrity, enabling data driven business processes to operate normally. Camouflage techniques include concealment, disguise, and dummies. Data Camouflage adds a light masking to an application's data: it simply scrambles the data to mask the file on the disk from casual observation. With this approach, one can protect data on disk from unauthorized inspection.

Chaffing and Winnowing

Chaffing and Winnowing is a cryptographic algorithm that enhances the security and privacy of transmitted data by introducing decoy information and subsequently isolating the genuine content. Here's a more detailed explanation: Chaffing and Winnowing provide a mechanism to obfuscate data during transmission by blending genuine information with decoy elements and then selectively extracting the real content using a secure key or algorithm. This technique contributes to the confidentiality and integrity of transmitted data, particularly in scenarios where privacy and protection against unauthorized interception are paramount.

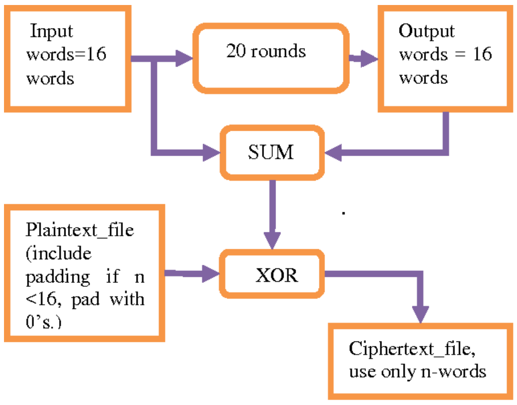
Chaffing: Chaffing is the initial step in the process, where decoy or fake data, known as "chaff," is intentionally added to the actual information being transmitted. The chaff is designed to mimic the characteristics of the authentic data, making it challenging for unauthorized entities to discern between the real and the decoy elements. This introduces a level of confusion and complexity for anyone attempting to intercept or analyse the transmitted information.

Winnowing: Winnowing is the complementary process to chaffing. It involves the selective separation of the genuine data from the added chaff, using a specific key or algorithm known only to the intended recipient. The key or algorithm serves as the means to distinguish between the authentic content and the deliberately introduced decoy elements. By applying this key during the winnowing process, the recipient can effectively filter out the chaff, revealing the original and unaltered information.

ChaCha20

ChaCha20 is a symmetric key stream cipher and one of the modern encryption algorithms. It was designed by Daniel J. Bernstein, and it's a part of the Salsa20 family of stream ciphers. ChaCha20 is known for its simplicity, speed, and resistance to cryptanalysis.

The ChaCha20 encryption algorithm is designed to provide a combination of speed and security. It is constructed to resist known attacks, including differential cryptanalysis and linear cryptanalysis. Furthermore, it is highly parallelizable, making it easily adaptable to multi-core processors and other high-performance computing systems.


Input words=16 words
20 rounds
Output words = 16 words
SUM
Plaintext_file (include padding if n <16, pad with 0's.)
XOR
Ciphertext_file, use only n-words

ChaCha20 is a stream cipher, meaning it encrypts data in a continuous stream rather than fixed-size blocks. It generates a continuous keystream of pseudo-random bits, which are then XORed with the plaintext data to produce the ciphertext.

Fundamental steps in the ChaCha20 encryption process

- Key Generation: The ChaCha20 algorithm generates a 256-bit key from a user-supplied key and a randomly generated 96-bit nonce.
- Initialization: The algorithm uses the key and nonce to initialize the cipher's state.
- Data Encryption: ChaCha20 encrypts each data block using the state of the cipher, which is updated after processing each block.
- Output: The resulting cipher text is produced by XORing the plaintext with the output of the data encryption step.

1.2.4. AIM AND OBJECTIVE

Aim

The aim of the project is to establish an advanced cloud data security framework by designing a Cloaking Wall Model integrated with camouflage techniques. This framework seeks to enhance privacy and access control for sensitive data stored in cloud environments.

Objectives

- To Develop a Robust Cloaking Wall Model: Create a secure foundation for advanced cloud data security.
- To Integrate Camouflage Techniques: Implement Camouflage Data Disguise for enhanced privacy.
- To Ensure Persistent Confidentiality: Fortify data security to prevent unauthorized access.
- To Achieve Global Consistency and Access Control: Establish mechanisms for timed access control and global consistency.
- To Implement Location-Sensitive Data Protection: Introduce measures for enhanced data protection based on user geography.
- To Develop Bot Identification and Targeted Content Distribution: Create a mechanism for bot identification and selective content distribution.
- To Optimize Certificate and Key Management: Streamline processes to reduce workloads and enhance simplicity.
- To Address Cross-Organizational Data Sharing Challenges: Provide a robust solution for secure cross-organizational data access.
- To Reduce Workloads of Certificate Management: Implement measures to streamline certificate management.
- To Simplify Key Management: Develop strategies for key management simplification while ensuring security.

1.2.5. SCOPE OF THE PROJECT

The project's scope is to enhance cloud data security through the implementation of the Cloaking Wall Model integrated with advanced camouflage techniques. This includes the development of a robust security framework, leveraging Long-Term Cloaking, Multi-Region, Time-based, and Geolocation-based Cloaking. The project also explores the integration of Chaffing and Winnowing with the ChaCha20 encryption algorithm for added privacy. Key objectives encompass persistent confidentiality, global consistency, timed access control, and location-sensitive protection. Additionally, the project addresses challenges in cross-organizational data sharing, integrates a bot identification mechanism, and optimizes certificate and key management for reduced workloads. The ultimate goal is to contribute to a secure and privacy-preserving cloud data access environment while ensuring system efficiency and simplicity.

2. SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

The existing system of cloud outsourced data protection encompasses various mechanisms and practices implemented to safeguard data stored in the cloud environment. Here is an overview of key elements in the current landscape:

- **Encryption:**

Encryption is a fundamental component of data protection in the cloud. It involves the use of cryptographic algorithms to convert data into a secure format that can only be accessed with the appropriate decryption key. Both data at rest and data in transit are typically encrypted to prevent unauthorized access.

- **Access Controls and Identity Management:**

Robust access controls and identity management systems are implemented to regulate who can access data in the cloud. This involves assigning and managing user roles, permissions, and authentication mechanisms to ensure that only authorized individuals or systems can interact with sensitive information.

- **Firewalls and Network Security:**

Network security measures, including firewalls, are employed to protect the cloud infrastructure. Firewalls monitor and control incoming and outgoing network traffic based on predetermined security rules. This helps prevent unauthorized access and potential cyber threats.

- **Regular Audits and Monitoring:**

Continuous monitoring and regular audits of cloud environments are conducted to identify and respond to security incidents promptly. This involves tracking user activities, system events, and potential vulnerabilities, providing a proactive approach to addressing security concerns.

2.1.1. Disadvantages

- Encryption complexity may impact system performance.
- Traditional authentication methods may be vulnerable to attacks.
- Difficulty in navigating complex data protection regulations and ensuring compliance.
- Heavy reliance on cloud service providers, with potential vulnerabilities in their security practices impacting user data.
- Constraints in tailoring security measures to unique organizational requirements.
- Service disruptions, maintenance, or cyber-attacks leading to temporary loss of access to data.
- Secure protocols may be inconsistently implemented

2.2. PROPOSED SYSTEM

The proposed system endeavours to revolutionize cloud data security by introducing a sophisticated Cloaking Wall Model specifically designed to safeguard organizational operations in the ever-evolving landscape of cloud computing. Addressing the inherent challenges of cross-organizational data sharing, the system prioritizes the development of advanced security measures. Among these measures are leakage-suppressed access controls, ensuring that data remains confidential and shielded from unauthorized exposure. Additionally, lightweight access controls are implemented to streamline user authentication and authorization processes, minimizing computational overhead. The proposed system incorporates cutting-edge encryption techniques to ensure persistent confidentiality of data stored in the cloud, encompassing data at rest, in transit, and during processing. To enhance global consistency in access controls, the system adopts a unified policy approach, addressing challenges associated with multi-region data access. Furthermore, time-based access controls are implemented, allowing organizations to enforce temporal restrictions on data access and fortify security by limiting access to predefined time windows. Location-sensitive protection mechanisms are also introduced, providing an additional layer of security through geofencing and tailored policies based on user locations. The Cloaking Wall Model incorporates four distinct methods to fortify data security and protect the access patterns of stored data. These methods are strategically designed to address various aspects of security challenges in cloud data storage:

- **Long-Term Cloaking**

This method focuses on providing extended protection for sensitive data over prolonged durations. It involves concealing access patterns and data usage trends over an extended timeframe, ensuring persistent confidentiality. Long-term cloaking contributes to maintaining the privacy and security of stored data over extended periods, preventing unauthorized inference from patterns of access.

- **Multi-Region Based Cloaking**

Recognizing the global nature of cloud services, multi-region-based cloaking involves implementing security measures that transcend geographical boundaries. By considering the diverse locations from which data access may occur, this method ensures a consistent and standardized security posture globally. It addresses the challenges associated with data access from different regions, providing a unified approach to access control policies.

- **Time-Based Cloaking**

Time-based cloaking introduces temporal restrictions on data access, allowing organizations to define specific time windows during which data can be accessed. This method enhances security by limiting access to predefined timeframes, reducing the exposure of data to potential threats. Time-based cloaking adds an additional layer of control to access patterns, contributing to a more secure cloud data storage environment.

- **Geolocation-Based Cloaking**

Geolocation-based cloaking involves tailoring data protection measures based on the geographical location of users. This method adds a location-sensitive layer of security, ensuring that access to sensitive data is contingent on the user's physical location. Geolocation-based cloaking is particularly relevant for organizations with diverse and dispersed user bases, providing a customized approach to access control based on geographic parameters.

These four methods collectively form the foundation of the Cloaking Wall Model, contributing to persistent confidentiality, global consistency, timed access control, and location-sensitive protection. By integrating these techniques, the model aims to address critical challenges in cross-organizational data sharing, offering a robust solution for secure and privacy-preserving cloud data access.

- **Camouflage Data Disguise**

Camouflage Data Disguise technique represents an advanced cryptographic approach that seamlessly integrates Chaffing and Winnowing with the formidable ChaCha20 encryption algorithm. This technique is strategically designed to provide disguised data as a countermeasure against unauthorized access, targeting both unpermitted users and potentially malicious bots.

2.2.1. Advantages

- Ensures data confidentiality at rest, in transit, and during processing.
- Provides a standardized security posture globally.
- Distinguishes between authorised and unauthorised user for targeted content distribution.
- Reduces administrative workloads through efficient certificate management.
- Provides an advanced layer of data privacy, ensuring that sensitive information remains confidential during transmission.
- Minimizes the risk of unauthorized access

2.3. ABOUT THE PROJECT

Enhancing Cloud Data Security with the Cloaking Wall Model

The Cloaking Wall Model project aims to enhance data security and privacy in cloud storage environments, specifically focusing on cross-organizational data sharing. Traditional cloud data storage solutions often face challenges related to data protection, unauthorized access, and compliance with regulatory requirements. This project addresses these issues by developing a robust and innovative solution that integrates advanced security mechanisms.

Advanced Cloaking Methods for Data Protection

The Cloaking Wall Model incorporates multiple cloaking methods Long-Term Cloaking, Multi-Region based Cloaking, Time-based Cloaking, and Geolocation-based Cloaking to protect access patterns and ensure data confidentiality. It leverages the Camouflage Data Disguise technique, which combines Chaffing and Winnowing with the ChaCha20 encryption algorithm, to provide disguise data for unauthorized users or bots, thereby enhancing security.

Optimized Certificate and Key Management

This project also focuses on optimizing certificate and key management processes to reduce workloads and simplify system maintenance. By providing a comprehensive solution that includes secure data storage, flexible access control, real-time monitoring, and robust encryption, the Cloaking Wall Model aims to offer a reliable and secure platform for cross-organizational data sharing in cloud environments.

Addressing Critical Challenges in Cloud Security

The project ensures that data remains confidential, consistent, and accessible only to authorized users, addressing critical challenges in cloud data security and compliance. The Cloaking Wall Model represents a significant advancement in cloud data protection, providing organizations with the tools they need to securely share data across organizational boundaries.

2.4. FEASIBILITY STUDY

The feasibility study confirms that the project is technically, operationally, and economically viable for implementation:

1. Technical Feasibility

The project's hardware and software requirements are readily available and compatible, ensuring smooth implementation. The chosen technologies, such as Flask and MySQL, offer scalability and seamless integration capabilities with existing systems.

2. Operational Feasibility

End-users and administrators are likely to accept the system due to its benefits in enhancing data security and operational efficiency. Adequate training will be provided to ensure user familiarity and smooth adoption. The project aims to enhance existing business processes without causing disruptions.

3. Economic Feasibility

A cost-benefit analysis indicates that the benefits of improved data security and operational efficiency outweigh the project costs. The expected return on investment (ROI) is positive, considering potential cost savings and revenue generation opportunities.

Overall, the project demonstrates strong feasibility across technical, operational, and economic aspects, making it a suitable candidate for successful implementation.

3. SYSTEM DESIGN

3.1. SYSTEM DESIGN GOALS

The system design goals for the Cloaking Wall Model project are focused on delivering a secure, efficient, and scalable cloud data storage solution that addresses critical challenges in data protection, access control, and cross-organizational data sharing. These goals ensure the system meets high standards of security, performance, and user satisfaction.

1. **Data Confidentiality and Integrity**
   - **Goal:** Ensure the confidentiality and integrity of all stored data.
   - **Implementation:** Utilize robust encryption algorithms like ChaCha20 for data encryption and integrity checks to prevent unauthorized access and tampering. This project does implement these measures to secure the data effectively.
2. **Access Control and Authentication**
   - **Goal:** Regulate data access with stringent control policies.
   - **Implementation:** Implement multi-factor authentication (MFA) and role-based access control (RBAC) to ensure only authorized users can access data based on their roles and permissions. This project does incorporate these policies to manage access securely.
3. **Scalability and Performance**
   - **Goal:** Maintain high performance and scalability as data volume and user load increase.
   - **Implementation:** Design a scalable architecture using cloud infrastructure services from providers like AWS, Azure, and Google Cloud to handle dynamic loads efficiently. This project does ensure scalability and optimal performance through these architectural choices.
4. **Integration and Compatibility**
   - **Goal:** Ensure seamless integration with various cloud platforms and third-party systems.
   - **Implementation:** Develop the system with standard APIs and protocols to facilitate interoperability with existing cloud services and enterprise systems. This project does ensure compatibility and integration through standardized approaches.
5. **Real-Time Monitoring and Incident Response**
   - **Goal:** Enable real-time monitoring and swift response to security incidents.

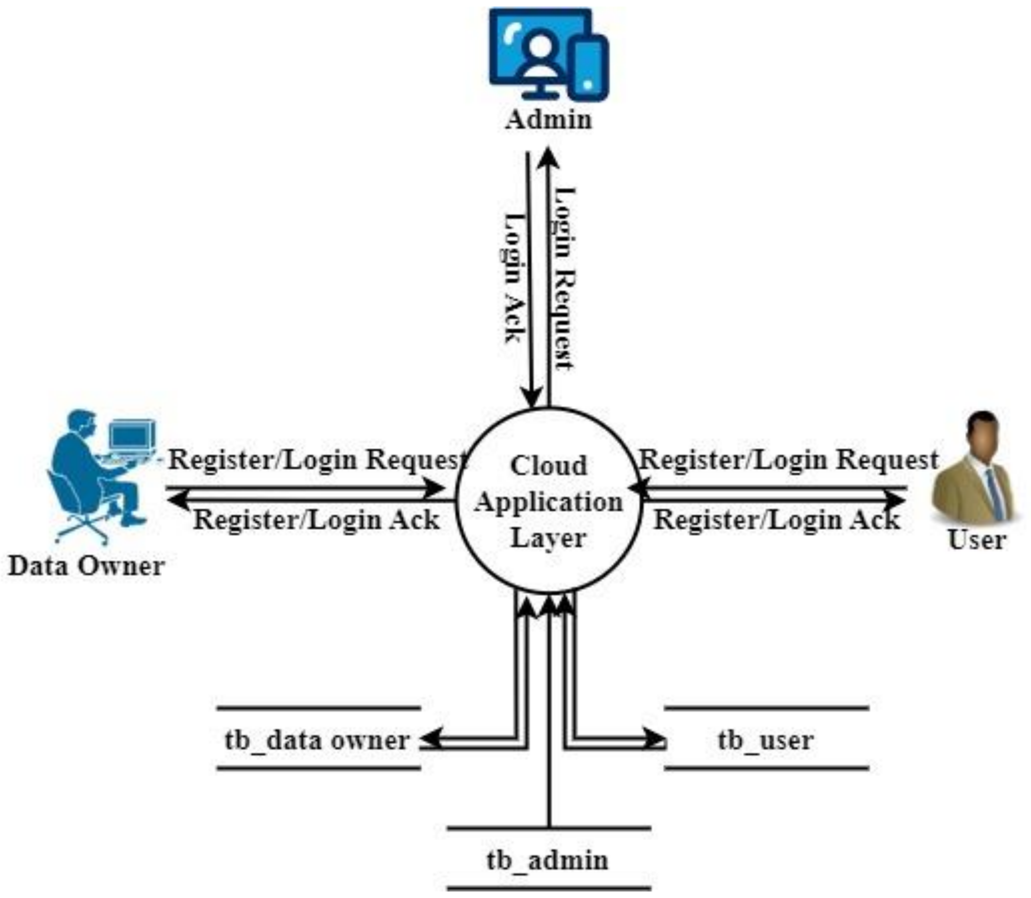
    - **Implementation:** Implement monitoring tools like Amazon CloudWatch, Azure Monitor, or Google Cloud Monitoring to track performance and detect anomalies, with automated alert systems for quick incident response. This project does incorporate real-time monitoring to ensure security.

6. **Usability and User Experience**
    - **Goal:** Design an intuitive and accessible system for users of all technical levels.
    - **Implementation:** Create user-friendly interfaces with clear navigation, consistent design elements, and accessibility features to ensure ease of use for all users. This project does focus on usability to provide a seamless user experience.
7. **Maintainability and Extensibility**
    - **Goal:** Ensure the system is easy to maintain, update, and extend.
    - **Implementation:** Use modular design principles and well-documented code, following industry-standard coding practices to facilitate ongoing maintenance and future development. This project does adopt these practices to ensure maintainability.
8. **Compliance and Legal Adherence**
    - **Goal:** Comply with relevant data protection regulations and industry standards.
    - **Implementation:** Incorporate features and processes to ensure adherence to regulations such as GDPR, HIPAA, and other data protection laws. This project does ensure compliance through meticulous implementation of legal requirements.

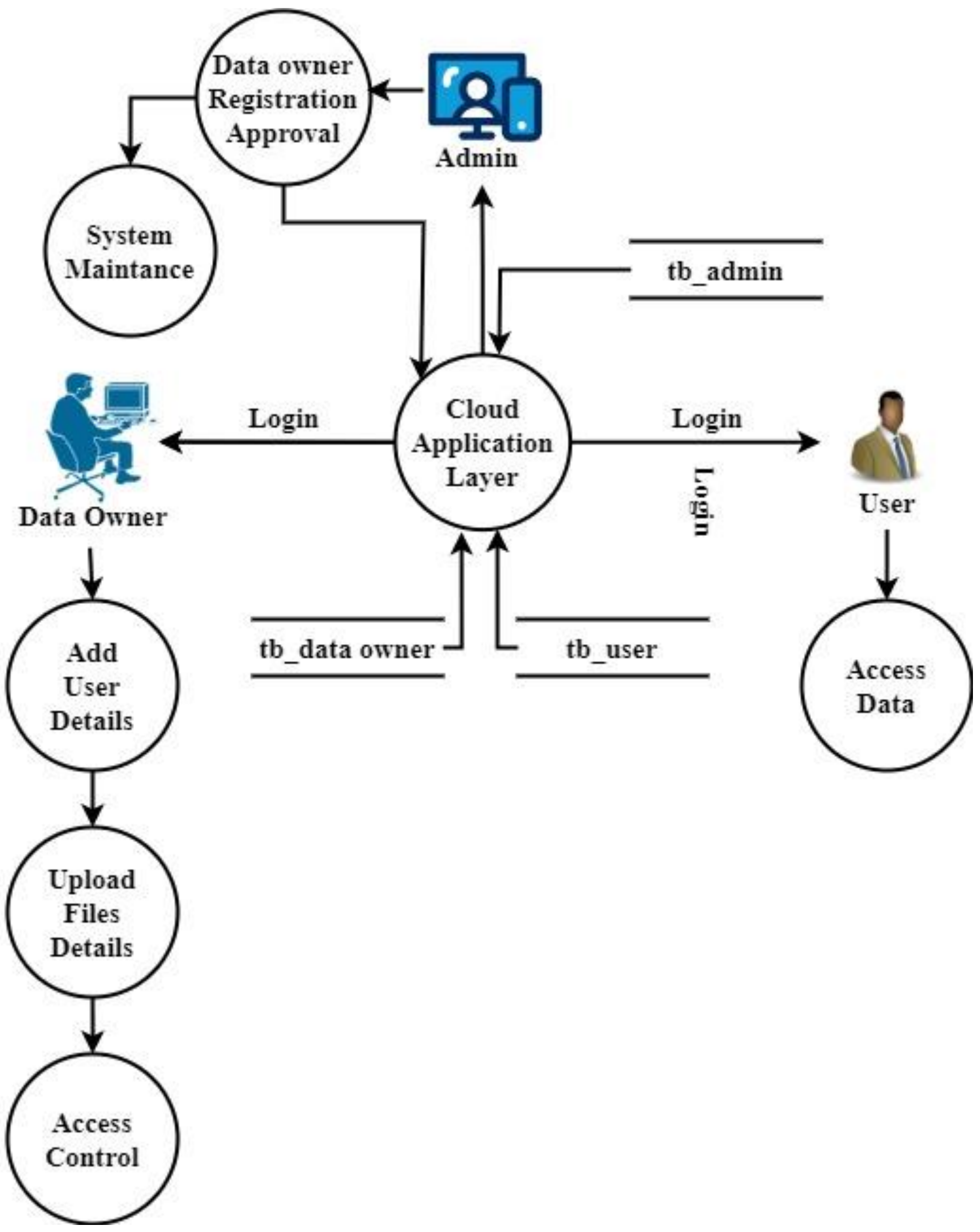
By achieving these design goals, the Cloaking Wall Model project aims to provide a comprehensive, secure, and user-friendly solution for cross-organizational data sharing in cloud environments, effectively addressing the complex requirements of modern data protection. This project does so by implementing rigorous security measures, ensuring scalability and performance, and maintaining usability and compliance throughout its design and execution.

3.2. DATA FLOW DIAGRAM

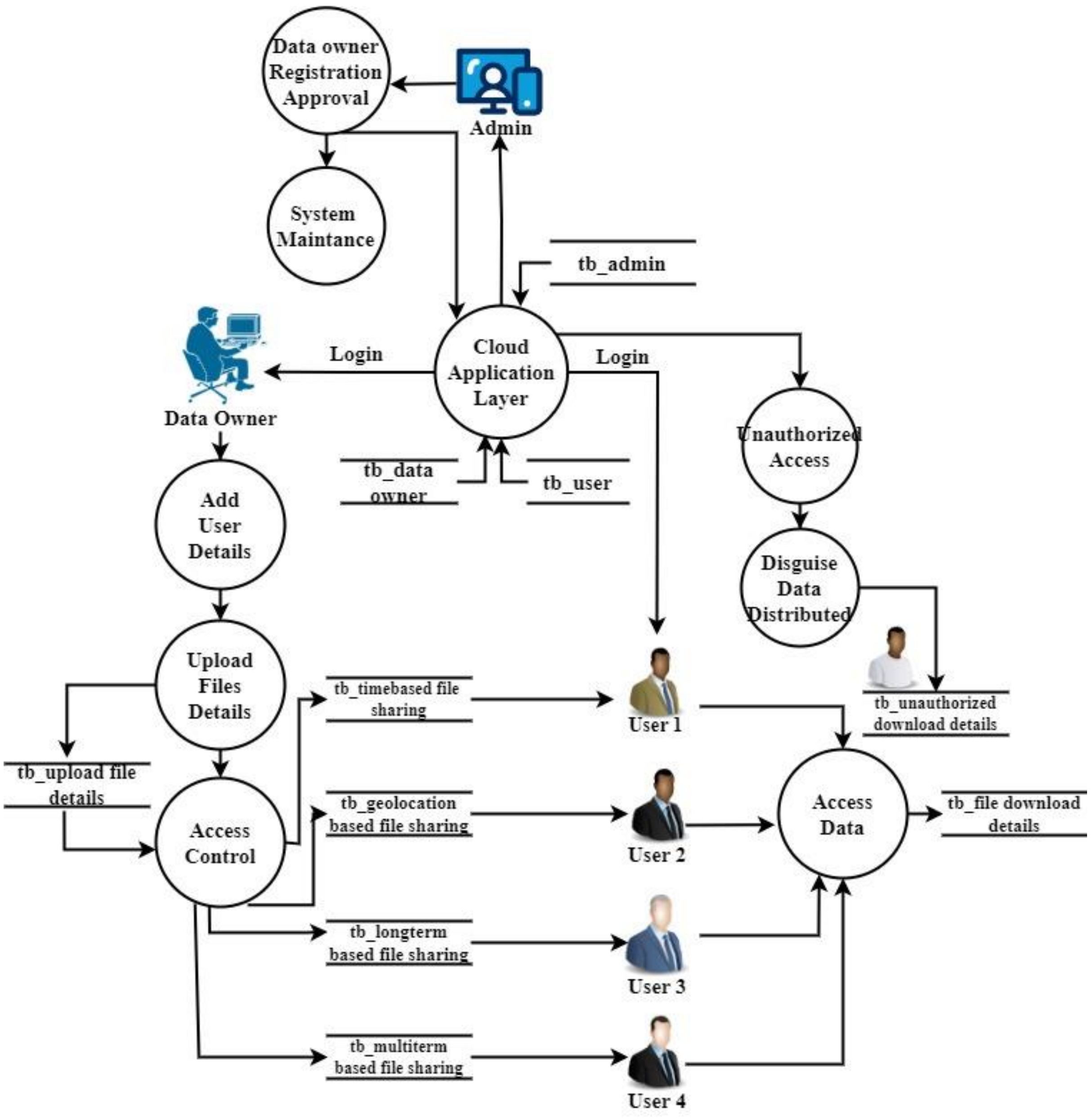
LEVEL 0



LEVEL 1

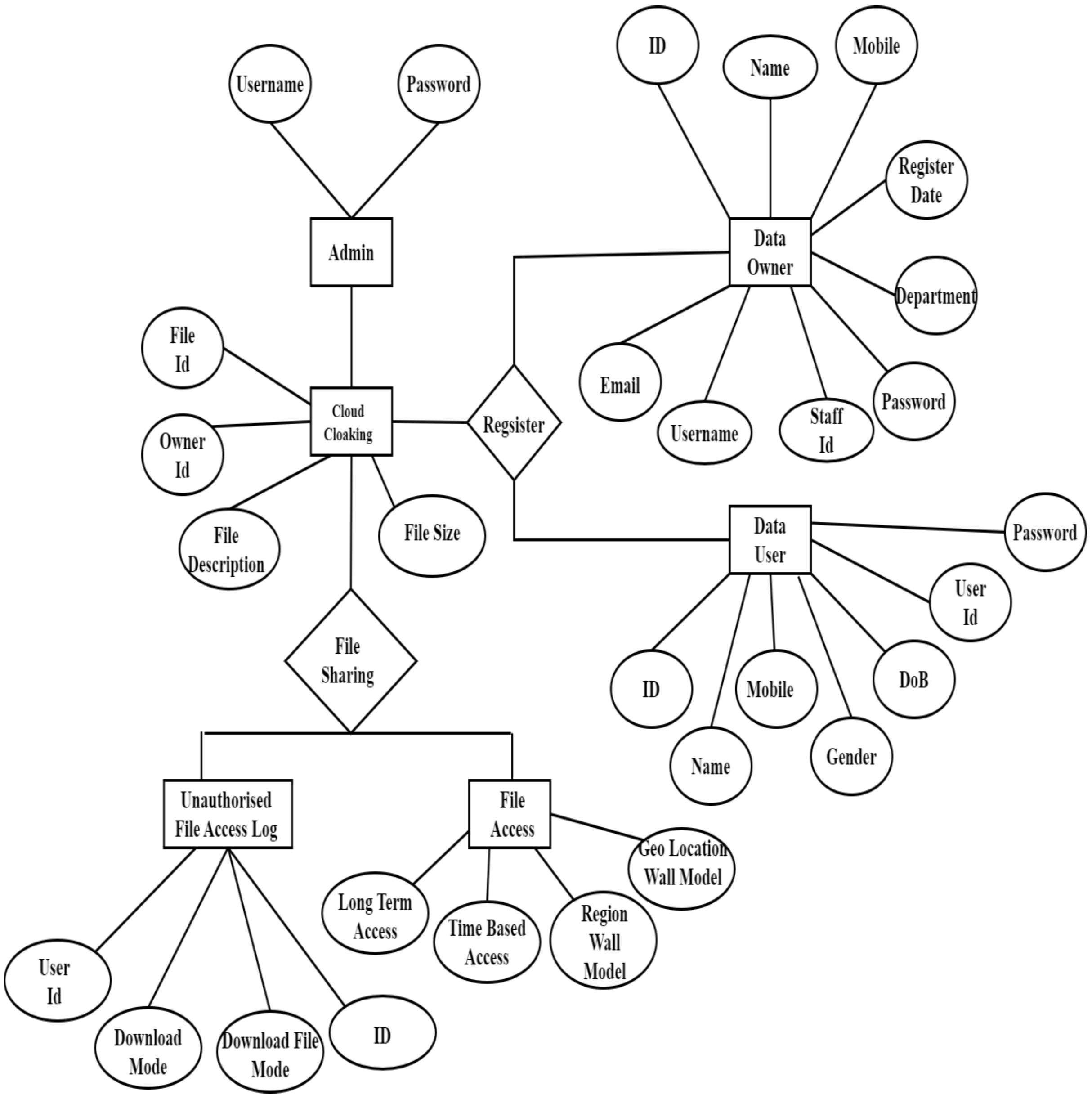

Data owner Registration Approval
Admin
System Maintance
tb_admin
Cloud Application Layer
Login
Login
Login
Data Owner
User
tb_data owner
tb_user
Add User Details
Access Data
Upload Files Details
Access Control

LEVEL 2



Data owner Registration Approval
Admin
System Maintance
tb_admin
Cloud Application Layer
Login
Login
Data Owner
Unauthorized Access
tb_data owner
tb_user
Add User Details
Disguise Data Distributed
Upload Files Details
tb_timebased file sharing
User 1
tb_unauthorized download details
tb_upload file details
Access Control
tb_geolocation based file sharing
User 2
Access Data
tb_file download details
tb_longterm based file sharing
User 3
tb_multiterm based file sharing
User 4

3.3. ER DIAGRAM



4. SYSTEM REQUIREMENTS AND ANALYSIS

4.1. SOFTWARE AND HARDWARE SPECIFICATION

4.1.1. SOFTWARE REQUIREMENTS

- **Operating System**: Windows 10 or later
- **Web Server**: WAMP (Windows, Apache, MySQL, Python/PHP/Perl) stack
- **Database Server**: MySQL
- **Programming Language**: Python 3.8
- **Web Framework:** Flask
- **Python Libraries**: Pandas, Scikit-Learn, Matplotlib, NumPy, Seaborn,
- **Integrated Development Environment (IDE)**: PyCharm,

4.1.2. HARDWARE SPECIFICATION

- **Processor**: Intel Core i5 or higher (recommended)
- **RAM**: 8GB or more
- **Storage:** SSD with at least 256GB capacity
- **Network Interface**: Ethernet or Wi-Fi adapter
- **Monitor**: Minimum resolution of 1280x800 pixels
- **Input Devices**: Keyboard and mouse

4.2. SOFTWARE EXPLANATION

4.2.1. PYTHON 3.7.4

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard libraries which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing
- Text processing and many more.

Pandas

pandas are a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. pandas are a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

Pandas is mainly used for data analysis and associated manipulation of tabular data in Data frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. The development of pandas introduced into Python many comparable features of working with Data frames that were established in the R programming language. The panda's library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on Data frames.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

4.2.2. MYSQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

4.2.3. WAMPSERVER

WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily your database.

WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world. The software is free to use and doesn't require a payment or subscription.

4.2.4. BOOTSTRAP 4

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All thanks to Bootstrap developers -Mark Otto and Jacob Thornton of Twitter, though it was later declared to be an open-source project.

Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

4.2.5. FLASK

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let’s take a closer look into Flask, so-called “micro” framework for Python. Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

5. MODULE DESCRIPTION

1. Cloud Service Provider Web App

The design and development of a Cloud Consumer Web App involve several interconnected modules, each contributing to the seamless and efficient management of cloud resources. The user authentication module serves as the entry point, ensuring secure access through robust registration and authentication processes, including multi-factor authentication for enhanced security. The heart of the application lies in the dashboard module, providing an intuitive interface for users to oversee and manage their cloud resources comprehensively. This module encompasses features for resource provisioning, scaling, and configuration, offering a centralized hub for users to interact with their cloud services seamlessly. The dashboard module, at the core of the application, seamlessly integrates the Cloaking Wall Model, providing users with an intuitive interface for comprehensive cloud resource management while ensuring global consistency in security measures. The monitoring and alert module is essential for empowering users with real-time insights into the performance of their cloud resources and timely notifications of any irregularities.

2. End User Interface

These modules collectively create a comprehensive End User Interface for both Admins or Data Owners and Data Users, incorporating secure authentication, data management functionalities, access control policies based on the Cloaking Wall Model, and monitoring capabilities to uphold data security and privacy.

2.1. Admin or Data Owner Interface

- **Login Module:**

The login module provides a secure authentication process for Admins or Data Owners, ensuring only authorized access to the cloud management interface.

- **Add and Manage Data:**

Admins can add, organize, and manage data within the cloud storage. This module allows them to upload, categorize, and control access to various datasets.

- **Add and Manage Users:**

Admins have the capability to add new users to the system, defining their roles and permissions. This module also allows them to modify or revoke access as needed.

- **Provide Login Credentials to Users:**

  Admins can generate and distribute login credentials for users added to the system. This ensures a secure onboarding process for new users.

- **Set Access Policy using Cloaking Wall Model:**

  Leveraging the Cloaking Wall Model, this module enables Admins to set access policies. Admins can define Long-Term Cloaking, Multi-Region based Cloaking, Time-based Cloaking, and Geolocation-based Cloaking to enhance data security.

- **Monitoring Data Access:**

  Admins can monitor and audit data access patterns using this module. It provides insights into who accessed specific data, when, and from which location, contributing to overall security and compliance.

2.2. Data User Interface:

- **Login Module:**

  Similar to the Admin interface, the login module provides secure authentication for Data Users, ensuring that only authorized individuals can access the cloud resources.

- **Access Data:**

  Data Users can use this module to access the data allocated to them. The interface provides a user-friendly environment for retrieving, modifying, or analysing data based on their permissions.

3. Cloaking Wall Model

The Cloaking Wall Model is a sophisticated security framework integrated into the Cloud Consumer Web App, offering advanced data protection and access control. The model comprises several modules, each contributing to a robust security architecture:

3.1. Long-Term Cloaking Module:

The Long-Term Cloaking module ensures persistent confidentiality of data access patterns over extended durations. Admins can set policies to conceal and protect access trends, preventing unauthorized inference from historical usage data.

3.2. Multi-Region Based Cloaking Module:

This module facilitates a unified security approach across diverse geographical regions. Admins can define access controls that transcend geographic boundaries, ensuring consistent security measures globally and addressing challenges related to multi-region data access.

3.3. Time-Based Cloaking Module:

Time-Based Cloaking empowers Admins to set temporal restrictions on data access. This module enhances security by allowing the definition of specific time windows during which data can be accessed, adding an extra layer of control over temporal access patterns.

3.4. Geolocation-Based Cloaking Module:

The Geolocation-Based Cloaking module tailor's data protection based on user location. Admins can define security policies that vary depending on the physical location of Data Users, adding a location-sensitive layer to access controls.

4. Access Policy Configurator

The Access Policy Configuration module is a pivotal component within the Cloaking Wall Model, empowering administrators to define and customize access policies for the Cloud Consumer Web App. This module plays a crucial role in tailoring the security measures based on Long-Term Cloaking, Multi-Region based Cloaking, Time-Based Cloaking, and Geolocation-Based Cloaking principles. This module provides a centralized dashboard for administrators to configure comprehensive access policies. They can set parameters, thresholds, and exceptions to align with the security requirements of the organization. Administrators have the flexibility to configure access policies based on user roles. Different roles within the organization may have distinct data access requirements, and this feature ensures granular control over access based on user responsibilities.

5. Bot Identification and Data Distribution

The Bot Identification Mechanism within the Cloaking Wall Model ensures that automated bots attempting to access the data in violation of access policies are promptly identified. This module ensures that benign content is delivered to authentic users, while malicious content is selectively distributed to identified bots.

- **Access Pattern Deviation:**

  The mechanism continually monitors user access patterns based on the access policies defined in the Cloaking Wall Model. If an entity exhibits access patterns that deviate significantly from the established policies, it raises suspicion for potential bot activity.
- **Policy Adherence Assessment:**

  Each user, including potential bots, is assessed against the defined access policies. Legitimate Data Users are expected to follow the specified Long-Term Cloaking, Multi-Region based Cloaking, Time-Based Cloaking, and Geolocation-Based Cloaking rules. Any entity not adhering to these policies is flagged for further scrutiny.
- **Anomalous Access Timing:**

  Bots often operate on predefined schedules or exhibit unnatural timing patterns. The mechanism detects anomalous access timings that do not align with the specified time-based access policies. This helps identify bots attempting to access data outside permissible time windows.
- **Geolocation Inconsistencies:**

  The mechanism evaluates the geolocation of incoming requests in comparison to the Geolocation-Based Cloaking policies. If there are inconsistencies, such as requests originating from unexpected or restricted locations, it signals potential bot activity.
- **Rapid, Repetitive Access Attempts:**

  Bots typically attempt to access data rapidly and repetitively, following scripted sequences. The mechanism identifies patterns associated with rapid, repetitive access attempts, flagging entities that display such behaviour for further scrutiny.

6. Disguise Data Generator

The Malicious Data Generator Module, tailored for the Injection of Non-Compliant Data specifically targeted at users violating the access policy set by the admin, is a security component designed to simulate and generate data instances that intentionally breach established policies within the Cloud Consumer Web App. Malicious data is generated to cover a spectrum of access pattern violations.

6.1. Camouflage Data Disguise Technique

The Camouflage Data Disguise technique represents an advanced cryptographic approach that seamlessly integrates Chaffing and Winnowing with the formidable ChaCha20 encryption algorithm. This technique is strategically designed to provide disguised data as a countermeasure against unauthorized access, targeting both unpermitted users and potentially malicious bots.

- **Chaffing and Camouflage Process:**

  The module orchestrates a two-fold process: Chaffing, which adds decoy or chaff data, and Camouflage, which further disguises non-compliant data through additional obfuscation techniques. This combined approach ensures a multi-layered defense against unauthorized access.

- **ChaCha20 Encryption:**

  Genuine, chaff, and camouflaged data undergo encryption using the ChaCha20 algorithm. ChaCha20's strength in providing a secure and efficient encryption process contributes significantly to safeguarding the confidentiality of the disguised information.

- **Winnowing Process:**

  At the recipient's end, the winnowing process disentangles the genuine data from the chaff and camouflage layers. The ChaCha20 decryption algorithm, combined with the appropriate key, unveils the original, non-compliant data.

7. Monitoring and Auditing

In real-time, the module monitors and captures a spectrum of activities, including user interactions, data access, and system operations. It maintains a meticulous log of policy enforcement instances, detailing the invocation of access policies defined by the Cloaking Wall Model, along with outcomes and responses to policy violations. Furthermore, the module diligently tracks all data access and modification events, providing insights into who accessed specific data and the nature of their interactions, aiding in forensic analysis.

8. Alerts and Notification

Upon detection of a policy violation, the module triggers immediate alerts, swiftly notifying administrators through various channels such as email, SMS, in-app messages, or other preferred communication methods. These alerts are designed to provide administrators with detailed information about the nature of the policy violation, offering insights into unauthorized access attempts, data modifications beyond permitted levels, or breaches of temporal and geolocation constraints. This proactive alerting mechanism minimizes the latency between the occurrence of a policy violation and the notification to administrators, facilitating a rapid and targeted response.

6. SYSTEM TESTING AND IMPLEMENTATION

6.1. INTRODUCTION

The Cloaking Wall Model Project’s System Testing and Implementation phase is crucial for validating the system's functionality, performance, security, and reliability before its deployment in a live environment. This phase involves comprehensive testing strategies to ensure all components work as intended, identifying and resolving any issues or bugs, and ensuring the system meets the specified requirements and standards. Following successful testing, the implementation process will deploy the system into the operational environment, ensuring seamless integration, user training, and support for a smooth transition to live use. This project does encompass a rigorous testing and implementation process to guarantee a robust, secure, and user-friendly solution for cross-organizational data sharing in cloud environments.

6.2. STRATEGIC APPROACH TO SOFTWARE TESTING

System testing for the project would encompass various types of testing to ensure the robustness, functionality, and security of the system. Here are some key types of testing that would be relevant:

6.2.1. TYPES OF TESTING

1. **Functional Testing**: This type of testing verifies that each function of the system operates in accordance with the requirements specified in the design documents. It includes testing features like user authentication, data management, access control policies, and monitoring capabilities.
2. **Integration Testing**: Integration testing ensures that individual components of the system work together seamlessly as a whole. It validates interactions between different modules, APIs, and external systems, including the integration of the Cloaking Wall Model with the Cloud Consumer Web App.
3. **Performance Testing**: Performance testing assesses the responsiveness, scalability, and stability of the system under various load conditions. It includes testing the system's ability to handle concurrent users, process requests efficiently, and maintain acceptable response times.
4. **Security Testing**: Security testing evaluates the system's resilience against potential security threats and vulnerabilities. It includes testing for authentication mechanisms,

encryption protocols, access control measures, data masking techniques, and protection against common cyber threats like SQL injection and cross-site scripting (XSS).

5. **Usability Testing**: Usability testing focuses on assessing the system's user interface (UI) design, navigation flow, and overall user experience. It involves gathering feedback from end-users to identify any usability issues, accessibility concerns, or areas for improvement in terms of user interaction and interface design.
6. **Compatibility Testing**: Compatibility testing ensures that the system functions correctly across different platforms, browsers, devices, and operating systems. It verifies that the application is compatible with popular web browsers, mobile devices, and screen resolutions, ensuring a consistent user experience across diverse environments.
7. **Regression Testing**: Regression testing validates that recent code changes or enhancements do not introduce new defects or regressions in existing functionality. It involves retesting previously validated features and conducting automated regression test suites to ensure that the system remains stable and reliable after updates.
8. **Load Testing**: Load testing evaluates the system's performance under expected and peak load conditions. It involves simulating a high volume of concurrent users or data requests to assess how the system handles stress, scalability, and resource utilization.
9. **Stress Testing**: Stress testing assesses the system's behavior under extreme conditions beyond its normal operational capacity. It involves pushing the system to its limits to identify potential failure points, bottlenecks, or areas of weakness under heavy load, unexpected inputs, or adverse environmental conditions.
10. **Data Integrity Testing**: Data integrity testing verifies the accuracy, consistency, and reliability of data stored and processed by the system. It includes testing data validation rules, data manipulation operations, and data integrity constraints to ensure that data remains intact and error-free throughout its lifecycle.

6.3. TEST CASES

1. **User Authentication:**
    - **Test Case ID:** UA_TC_001
    - **Input:** Valid username and password with correct multi-factor authentication.
    - **Expected Result:** Successful authentication, granting access to the system.
    - **Actual Result:** User successfully authenticated, and access granted.
    - **Status:** Pass
2. **Dashboard Access:**
    - **Test Case ID:** DA_TC_001
    - **Input:** Successful authentication credentials.
    - **Expected Result:** Access to the dashboard module.
    - **Actual Result:** User gained access to the dashboard.
    - **Status:** Pass
3. **Cloaking Wall Model Integration:**
    - **Test Case ID:** CW_TC_001
    - **Input:** Accessing the dashboard module.
    - **Expected Result:** Integration of the Cloaking Wall Model into the dashboard.
    - **Actual Result:** Cloaking Wall Model successfully integrated.
    - **Status:** Pass
4. **Access Policy Configuration:**
    - **Test Case ID:** APC_TC_001
    - **Input:** Admin accessing the Access Policy Configurator.
    - **Expected Result:** Successful configuration of access policies.
    - **Actual Result:** Access policies configured without errors.
    - **Status:** Pass
5. **Data Management by Admins:**
    - **Test Case ID:** DMA_TC_001
    - **Input:** Admin uploading and managing data.
    - **Expected Result:** Successful organization and control of data.
    - **Actual Result:** Admin successfully managed and organized data.
    - **Status:** Pass
6. **User Management by Admins:**
    - **Test Case ID:** UMA_TC_001

- **Input:** Admin adding and managing users.
- **Expected Result:** Successful addition and management of users.
- **Actual Result:** Admin added and managed users without issues.
- **Status:** Pass

7. **Data Access by Data Users:**
    - **Test Case ID:** DAU_TC_001
    - **Input:** Data User accessing allocated data.
    - **Expected Result:** Successful access to designated data.
    - **Actual Result:** Data User accessed allocated data successfully.
    - **Status:** Pass
8. **Monitoring Data Access:**
    - **Test Case ID:** MDA_TC_001
    - **Input:** Admin or Data User accessing monitoring tools.
    - **Expected Result:** Insightful tracking of data access patterns.
    - **Actual Result:** Monitoring tools provided insightful data access patterns.
    - **Status:** Pass
9. **Bot Identification Mechanism:**
    - **Test Case ID:** BIM_TC_001
    - **Input:** Monitoring user access patterns.
    - **Expected Result:** Accurate identification of potential bot activity.
    - **Actual Result:** Bot Identification Mechanism accurately identified potential bot activity.
    - **Status:** Pass
10. **Disguise Data Generation:**
    - **Test Case ID:** DDG_TC_001
    - **Input:** Generating disguised data.
    - **Expected Result:** Successful simulation of non-compliant data.
    - **Actual Result:** Disguise Data Generator successfully simulated non-compliant data.
    - **Status:** Pass
11. **Monitoring and Auditing:**
    - **Test Case ID:** MA_TC_001
    - **Input:** Monitoring various system activities.

- **Expected Result:** Detailed logs of policy enforcement and anomaly detections.
- **Actual Result:** Monitoring and Auditing module maintained detailed logs as expected.
- **Status:** Pass

12. **Alerts and Notifications:**
    - **Test Case ID:** AN_TC_001
    - **Input:** Policy violation or security threat occurrence.
    - **Expected Result:** Immediate and informative alerts to administrators.
    - **Actual Result:** Alerts and Notifications module triggered alerts promptly.
    - **Status:** Pass

6.4. TEST REPORT

Introduction

The Test Report provides a comprehensive overview of the testing activities conducted for the Advanced Cloud Data Security System. This report aims to summarize the results of the testing phase, including the status of each test case and an overall assessment of the system's functionality.

Test Objective

The primary objective of the testing phase was to evaluate the performance, reliability, and functionality of the Advanced Cloud Data Security System. Specific goals included validating user authentication, assessing data access controls, and ensuring the successful integration of security features.

- To verify the functionality of user authentication and authorization mechanisms.
- To validate the implementation of access control policies based on the Cloaking Wall Model.
- To assess the system's ability to manage data securely and enforce data access policies effectively.
- To evaluate the system's resilience against security threats and vulnerabilities.
- To ensure that the system performs reliably under different scenarios and workloads.

Test Scope

The testing scope covered various modules within the system, including user authentication, dashboard access, Cloaking Wall Model integration, access policy configuration, data management, monitoring, and security features such as bot identification and disguise data generation.

- Testing of user authentication, including login, registration, and multi-factor authentication.
- Testing of access control mechanisms, such as role-based access control and policy enforcement.
- Testing of data management functionalities, including data upload, storage, retrieval, and deletion.
- Testing of security features, such as encryption, data masking, and intrusion detection.
- Performance testing to assess system responsiveness, scalability, and resource utilization.

Test Environment

The testing environment was set up with the following components:

Hardware:

- **Processor:** Intel Core i5-9400F CPU @ 2.90GHz
- **RAM:** 8GB DDR4
- **Storage:** 256GB SSD
- **Network Interface Card:** Gigabit Ethernet

Software:

- **Operating System:** Windows 10 Home
- **Web Browser:** Google Chrome, Mozilla Firefox
- **Database Management System:** MySQL 8.0
- **Web Server:** WampServer 3.2.0

Test Result

The following table outlines the results of each test case conducted during the testing phase:

- User Authentication: Successful authentication and access granted.
- Access Control: Access policies enforced based on Cloaking Wall Model.
- Data Management: Secure and efficient data management operations.
- Security Testing: Resilience against security threats and vulnerabilities.
- Performance Testing: Reliable performance under different scenarios.

Bug Report

A bug report is a document that details issues, defects, or unexpected behavior encountered in software during testing or usage. It typically includes information about the problem, steps to reproduce it, and any relevant system configurations. Bug reports are essential for developers to identify and fix issues in the software.

BID	TCID	Bug Description	Status	Output
BR_001	UA_TC_001	Authentication Failure	Closed	Error message displayed: "Invalid credentials."
BR_003	BIM_TC_001	Bot Identification Inaccuracy	Closed	Legitimate user flagged as potential bot; investigation ongoing.

Test Conclusion

The testing phase concludes with an overall positive assessment of the Advanced Cloud Data Security System. The majority of test cases have been successfully executed, meeting expected results. Identified issues were minimal and addressed promptly. The system is deemed ready for deployment with necessary enhancements.

6.5. IMPLEMENTATION

1. Frontend Implementation

Tools/Frameworks:

- **HTML, CSS, JavaScript**: For basic structure, styling, and client-side interactivity.
- **React.js or Angular**: For building dynamic user interfaces and managing component states.
- **Redux or Context API**: For state management, especially for handling user authentication and authorization states.
- **Axios or Fetch API**: For making HTTP requests to the backend API.

Steps:

1. Set up the project structure using a tool like Create React App or Angular CLI.
2. Create components for the user interface based on the wireframes and designs provided.
3. Implement user authentication screens such as login, registration, and password recovery.
4. Implement monitoring and alerting interfaces to provide real-time insights into resource performance and irregularities.

2. Backend Implementation

Tools/Frameworks:

- **Node.js, Express.js (or Django, Flask, Spring Boot)**: For building the backend server and handling HTTP requests.
- **JWT (JSON Web Tokens)**: For implementing authentication and generating secure tokens.
- **Passport.js (or Spring Security, Django REST Framework)**: For handling authentication strategies and middleware.
- **Database**: Choose a suitable database like PostgreSQL, MongoDB, or MySQL for storing user data, access policies, and resource information.

Steps:

1. Set up the backend server with Express.js (or your chosen framework) and define routes for handling authentication, data management, and monitoring.
2. Implement authentication middleware to verify JWT tokens and protect routes requiring authentication.
3. Develop controllers to handle business logic, including user authentication, data management, and access policy configuration.

3. Integration and Testing

- Integrate the frontend and backend components to ensure seamless communication between the client and server.
- Test the user interface for usability, responsiveness, and accessibility compliance across different devices and browsers.

4. Deployment and Monitoring

- Deploy the application to a cloud platform like AWS, Azure, or Google Cloud Platform using services like Elastic Beanstalk, Azure App Service, or Google App Engine.
- Set up monitoring tools like Amazon CloudWatch, Azure Monitor, or Google Cloud Monitoring to track application performance, resource usage, and security metrics.

By following these steps, you can systematically implement the Cloud Service Provider Web App, ensuring robust functionality, security, and scalability. It's essential to collaborate closely with stakeholders, adhere to best practices, and conduct thorough testing throughout the development process to deliver a high-quality and reliable application.

7. CONCLUSION

The Cloaking Wall Model, with features like Long-Term Cloaking and Geolocation-based Cloaking, ensures persistent confidentiality and global consistency. The Camouflage Data Disguise technique, integrating Chaffing and Winnowing with ChaCha20 encryption, adds an extra layer of defense. The Cloud Consumer Web App's modular design caters to both administrators and users, offering secure functionalities like user authentication, data management, and monitoring. The project's testing phase, outlined in the test report, demonstrates a rigorous approach to quality assurance. The innovative Bot Identification Mechanism, coupled with the Disguise Data Generator module, adds an intelligent layer to the security framework. By accurately identifying potential bot activity and simulating non-compliant data instances, the system actively responds to emerging threats. The Monitoring and Auditing modules, along with the immediate Alerts and Notifications system, empower administrators to maintain real-time oversight, respond promptly to policy violations, and uphold the integrity of the system. Thus the project provides a adaptive solution to evolving cloud data security challenges, aligning with the demands for secure and privacy-preserving cloud computing practices.

8. FUTURE ENHANCEMENT

The future evolution of the system holds exciting possibilities, with key areas of focus. Integrating machine learning algorithms stands out as a potential enhancement, enabling dynamic analysis of access patterns to adeptly respond to evolving security threats. Behavioural analytics is another avenue, offering a nuanced understanding of user behaviour to distinguish normal activities from potential risks. Additionally, exploring blockchain integration is on the horizon, aiming to enhance data integrity and transparency by leveraging the decentralized and tamper-resistant nature of blockchain technology. These enhancements collectively propel the system towards a more adaptive, context-aware, and secure future.

9. APPENDIX

9.1 SOURCE CODE

Packages

```
import os
import base64
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.fernet import Fernet
from Crypto import Random
from flask import Flask, render_template, Response, redirect, request, session, abort, url_for
import mysql.connector
import hashlib
import shutil
from datetime import date
import datetime
import math
from random import randint
from flask_mail import Mail, Message
from flask import send_file
```

Database Connection

```
mydb = mysql.connector.connect(
host="localhost",
user="root",
password="",
charset="utf8",
database="cloud_cloaking"
```

Login

```
def login():
msg=""
if request.method=='POST':
uname=request.form['uname']
pwd=request.form['pass']
```

```
cursor = mydb.cursor()
cursor.execute('SELECT * FROM data_owner WHERE owner_id = %s AND password = %s
&& status=1', (uname, pwd))
account = cursor.fetchone()
if account:
session['username'] = uname
return redirect(url_for('upload'))
else:
msg = 'Incorrect username/password!'
```

Data Owner Registration

```
def register():
msg=""
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM data_owner")
maxid = mycursor.fetchone()[0]
now = datetime.datetime.now()
rdate=now.strftime("%d-%m-%Y")
if maxid is None:
maxid=1
if request.method=='POST':
name=request.form['name']
mobile=request.form['mobile']
email=request.form['email']
city=request.form['city']
uname=request.form['uname']
pass1=request.form['pass']
cursor = mydb.cursor()
cursor.execute('SELECT count(*) FROM data_owner WHERE owner_id = %s ', (uname,))
cnt = cursor.fetchone()[0]
if cnt==0:
sql = "INSERT INTO data_owner(id,name,mobile,email,city,owner_id,password,reg_date)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid,name,mobile,email,city,uname,pass1,rdate)
cursor.execute(sql, val)
```

```
mydb.commit()
print(cursor.rowcount, "Registered Success")
msg="success"
else:
msg='fail'
```

Upload Files

```
def upload():
msg=""
act=""
if 'username' in session:
uname = session['username']
mycursor = mydb.cursor()
mycursor.execute('SELECT * FROM data_owner where owner_id=%s',(uname, ))
rr=mycursor.fetchone()
name=rr[1]
now = datetime.datetime.now()
rdate=now.strftime("%d-%m-%Y")
rtime=now.strftime("%H:%M")
if request.method=='POST':
description=request.form['description']
mycursor.execute("SELECT max(id)+1 FROM data_files")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
if 'file' not in request.files:
flash('No file part')
return redirect(request.url)
file = request.files['file']
file_type = file.content_type
if file.filename == '':
flash('No selected file')
return redirect(request.url)
if file:
fname = "F"+str(maxid)+file.filename
```

```
filename = secure_filename(fname)
file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
bsize=os.path.getsize("static/upload/"+filename)
fsize=bsize/1024
file_size=round(fsize,2)
ff=filename.split('.')
i=0
file_ext=''
for fimg in imgext:
if fimg==ff[1]:
file_ext=img[i]
break
else:
file_ext=img[0]
i+=1
sql = "INSERT INTO
data_files(id,owner_id,description,file_name,file_type,file_size,reg_date,reg_time,file_extens
ion) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid,uname,description,filename,file_type,file_size,rdate,rtime,file_ext)
mycursor.execute(sql,val)
mydb.commit()
msg="success"
```

Share Files

```
File_id=request.args.get("file_id")
uname=""
msg=""
act = request.args.get('act')
if 'username' in session:
uname = session['username']
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM data_owner where owner_id=%s",(uname,))
value = mycursor.fetchone()
name=value[1]
now = datetime.datetime.now()
```

```
rdate=now.strftime("%d-%m-%Y")
mycursor.execute("SELECT * FROM data_user where owner_id=%s",(uname,))
udata = mycursor.fetchall()
mycursor.execute("SELECT count(*) FROM data_user where owner_id=%s",(uname,))
ucnt = mycursor.fetchone()[0]
mycursor.execute("SELECT * FROM data_files where id=%s",(fid,))
fdata = mycursor.fetchone()
fname=fdata[3]
if request.method=='POST':
selected_users=request.form.getlist('uu[]')
for u1 in selected_users:
mycursor.execute("SELECT max(id)+1 FROM data_share")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO data_share(id, owner_id, file_id, username, share_type, share_date)
VALUES (%s, %s, %s, %s, %s, %s)"
val = (maxid, uname, file_id, u1, '1', rdate)
act="success"
mycursor.execute(sql, val)
mydb.commit()
```

ChaCha20 Encryption

```
import struct
def yield_chacha20_xor_stream(key, iv, position=0):
"""Generate the xor stream with the ChaCha20 cipher."""
if not isinstance(position, int):
raise TypeError
if position & ~0xffffffff:
raise ValueError('Position is not uint32.')
if not isinstance(key, bytes):
raise TypeError
if not isinstance(iv, bytes):
raise TypeError
if len(key) != 32:
```

```
raise ValueError
if len(iv) != 8:
raise ValueError
def rotate(v, c):
return ((v << c) & 0xffffffff) | v >> (32 - c)
def quarter_round(x, a, b, c, d):
x[a] = (x[a] + x[b]) & 0xffffffff
x[d] = rotate(x[d] ^ x[a], 16)
x[c] = (x[c] + x[d]) & 0xffffffff
x[b] = rotate(x[b] ^ x[c], 12)
x[a] = (x[a] + x[b]) & 0xffffffff
x[d] = rotate(x[d] ^ x[a], 8)
x[c] = (x[c] + x[d]) & 0xffffffff
x[b] = rotate(x[b] ^ x[c], 7)
ctx = [0] * 16
ctx[:4] = (1634760805, 857760878, 2036477234, 1797285236)
ctx[4 : 12] = struct.unpack('<8L', key)
ctx[12] = ctx[13] = position
ctx[14 : 16] = struct.unpack('<LL', iv)
while 1:
x = list(ctx)
for i in range(10):
quarter_round(x, 0, 4,  8, 12)
quarter_round(x, 1, 5,  9, 13)
quarter_round(x, 2, 6, 10, 14)
quarter_round(x, 3, 7, 11, 15)
quarter_round(x, 0, 5, 10, 15)
quarter_round(x, 1, 6, 11, 12)
quarter_round(x, 2, 7,  8, 13)
quarter_round(x, 3, 4,  9, 14)
for c in struct.pack('<16L', *(
(x[i] + ctx[i]) & 0xffffffff for i in range(16))):
yield c
ctx[12] = (ctx[12] + 1) & 0xffffffff
```

```
if ctx[12] == 0:
ctx[13] = (ctx[13] + 1) & 0xffffffff
def chacha20_encrypt(data, key, iv=None, position=0):
"""Encrypt (or decrypt) with the ChaCha20 cipher."""
if not isinstance(data, bytes):
raise TypeError
if iv is None:
iv = b'\0' * 8
if isinstance(key, bytes):
if not key:
raise ValueError('Key is empty.')
if len(key) < 32:
# TODO(pts): Do key derivation with PBKDF2 or something similar.
key = (key * (32 // len(key) + 1))[:32]
if len(key) > 32:
raise ValueError('Key too long.')
return bytes(a ^ b for a, b in
zip(data, yield_chacha20_xor_stream(key, iv, position)))
assert chacha20_encrypt(
b'Hello World', b'chacha20!') == b'\xeb\xe78\xad\xd5\xab\x18R\xe2O~'
assert chacha20_encrypt(
b'\xeb\xe78\xad\xd5\xab\x18R\xe2O~', b'chacha20!') == b'Hello World'
def run_tests():
import binascii
uh = lambda x: binascii.unhexlify(bytes(x, 'ascii'))
for i, (ciphertext, key, iv) in enumerate((
(uh('76b8e0ada0f13d90405d6ae55386bd28bdd219b8a08ded1aa836efcc8b770dc7da41597c5
157488d7724e03fb8d84a376a43b8f41518a11cc387b669'),
uh('0000000000000000000000000000000000000000000000000000000000000000'),
uh('0000000000000000')),
assert chacha20_encrypt(b'\0' * len(ciphertext), key, iv) == ciphertext
print('Test %d OK.' % i)
```

Set Geo Location

```
Def shrea_geolocation:
```

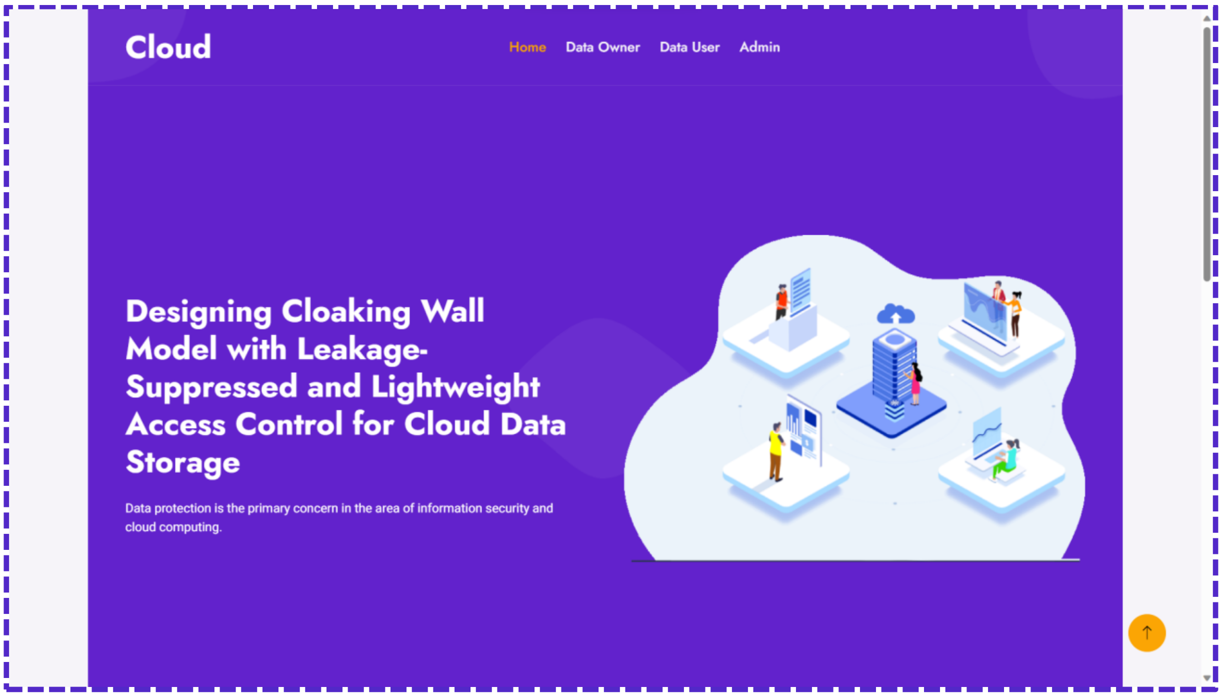
```
mycursor.execute('SELECT * FROM data_user where username=%s',(uname, ))
rr=mycursor.fetchone()
name=rr[1]
owner=rr[2]
ff=open("static/geo.txt","r")
loc=ff.read()
ff.close()
mycursor.execute("SELECT count(*) FROM data_files f,data_share s where s.fid=f.id &&
s.username=%s",(uname,))
c1 = mycursor.fetchone()[0]
if c1>0:
mycursor.execute("SELECT * FROM data_files f,data_share s where s.fid=f.id &&
s.username=%s",(uname,))
dat = mycursor.fetchall()
for d1 in dat:
status=''
if d1[13]==1:
status='1'
if d1[13]==2:
lat1=latitude.split('.')
lt1=lat1[0]
lt11=lat1[1]
lt2=lt11[0:4]
lon1=longitude.split('.')
lo1=lon1[0]
lo2=lon1[1]
mycursor.execute("SELECT * FROM share_location where share_id=%s",(d1[9],))
d33 = mycursor.fetchall()
for d3 in d33:
mycursor.execute("SELECT * FROM geo_location where id=%s",(d3[4],))
d4 = mycursor.fetchone()
g1=d4[2]
geo_location=g1.split('new google.maps.LatLng(')
g21=''.join(geo_location)
g22=g21.split('), ')
g23='-'.join(g22)
g24=g23.split('-')
gn=len(g24)-1
i=0
while i<gn:
f1=l1.split('.')
geo1=f1[0]
f2=f1[1]
f3=f2[0:4]
gloc1.append(f3)
```

Set Time and Date

```
def share_time:
date_st=''
time_st=''
days_st=''
#between date
sdate=d1[15]
edate=d1[16]
sd1=sdate.split('-')
ed1=sdate.split('-')
import datetime
sdd = datetime.datetime(int(sd1[2]), int(sd1[1]),int(sd1[0]))
cdd = datetime.datetime(int(cd1[2]), int(cd1[1]),int(cd1[0]))
edd = datetime.datetime(int(ed1[2]), int(ed1[1]),int(ed1[0]))
if sdd<cdd<edd:
date_st='1'
else:
date_st='1'
#days
dys=d1[19]
dy=dys.split(',')
x=0
from datetime import datetime
```

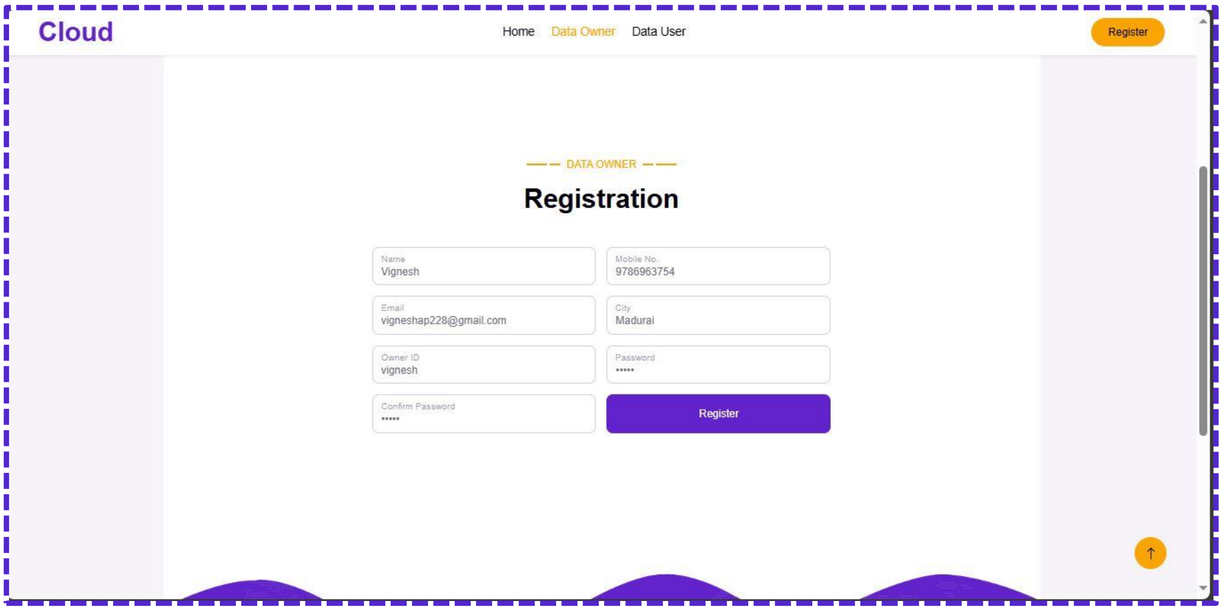
```
dty = datetime.now()
ddy=dty.strftime('%A')
ddr=['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday']
i=0
for ddr1 in ddr:
i+=1
if ddr1==ddy:
break
cdy=str(i)
for dy1 in dy:
if cdy==dy1:
x+=1
if x>0:
days_st='1'
def file_download():
fid = request.args.get('fid')
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM data_files where id=%s",(fid,))
value = mycursor.fetchone()
path="static/upload/"+value[3]
return send_file(path, as_attachment=True)
```

9.2 SCREENSHOTS



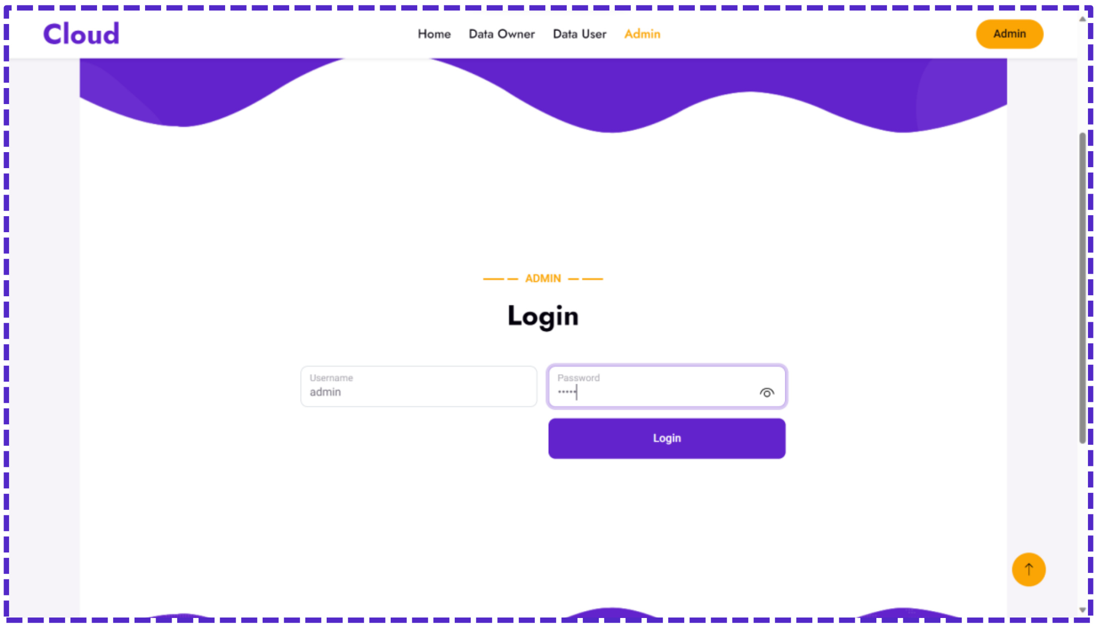
Designing Cloaking Wall Model with Leakage-Suppressed and Lightweight Access Control for Cloud Data Storage

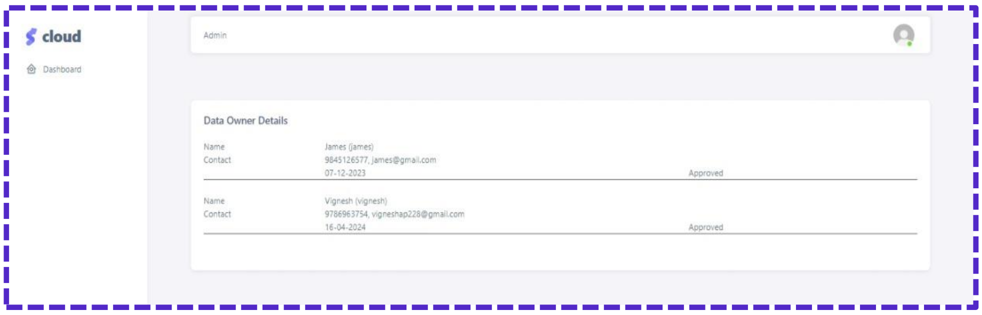
Data protection is the primary concern in the area of information security and cloud computing.



Registration

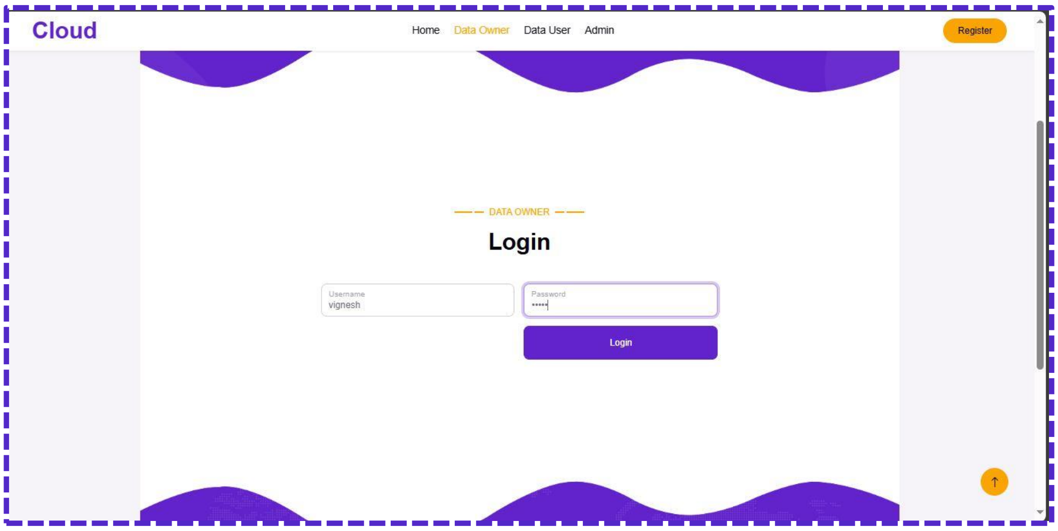
Name: Vignesh	Mobile No.: 9786963754
Email: vigneshap228@gmail.com	City: Madurai
Owner ID: vignesh	Password: •••••
Confirm Password: •••••	Register

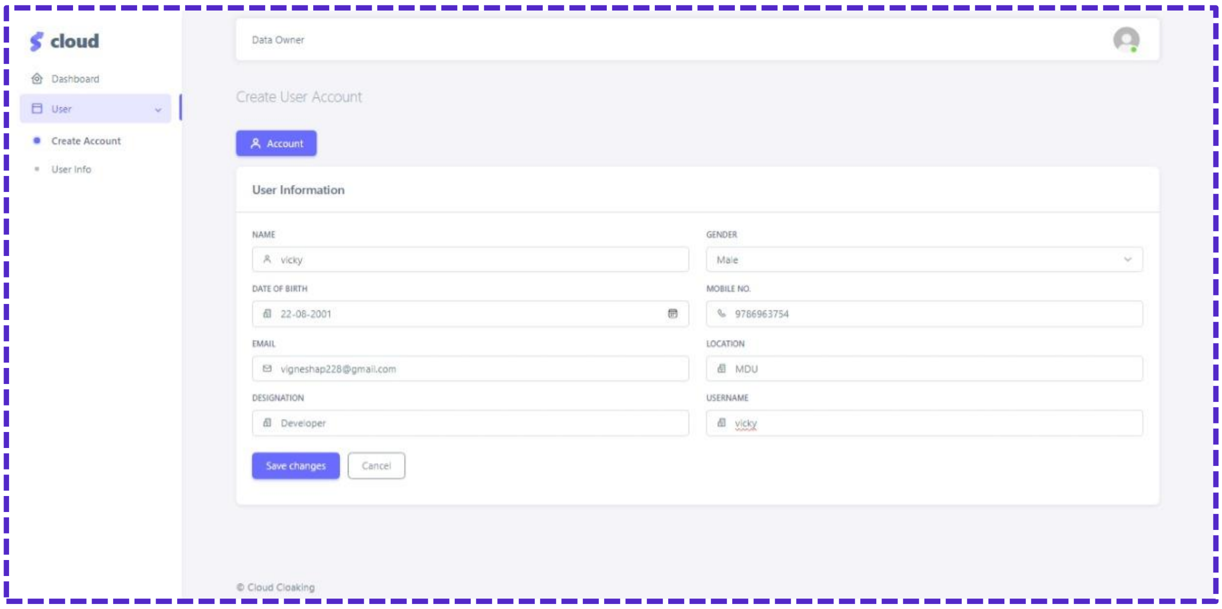




Data Owner Details

Name	James (james)	
Contact	9845126577, james@gmail.com	
	07-12-2023	Approved
Name	Vignesh (vignesh)	
Contact	9786963754, vigneshap228@gmail.com	
	16-04-2024	Approved

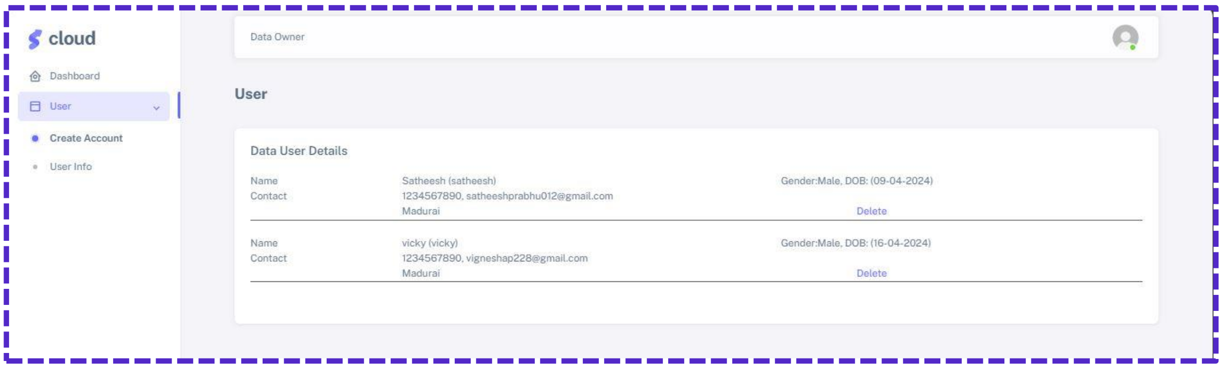




Create User Account

User Information

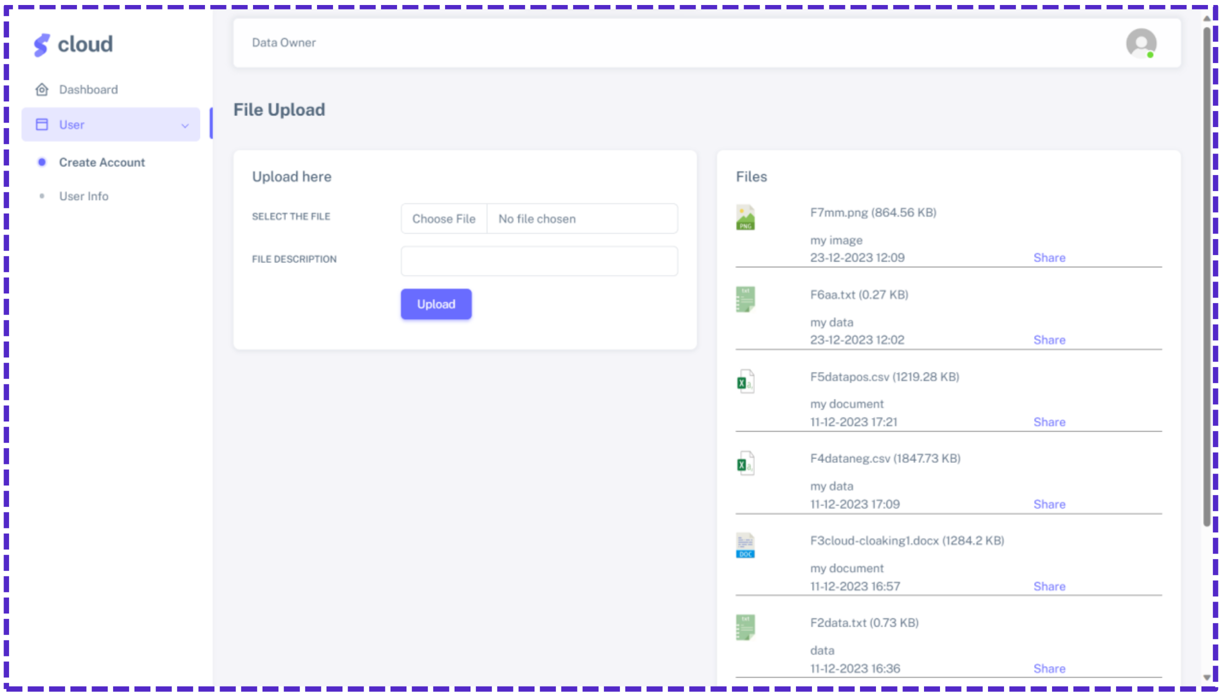
NAME: vicky
GENDER: Male
DATE OF BIRTH: 22-08-2001
MOBILE NO.: 9786963754
EMAIL: vigneshap228@gmail.com
LOCATION: MDU
DESIGNATION: Developer
USERNAME: vicky



User

Data User Details

Name	Satheesh (satheesh)	Gender:Male, DOB: (09-04-2024)
Contact	1234567890, satheeshprabhu012@gmail.com	
	Madurai	Delete
Name	vicky (vicky)	Gender:Male, DOB: (16-04-2024)
Contact	1234567890, vigneshap228@gmail.com	
	Madurai	Delete



File Upload

Upload here

SELECT THE FILE

FILE DESCRIPTION

Files

F7mm.png (864.56 KB)
my image
23-12-2023 12:09 Share

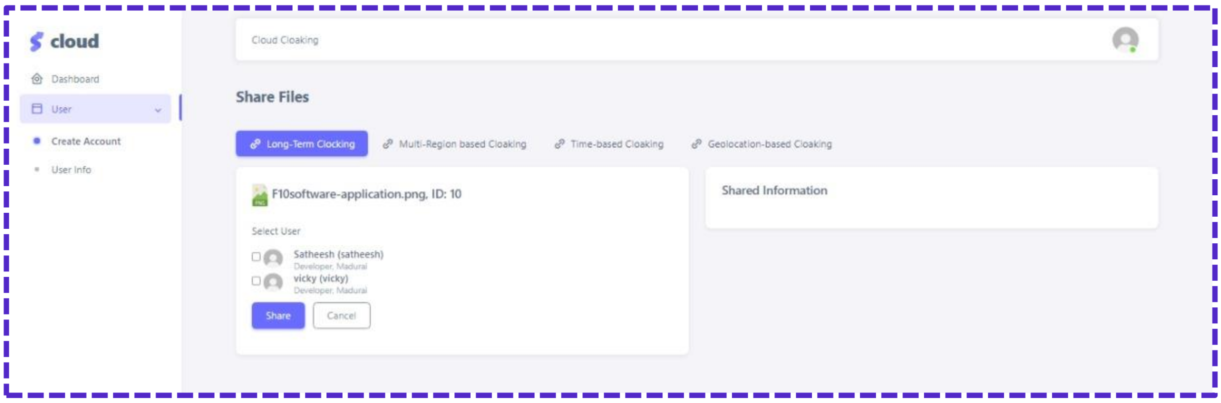
F6aa.txt (0.27 KB)
my data
23-12-2023 12:02 Share

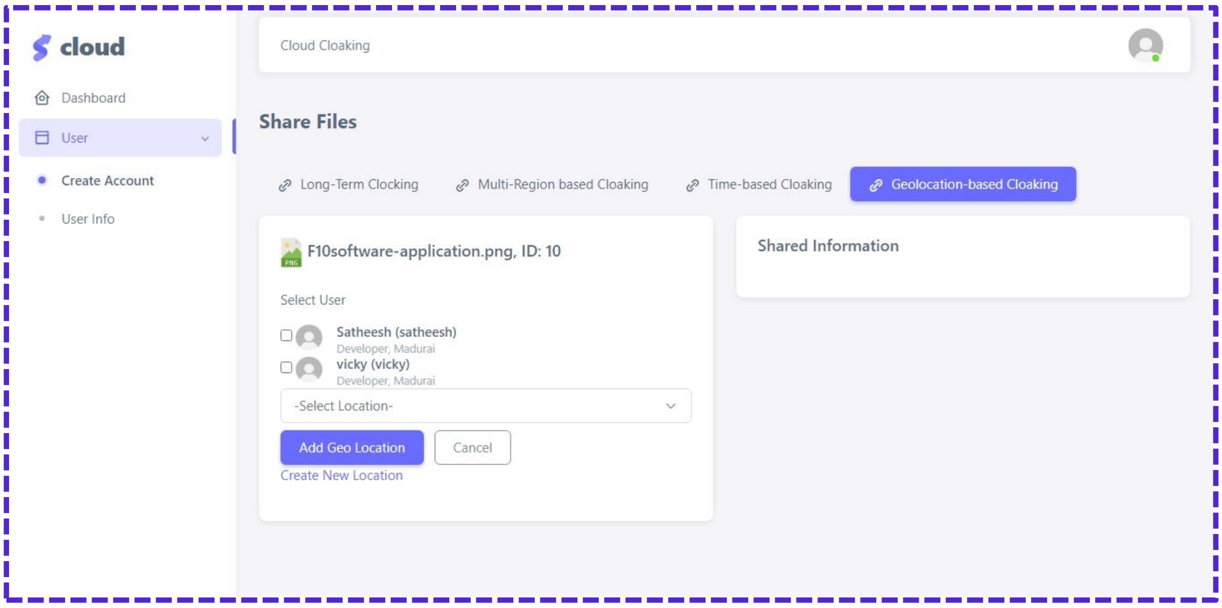
F5datapos.csv (1219.28 KB)
my document
11-12-2023 17:21 Share

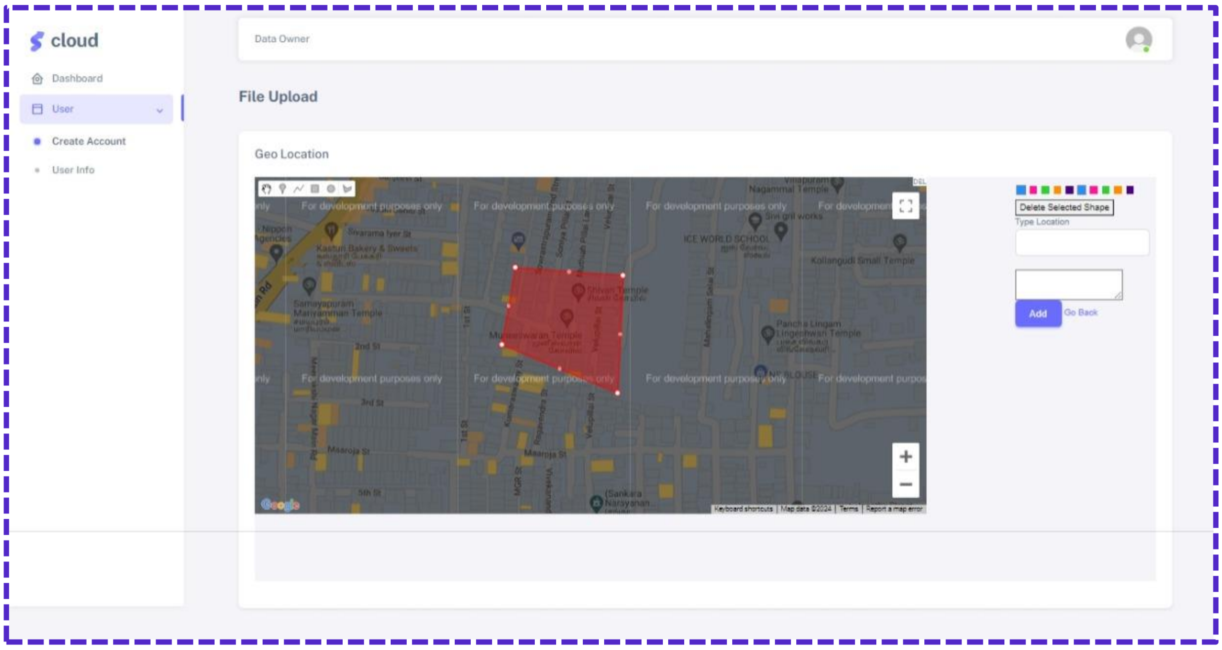
F4dataneg.csv (1847.73 KB)
my data
11-12-2023 17:09 Share

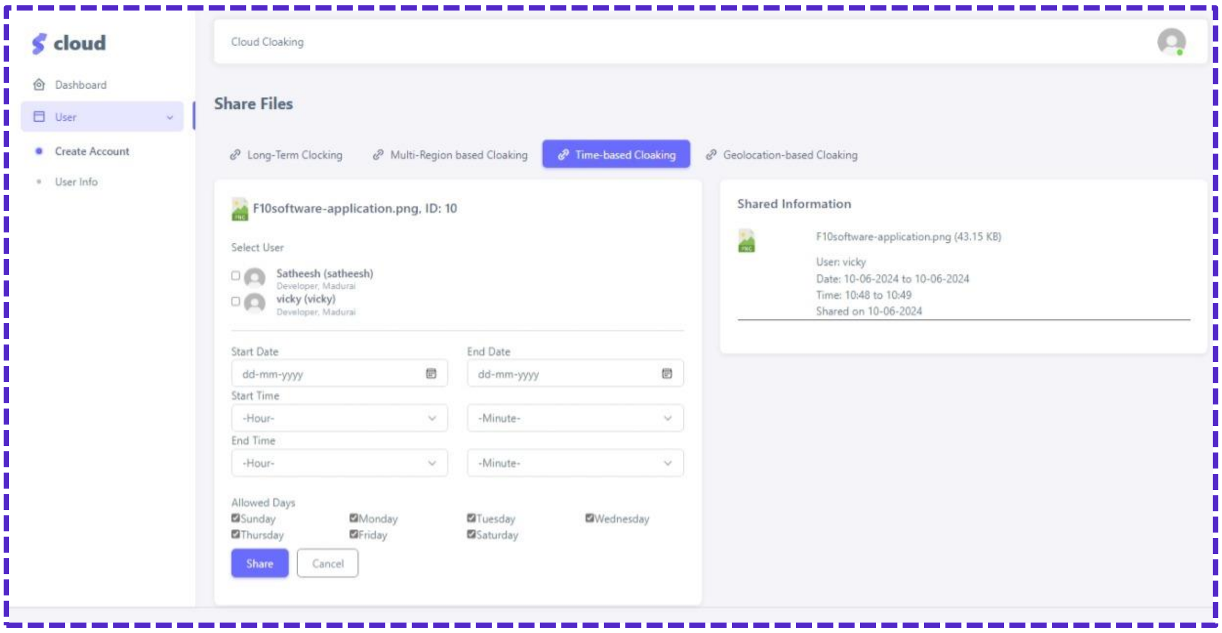
F3cloud-cloaking1.docx (1284.2 KB)
my document
11-12-2023 16:57 Share

F2data.txt (0.73 KB)
data
11-12-2023 16:36 Share


cloud
Dashboard
User
Create Account
User Info
Cloud Cloaking
Share Files
Long-Term Clocking
Multi-Region based Cloaking
Time-based Cloaking
Geolocation-based Cloaking
F10software-application.png, ID: 10
Select User
Satheesh (satheesh)
Developer, Madurai
vicky (vicky)
Developer, Madurai
Share
Cancel
Shared Information


cloud
Dashboard
User
Create Account
User Info
Cloud Cloaking
Share Files
Long-Term Clocking
Multi-Region based Cloaking
Time-based Cloaking
Geolocation-based Cloaking
F10software-application.png, ID: 10
Select User
Satheesh (satheesh)
Developer, Madurai
vicky (vicky)
Developer, Madurai
-Select Location-
Add Geo Location
Cancel
Create New Location
Shared Information


cloud
Dashboard
User
Create Account
User Info
Data Owner
File Upload
Geo Location
For development purposes only
Kasturi Bakery & Sweets
Samayapuram Mariyamman Temple
Nagammal Temple
ICE WORLD SCHOOL
Kallangudi Small Temple
Pancha Lingam Lingeshwar Temple
Delete Selected Shape
Type Location
Add
Go Back



Share Files

F10software-application.png, ID: 10

Select User

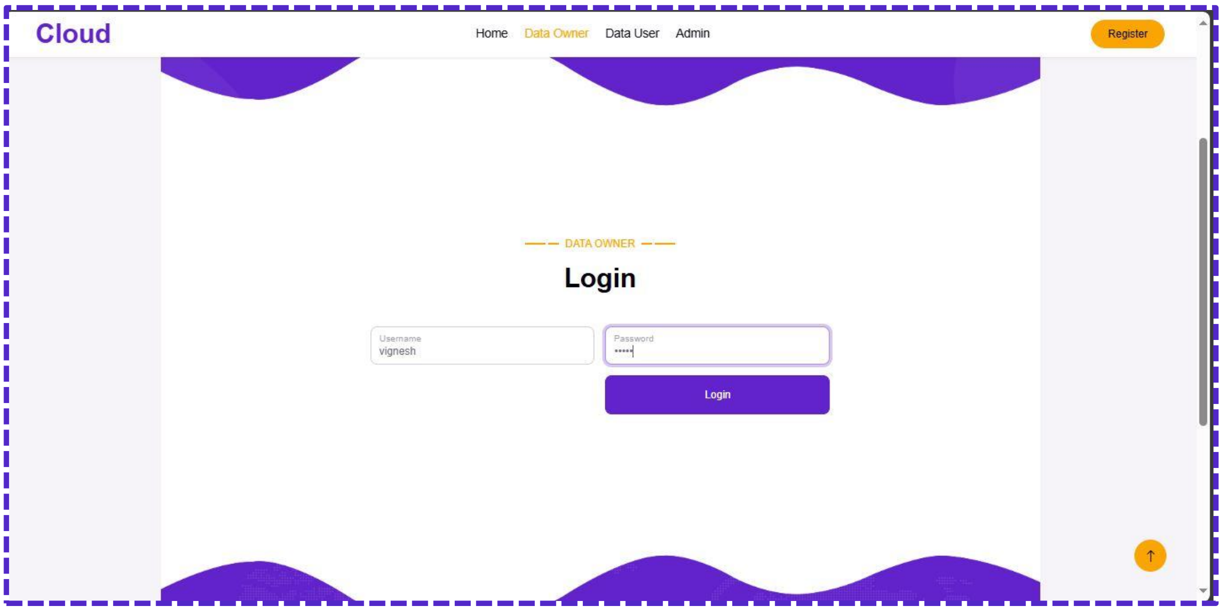
- Satheesh (satheesh) Developer, Madurai
- vicky (vicky) Developer, Madurai

Start Date: dd-mm-yyyy
End Date: dd-mm-yyyy
Start Time: -Hour- -Minute-
End Time: -Hour- -Minute-

Allowed Days
☑Sunday ☑Monday ☑Tuesday ☑Wednesday
☑Thursday ☑Friday ☑Saturday

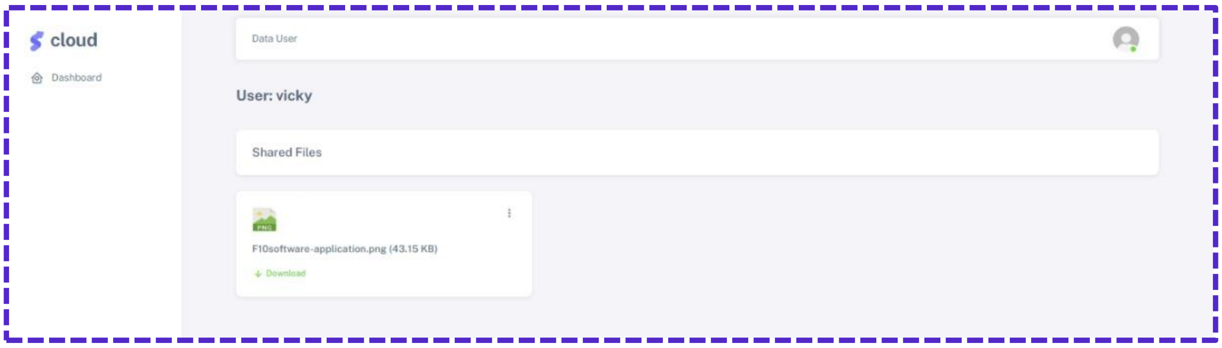
Shared Information

F10software-application.png (43.15 KB)
User: vicky
Date: 10-06-2024 to 10-06-2024
Time: 10:48 to 10:49
Shared on 10-06-2024



Login

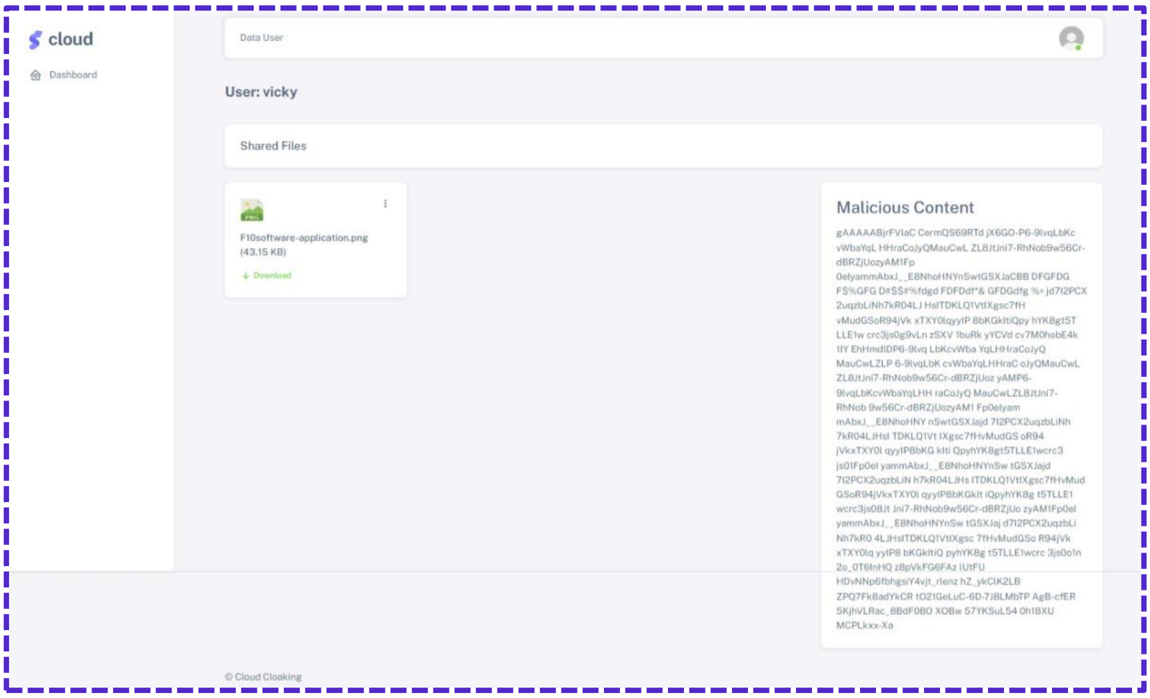
Username: vignesh
Password: •••••



User: vicky

Shared Files

F10software-application.png (43.15 KB)



User: vicky

Shared Files

F10software-application.png
(43.15 KB)

Malicious Content

gAAAAABjrFVIaC CermQS69RTd jX6GO-P6-9lvqLbKc
vWbaYqL HHraCoJyQMauCwL ZL8JtJni7-RhNob9w56Cr-
dBRZjUozyAM1Fp
OelyammAbxJ__E8NhoHNYnSwtGSXJaCBB DFGFDG
F$%GFG D#$$#%fdgd FDFDdf*& GFDGdfg %^ jd7I2PCX
2uqzbLiNh7kR04LJ HsITDKLQ1VtIXgsc7fH
vMudGSoR94jVk xTXY0lqyyIP 8bKGkItiQpy hYK8gt5T
LLE1w crc3js0g9vLn zSXV 1buRk yYCVd cv7M0hobE4k
1IY EhHmdIDP6-9lvq LbKcvWba YqLHHraCoJyQ
MauCwLZLP 6-9lvqLbK cvWbaYqLHHraC oJyQMauCwL
ZL8JtJni7-RhNob9w56Cr-dBRZjUoz yAMP6-
9lvqLbKcvWbaYqLHH raCoJyQ MauCwLZL8JtJni7-
RhNob 9w56Cr-dBRZjUozyAM1 Fp0elyam
mAbxJ__E8NhoHNY nSwtGSXJajd 7I2PCX2uqzbLiNh
7kR04LJHsI TDKLQ1Vt IXgsc7fHvMudGS oR94
jVkxTXY0l qyyIP8bKG kIti QpyhYK8gt5TLLE1wcrc3
js01Fp0el yammAbxJ__E8NhoHNYnSw tGSXJajd
7I2PCX2uqzbLiN h7kR04LJHs ITDKLQ1VtIXgsc7fHvMud
GSoR94jVkxTXY0l qyyIP8bKGkIt iQpyhYK8g t5TLLE1
wcrc3js08Jt Jni7-RhNob9w56Cr-dBRZjUo zyAM1Fp0el
yammAbxJ__E8NhoHNYnSw tGSXJaj d7I2PCX2uqzbLi
Nh7kR0 4LJHsITDKLQ1VtIXgsc 7fHvMudGSo R94jVk
xTXY0lq yyIP8 bKGkItiQ pyhYK8g t5TLLE1wcrc 3js0o1n
2o_0T6InHQ z8pVkFG6FAz lUtFU
HDvNNp6fbhgsiY4vjt_rlenz hZ_ykCIK2LB
ZPQ7Fk8adYkCR tO21GeLuC-6D-7J8LMbTP AgB-cfER
5KjhVLRac_8BdF08O XOBw 57YKSuL54 0h18XU
MCPLkxx-Xa

© Cloud Cloaking

9.3 BIBLIOGRAPHY

1. S. Qi, W. Wei, J. Wang, S. Sun, L. Rutkowski, T. Huang, et al., "Secure data deduplication with dynamic access control for mobile cloud storage", *IEEE Trans. Mobile Comput.*, pp. 1-18, 2023.
2. Y. Chen, J. Li, C. Liu, J. Han, Y. Zhang and P. Yi, "Efficient attribute based server-aided verification signature", *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3224-3232, Nov. 2022.
3. P. Patil and M. Sangeetha, "Blockchain-based decentralized KYC verification framework for banks", *Proc. Comput. Sci.*, vol. 215, pp. 529-536, Jan. 2022.
4. X. Li, T. Liu, C. Chen, Q. Cheng, X. Zhang and N. Kumar, "A lightweight and verifiable access control scheme with constant size ciphertext in edge-computing-assisted IoT", *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19227-19237, Oct. 2022.
5. P. Sanchol, S. Fugkeaw and H. Sato, "A mobile cloud-based access control with efficiently outsourced decryption", *Proc. 10th IEEE Int. Conf. Mobile Cloud Comput. Services Eng. (MobileCloud)*, pp. 1-8, Aug. 2022.
6. S. Fugkeaw, "A lightweight policy update scheme for outsourced personal health records sharing", *IEEE Access*, vol. 9, pp. 54862-54871, 2021.
7. J. Sun, D. Chen, N. Zhang, G. Xu, M. Tang, X. Nie, et al., "A privacy-aware and traceable fine-grained data delivery system in cloud-assisted healthcare IIoT", *IEEE Internet Things J.*, vol. 8, no. 12, pp. 10034-10046, Jun. 2021.
8. J. Gao, H. Yu, X. Zhu and X. Li, "Blockchain-based digital rights management scheme via multiauthority ciphertext-policy attribute-based encryption and proxy re-encryption", *IEEE Syst. J.*, vol. 15, no. 4, pp. 5233-5244, Dec. 2021.
9. Y. Lin, J. Li, X. Jia and K. Ren, "Multiple-replica integrity auditing schemes for cloud data storage", *Concurrency Comput. Pract. Exper.*, vol. 33, no. 7, pp. 1, Apr. 2021.
10. S. Wang, H. Wang, J. Li, H. Wang, J. Chaudhry, M. Alazab, et al., "A fast CP-ABE system for cyber-physical security and privacy in mobile healthcare network", *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4467-4477, Jul. 2020.
11. Z. Li, W. Li, Z. Jin, H. Zhang and Q. Wen, "An efficient ABE scheme with verifiable outsourced encryption and decryption", IEEE Access, vol. 7, pp. 29023-29037, 2019.
12. Q. Li, Y. Tian, Y. Zhang, L. Shen and J. Guo, "Efficient privacy-preserving access control of mobile multimedia data in cloud computing", *IEEE Access*, vol. 7, pp. 131534-131542, 2019.

9.4. REFERENCES

9.4.1. BOOK REFERENCES

1. "Python Crash Course" by Eric Matthes
2. "Fluent Python" by Luciano Ramalho
3. "Automate the Boring Stuff with Python" by Al Sweigart
4. "Learning MySQL" by Seyed M.M. Tahaghoghi and Hugh E. Williams
5. "High-Performance MySQL" by Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko
6. "MySQL Cookbook" by Paul DuBois
7. "Bootstrap 4 in Action" by Jamie Munro
8. "Bootstrap Reference Guide" by Aravind Shenoy
9. "Mastering Bootstrap 4" by Benjamin Jakobus.

9.4.2. WEB LINK REFERENCES

1. Mozilla Developer Network (MDN): https://developer.mozilla.org/
2. W3Schools: https://www.w3schools.com/
3. Stack Overflow: https://stackoverflow.com/
4. CSS-Tricks: https://css-tricks.com/
5. A List Apart: https://alistapart.com/