A Major Project Report on

DATA LEAKAGE DETECTION IN CLOUD COMPUTING

Submitted in partial fulfillment for the
degree of Bachelor of Technology in
Computer Science and Technology

Submitted by
(54) Dhrithi Shetty
(55) Nidhi Shetty
(56) Sneha Shinde

Under the guidance of
Prof. Monica Charate

Usha Mittal Institute Of Technology
S.N.D.T. WOMEN'S UNIVERSITY
MUMBAI-400049
2024-2025

CERTIFICATE

This is to certify that **Dhrithi Shetty, Nidhi Shetty, Sneha Shinde** have completed Major Project report Phase-III(A) on the topic **"Data Leakage Detection in Cloud Computing"** satisfactorily for the Bachelor's Degree in **Computer Science and Technology** under the guidance of **Prof. Monica Charate** during the year 2024-25 as prescribed by S.N.D.T Women's University, Mumbai.

Guide
Prof.Monica Charate

HOD
Prof. Kumud Wasnik

Principal
Dr. Yogesh Nerkar

Examiner 1

Examiner 2

ACKNOWLEDGEMENT

We would like to express our sincere appreciation to everyone who gave their invaluable assistance and support in this project. We are highly indebted to **Prof. Monica Charate** for her guidance and constant supervision as well as for providing necessary information regarding the project also for her support in this project. Without her help, this project would not have been possible.

We would also like to thank our Head of Department, **Prof. Kumud Wasnik** for her insights in shaping the direction and content of this report. We would also like to extend our gratitude to our classmates who provided valuable feedback and encouragement throughout the process.

Finally, we would like to express our appreciation to our families for their unwavering support and understanding during the long hours and late nights spent working on this project.

Thank you all for your contributions, support, and encouragement.

(54) Dhrithi Shetty
(55) Nidhi Shetty
(56) Sneha Shinde

ABSTRACT

In today's digital era, the Internet plays a vital role in how we communicate, collaborate, and share information. It has become a crucial platform for exchanging data, accessible to those with the right permissions. This shift towards online sharing has underscored the critical need for dependable cloud storage solutions.

Cloud storage offers efficient access to information but also introduces risks of unauthorized access and data leakage. Safeguarding data stored in the cloud, ensuring it remains secure and accessible only to authorized users, is a paramount challenge.

Our project addresses this challenge by developing a system to securely store and monitor data leakage in cloud-shared data. Through advanced encryption techniques, the system ensures data protection before storage, allowing decryption and access only to authorized users. Emphasizing traceability, the system tracks and identifies instances of data leakage, crucial for accountability and proactive security measures.

In summary, this project aims to provide a robust solution for secure cloud storage. By encrypting data, restricting access, and focusing on data leakage prevention, our system aims to protect shared data effectively in the cloud, maintaining a balance between accessibility and stringent security standards.

Contents

Chapter 1

INTRODUCTION

Cloud computing has become an essential tool for organizations, providing scalable and flexible data storage. However, storing sensitive information in the cloud brings increased risks of unauthorized access and data leaks. While encryption offers protection, it is often not enough to identify the source of leaks or prevent misuse in shared environments. This highlights the need for more advanced security measures that ensure both data protection and accountability.

This project proposes a system that integrates watermarking and fake object insertion to address these challenges. Watermarking embeds unique identifiers within files, allowing leaks to be traced back to specific users. This ensures that every file accessed in the cloud can be linked to the user responsible, adding an extra layer of accountability. Meanwhile, fake object insertion creates decoy files designed to lure and detect malicious users, helping identify threats before they cause harm.

The system also features real-time monitoring, enabling administrators to view access logs and receive alerts for suspicious activity. When anomalies are detected, admins can investigate user patterns, check for interactions with fake objects, and trace leaks using the embedded watermarks. This multi-faceted approach enhances the ability to identify and respond to potential threats quickly and efficiently.

By combining watermarking and fake object insertion, the system offers a robust solution for cloud security. It protects sensitive data from unauthorized access, provides the ability to trace leaks, and ensures prompt action in response to security breaches. This comprehensive approach strengthens data integrity and confidentiality in the cloud.

Chapter 2

LITERATURE SURVEY

2.1 Technical Papers

A literature survey, also known as a literature review, is an essential part of research that involves a comprehensive analysis and synthesis of previously published research on a specific topic. The purpose of a literature survey is to identify gaps in knowledge, assess the state of research, and provide a foundation for further research.

In our review of data leakage detection systems, several important papers were analyzed, with Paper 1 serving as a base research paper for our project. This paper integrates AES-128 encryption with digital watermarking, addressing significant issues in detecting data leaks. Its comprehensive approach to embedding watermarks and securing data through encryption has been pivotal in shaping our project's direction. Additional papers were reviewed to complement our approach, each offering valuable insights or techniques. The table below summarizes the contributions and limitations of these papers:

No.	Paper Name	Author(s)	Remarks
1	Secure Data Storage and Sharing Techniques for Data Protection in Cloud Environments: A Systematic Review, Analysis, and Future Directions (2022)	Ishu Gupta Ashutosh Kumar Singh Chung-Nan Lee Rajkumar Buyya	This paper details a data leak detection approach using AES-128 encryption and unique watermarks to trace leaks. While effective in identifying guilty agents, it faces scalability and performance challenges.
2	Avoiding the data leakage and providing privacy to data in Networking (2016)	Madhavi Suryawanshi, Prof . Sarita Patil	The authors propose a system that enhances data leak detection security by introducing encryption and techniques to protect sensitive information during transmission and storage. AES is used to encrypt data. This algorithm is favoured for its speed and security.
3	A Survey on Data Breach Challenges in Cloud Computing Security: Issues and Threats(2017)	R.Barona, E.A.Mary Anita	The paper provides a broad overview of data breaches in cloud computing, discussing issues like data integrity and threat analysis. However, it is mostly theoretical and lacks practical solutions or empirical data
4	Dynamic Data Leakage using Guilty Agent Detection over Cloud (2017)	Govinda. K, Divya Joseph	The authors discuss various data leakage detection methods, including watermarking techniques, and time-stamping. They propose a model that integrates these techniques to enhance data security in cloud computing.
5	An Analysis of Data Leakage and Prevention Techniques in Cloud Environments (2015)	T. Brindha, R. S. Shaji	The paper reviews data leakage issues and proposes preventive measures, including strategies. It discusses guilty agents, however, it is generic and not specific to cloud environments and lacks real-world implementations
6	Data Leakage Detection (2011)	Panagiotis Papadimitriou, Hector Garcia-Molina	This paper outlines a framework for detecting data leaks using strategic data allocation methods and additional techniques to enhance detection. It effectively balances leak detection with minimal disruption but faces challenges related to probabilistic accuracy and assumptions about user behavior.

2.2 Existing System

The existing systems, despite their implemented features, show significant limitations in addressing modern data security challenges. One system utilizes the Bell-LaPadula model for access control, which relies on a rigid, hierarchical security classification structure. While this model ensures that subjects can only read or write data within their clearance levels, it is not equipped to handle the dynamic and evolving nature of modern data leakage threats. The system uses AES-128 encryption, which is now considered less secure compared to more robust standards like AES-256, making it vulnerable to sophisticated attacks. Moreover, its watermarking technique, which embeds simple organizational logos or ASCII values into documents, is relatively basic. This method lacks advanced protection, making it susceptible to tampering and attacks, potentially compromising the integrity of the data.

Another system focuses on embedding watermark data by modifying the intensity values of image pixels to conceal information. The watermarking method itself is not sufficiently secure, as it lacks a traceable identification mechanism to pinpoint the source of data leakage or unauthorized access. Additionally, like the first system, it uses AES-128 encryption, which is less secure than more advanced encryption techniques available today. This system also does not incorporate a comprehensive authentication or malicious activity detection framework, leaving it open to unauthorized access, data tampering, and other security vulnerabilities, especially in multi-user environments. Both systems fall short in providing scalable, robust, and traceable data security solutions that are critical for modern cloud environments where data confidentiality and integrity are paramount.

Chapter 3

PROBLEM STATEMENT

In cloud computing, safeguarding sensitive data from unauthorized access and potential leaks is a growing challenge. While encryption offers protection, it lacks mechanisms to trace data leaks back to individual users or detect malicious activities within shared environments. Organizations face difficulties in identifying breaches early and tracking the source of leaks, which can lead to significant data loss and security risks. This project addresses these concerns by proposing a system that combines watermarking and fake object insertion to detect unauthorized access, trace data leaks to specific users, and alert administrators in real-time for timely intervention.

Chapter 4

PROPOSED SYSTEM

The proposed system for detecting data leaks in a shared cloud environment leverages encryption, watermarking, and continuous monitoring to enhance data security. When a user uploads a file, it undergoes AES-256 encryption to ensure secure storage in the cloud. A unique watermark is embedded into the file each time a user accesses it, associating the watermark with the specific user. The watermark metadata, which tracks user identity and access time, is stored in a secure database for traceability. This allows for easy identification of the user in the event of a leak, even if the file is shared outside the intended environment.

In addition to watermarking, the system injects fake objects into the cloud storage as decoys to detect unauthorized access attempts. Each time a user accesses a file, the system verifies their authorization, decrypts the file if they are legitimate, and embeds a unique watermark to monitor their actions. The system continuously monitors access logs to detect any abnormal or suspicious behavior. If such behavior is identified, the admin is alerted. The admin can then investigate by reviewing the access logs, checking if the user has interacted with fake objects, and tracing any potential leak via the watermark metadata. Appropriate actions, such as blocking access or alerting the user, can be taken to prevent further data leaks. This approach ensures that any data shared in the environment can be traced back to the user, enhancing overall cloud security.

4.1 Algorithm

4.1.1 AES-256 Encryption for Data Security in Cloud Computing

Objective: To secure files in a cloud environment by encrypting them using AES-256, ensuring that only authorized users can decrypt and access the data. Steps:

1: Input:

- Original plaintext file P
- 256-bit encryption key K (32 bytes)

2: Key Expansion:

- Use the 256-bit key K to generate a series of round keys using the AES key expansion algorithm. These round keys will be used in each round of encryption.

3: Initial Round:

- AddRoundKey: XOR the first round key with the plaintext P.

4: Main Rounds (14 Rounds for AES-256):

- **SubBytes:** Substitute each byte of the block using a predefined substitution box (S-Box), which provides a non-linear transformation.
- **ShiftRows:** Shift each row of the block to the left by different offsets. Row 0 is left unchanged, row 1 is shifted by 1 byte, row 2 by 2 bytes, and row 3 by 3 bytes.
- **MixColumns (For all rounds except the last):** Mix the columns of the block by applying a linear transformation to each column, ensuring diffusion.
- **AddRoundKey:** XOR the block with the next round key derived from the key expansion.

5: Final Round:

- **SubBytes:** Apply byte substitution using the S-Box.
- **ShiftRows:** Apply row shifting as before.
- **AddRoundKey:** XOR the block with the final round key.

6: Output:

- The final encrypted file C (ciphertext) is produced after completing the 14 rounds of AES-256.

7. Decryption:

- The reverse process is applied using the same 256-bit key K, where each step is performed in reverse (Inverse ShiftRows, Inverse SubBytes, Inverse MixColumns, and AddRoundKey) to recover the original plaintext P from the ciphertext C.

4.1.2 Watermarking Using Embedded Data in Files

Objective: To embed and extract a unique watermark in a file, enabling traceability and detection of unauthorized access or leaks.

Steps:

1. Convert Identifier to Binary:

- Convert the unique identifier (UID) into its binary representation. This involves breaking down the UID into its constituent parts and converting each part to binary format.

2. Format Binary Data:

- Convert the binary data into a format suitable for embedding within the file. This often involves translating binary values into special characters or patterns that can be embedded without affecting the primary content of the file.

3. Embed Watermark into File:

- Append or insert the formatted binary data into the file. Ensure that the watermark is added in a way that does not interfere with the file's main content or functionality.

4. Extract and Interpret Watermark:

- Read the relevant part of the file to extract the embedded watermark data. Convert the extracted data back into binary format for further processing.

5. Decode Binary Data to Identifier:

- Convert the extracted binary data back into the original UID format. This involves parsing the binary data and translating it back into the original identifier.

4.1.3 Fake Object Insertion

Objective: To create and insert dummy or decoy files into a system to detect unauthorized access or behavior. Steps:

1. Define Fake Object:

- Create a template or decide the content for the fake object. This could be random data or specific patterns designed to act as a decoy.

2. Generate Fake Object:

- Create a new file with the defined content. Ensure it mimics the structure or type of real files to blend in.

3. Insert Fake Object:

- Place the fake object in the desired location within the file system. This could be alongside real files or within directories that are likely to be accessed.

4. Monitor Access:

- Track access to the fake objects. This can involve logging access attempts or monitoring changes to the fake files.

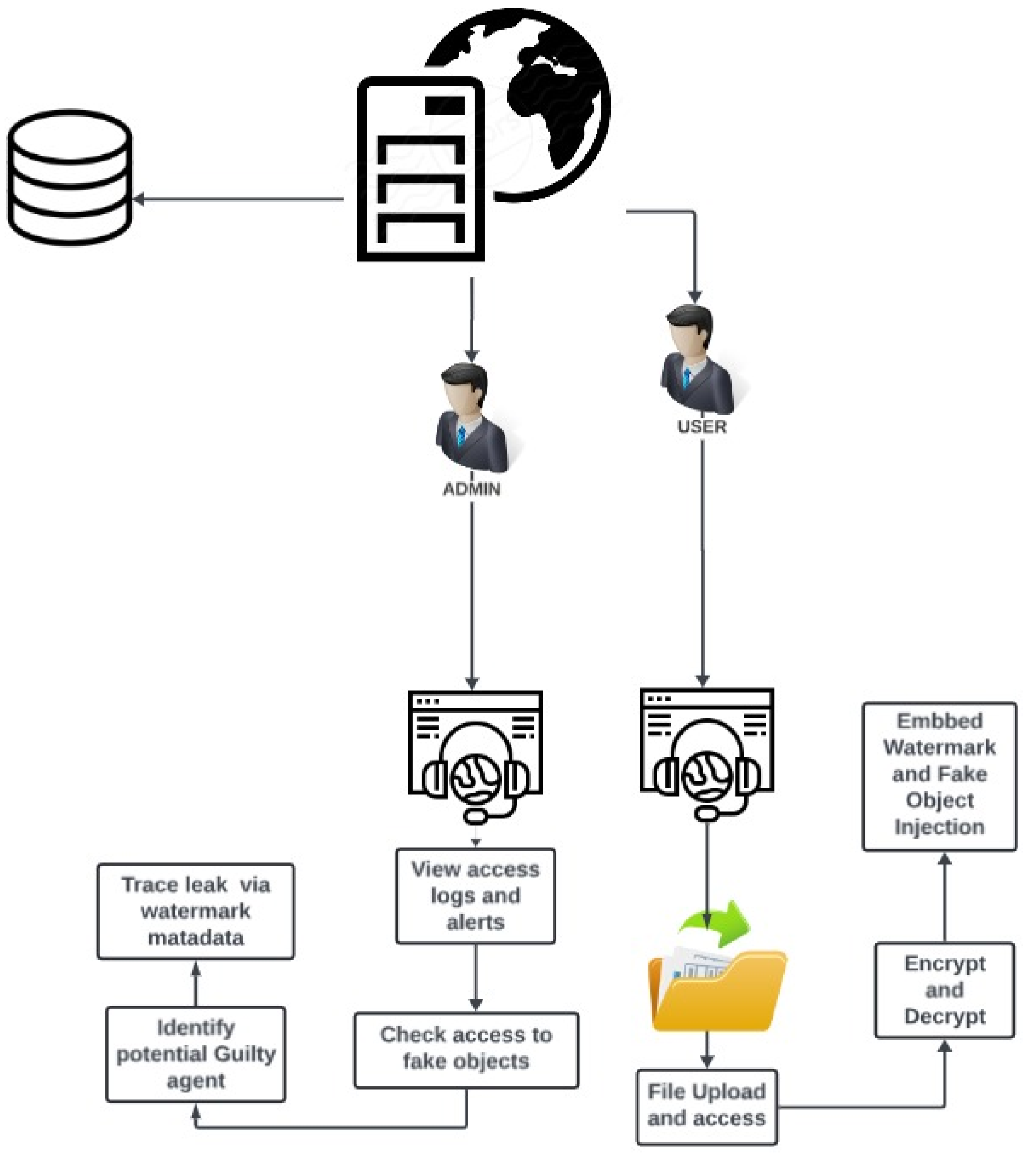
5. Analyze Interactions:

- Review the logs or monitor data to determine if any unauthorized access or suspicious behavior involves the fake objects.

6. Take Action:

- Based on the analysis, take appropriate actions, such as alerting administrators or securing the system to prevent further unauthorized access.

4.2 Architecture



ADMIN
USER
Trace leak via watermark matadata
View access logs and alerts
Embbed Watermark and Fake Object Injection
Identify potential Guilty agent
Check access to fake objects
Encrypt and Decrypt
File Upload and access

Fig 1: Architecture of Data Leakage Detection in Cloud Computing

4.3 Workflow

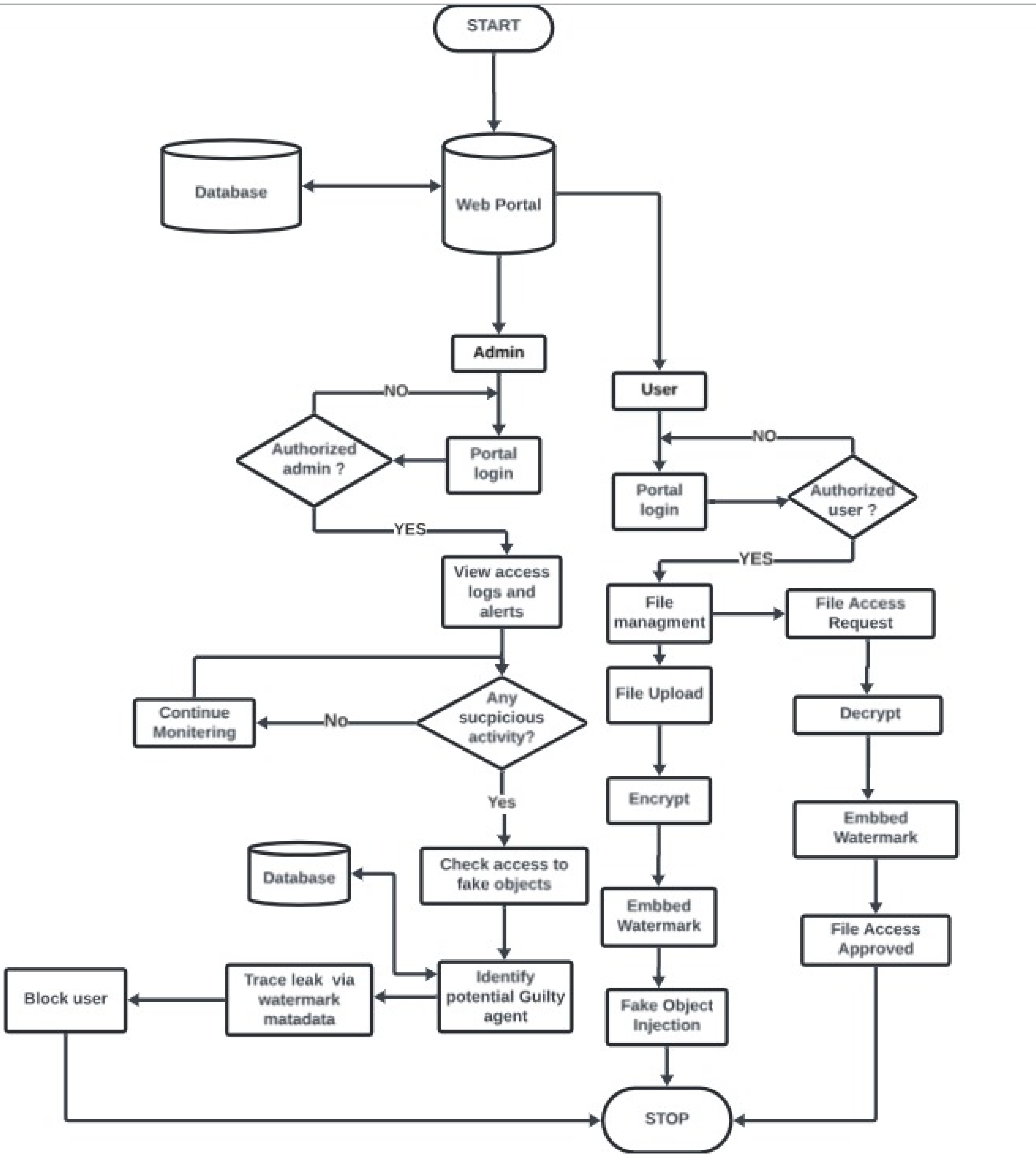


Fig 2: Workflow diagram of Data Leakage Detection in Cloud Computing

Chapter 5

HARDWARE & SOFTWARE REQUIREMENTS

5.1 Hardware Requirements

Minimum hardware requirements are dependent on the particular software being developed. Applications that need to store large arrays/objects in memory will require more RAM whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor:

- Operating system : Windows(windows 11), Linux(5.15 kernel)
- Processor : Intel Core i3 or higher
- RAM : 4 GB (minimum), 8 GB or higher recommended
- Speed: 2.4 GHz or faster
- Hard disk : minimum 250GB

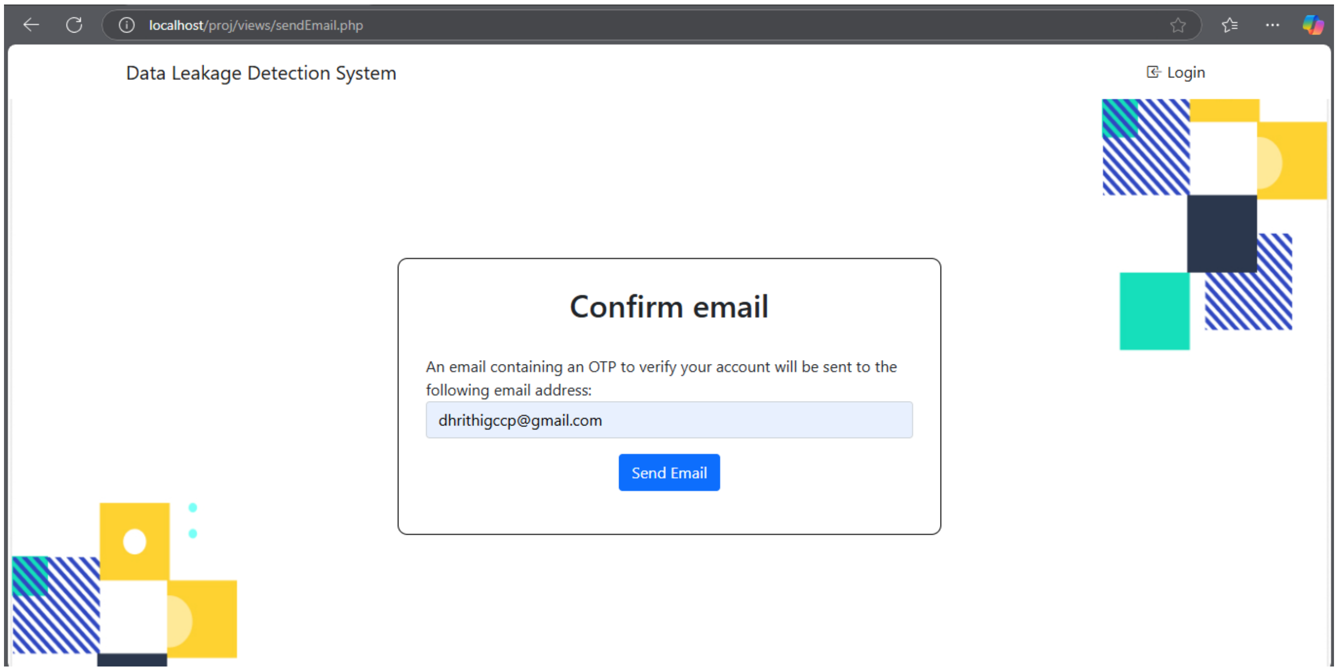
5.2 Software Requirements

The functional requirements and overarching description documentation encompass aspects such as the product's perspective, features, operating system, operating environment, graphics requisites, design constraints, and user documentation.For the development environment, the following options are recommended:

- Operating System : Windows 10 or higher
- Application Server : XAMPP, Apache
- Frontend : HTML, CSS, JavaScript
- Backend : PHP
- Database : MySQL

Chapter 6

IMPLEMENTATION STATUS



localhost/proj/views/sendEmail.php

Data Leakage Detection System

Login

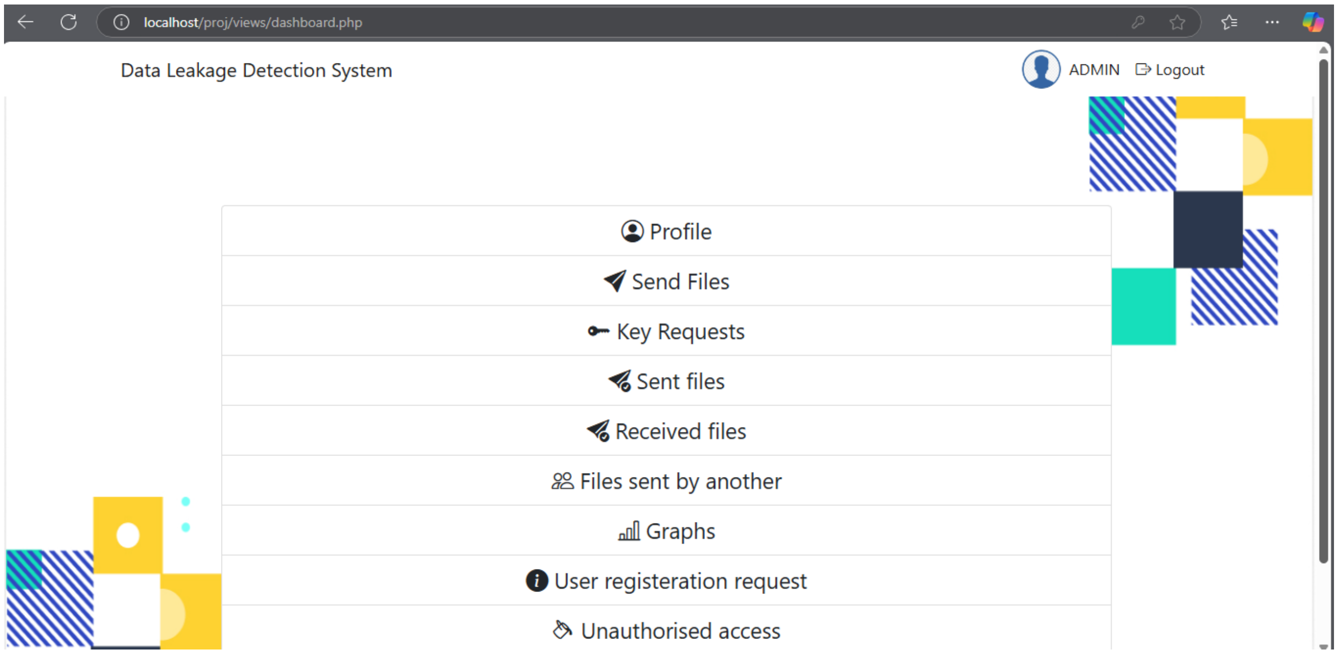
Confirm email

An email containing an OTP to verify your account will be sent to the following email address:

dhrithigccp@gmail.com

Send Email

Fig 6(a): Sign Up Page



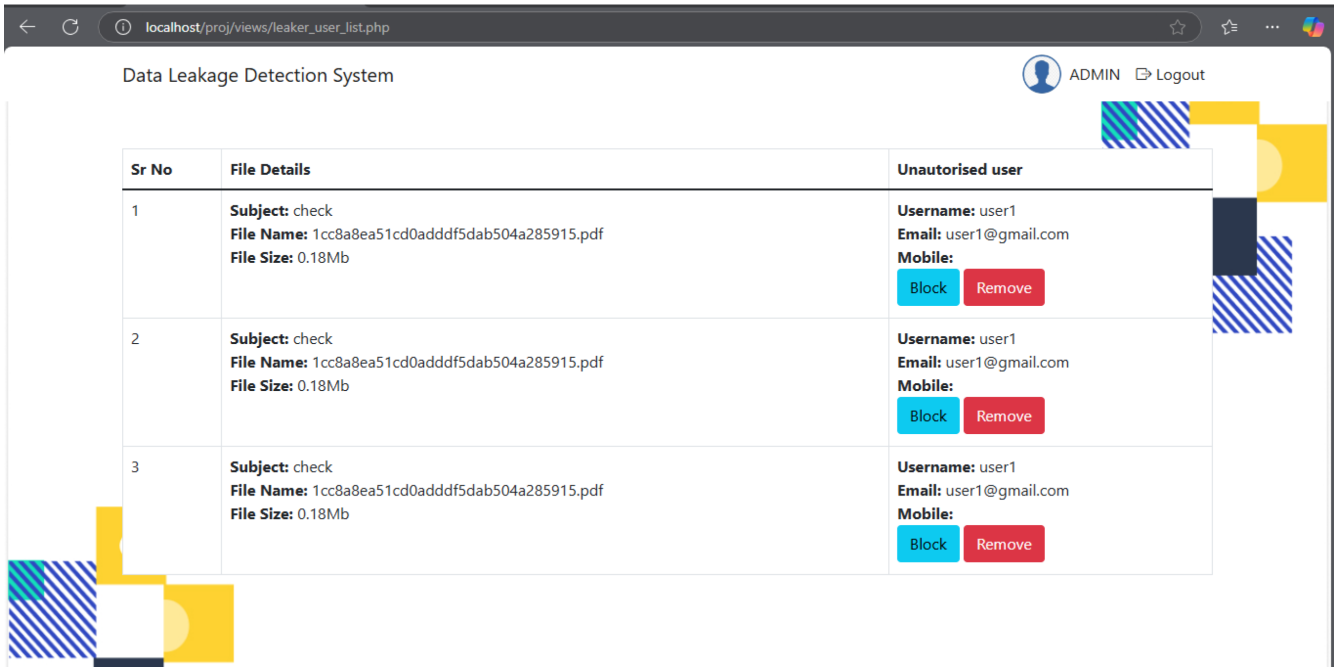
localhost/proj/views/dashboard.php

Data Leakage Detection System

ADMIN Logout

- Profile
- Send Files
- Key Requests
- Sent files
- Received files
- Files sent by another
- Graphs
- User registeration request
- Unauthorised access

Fig 6(b): Admin Dashboard



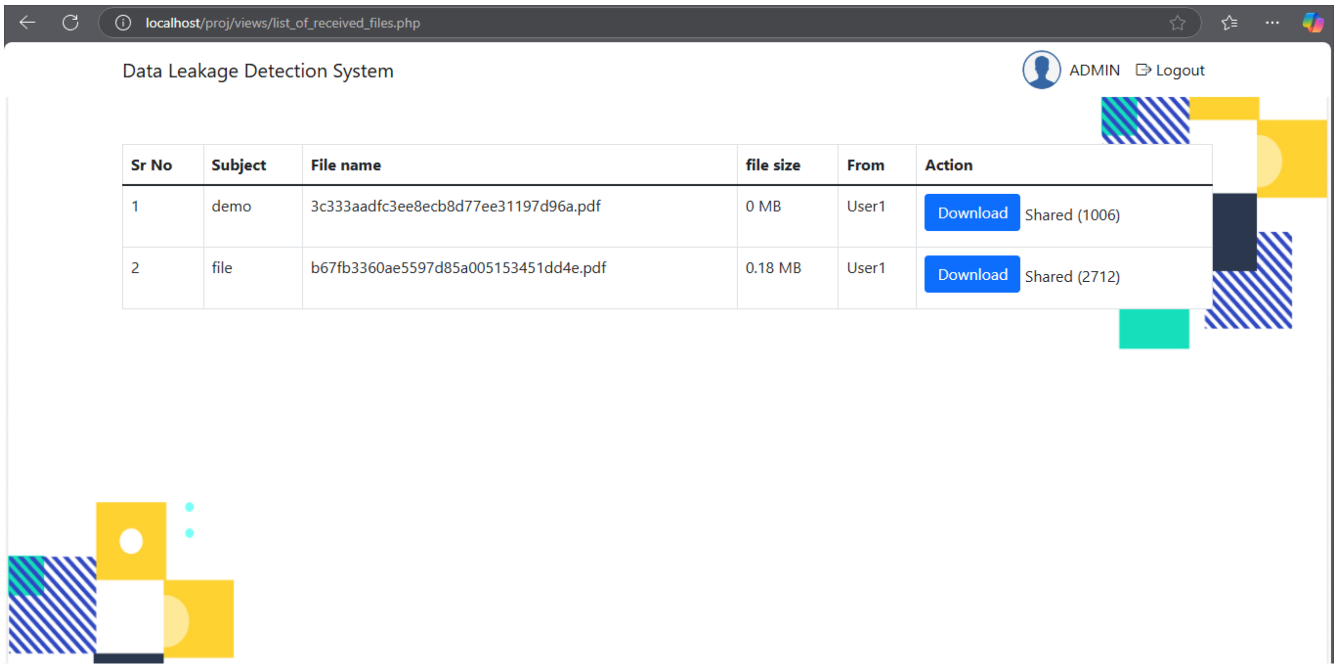
localhost/proj/views/leaker_user_list.php

Data Leakage Detection System

ADMIN Logout

Sr No	File Details	Unautorised user
1	**Subject:** check **File Name:** 1cc8a8ea51cd0adddf5dab504a285915.pdf **File Size:** 0.18Mb	**Username:** user1 **Email:** user1@gmail.com **Mobile:** Block Remove
2	**Subject:** check **File Name:** 1cc8a8ea51cd0adddf5dab504a285915.pdf **File Size:** 0.18Mb	**Username:** user1 **Email:** user1@gmail.com **Mobile:** Block Remove
3	**Subject:** check **File Name:** 1cc8a8ea51cd0adddf5dab504a285915.pdf **File Size:** 0.18Mb	**Username:** user1 **Email:** user1@gmail.com **Mobile:** Block Remove

Fig 6(c): Leaker Information



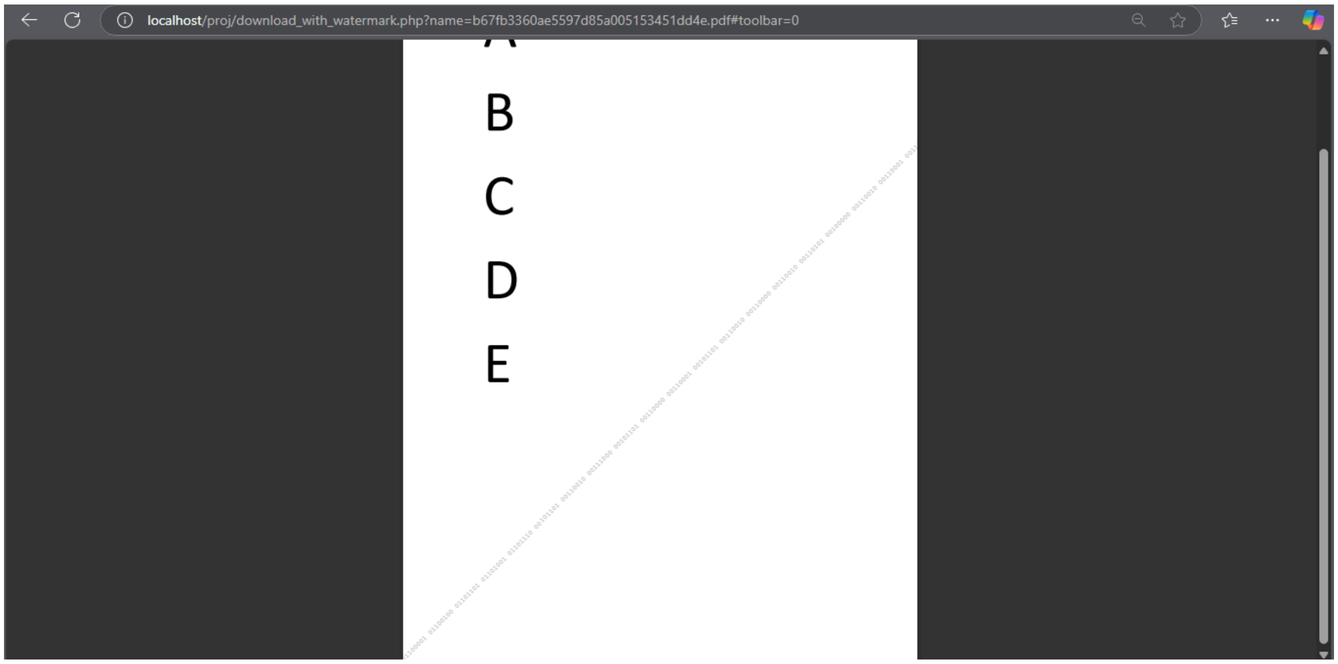
localhost/proj/views/list_of_received_files.php

Data Leakage Detection System

ADMIN Logout

Sr No	Subject	File name	file size	From	Action
1	demo	3c333aadfc3ee8ecb8d77ee31197d96a.pdf	0 MB	User1	Download Shared (1006)
2	file	b67fb3360ae5597d85a005153451dd4e.pdf	0.18 MB	User1	Download Shared (2712)

Fig 6(d): Files


localhost/proj/download_with_watermark.php?name=b67fb3360ae5597d85a005153451dd4e.pdf#toolbar=0

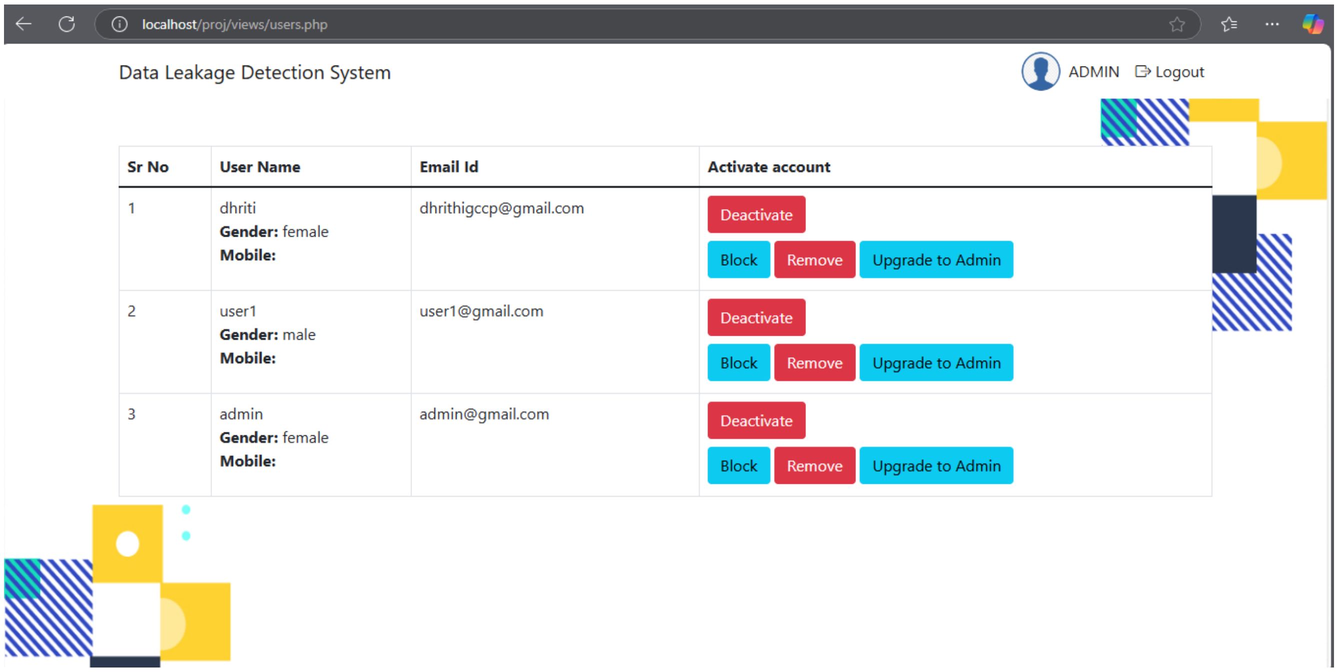
B

C

D

E

Fig 6(e): Watermarked File


localhost/proj/views/users.php

Data Leakage Detection System

ADMIN Logout

Sr No	User Name	Email Id	Activate account
1	dhriti **Gender:** female **Mobile:**	dhrithigccp@gmail.com	Deactivate Block Remove Upgrade to Admin
2	user1 **Gender:** male **Mobile:**	user1@gmail.com	Deactivate Block Remove Upgrade to Admin
3	admin **Gender:** female **Mobile:**	admin@gmail.com	Deactivate Block Remove Upgrade to Admin

Fig 6(f): User List

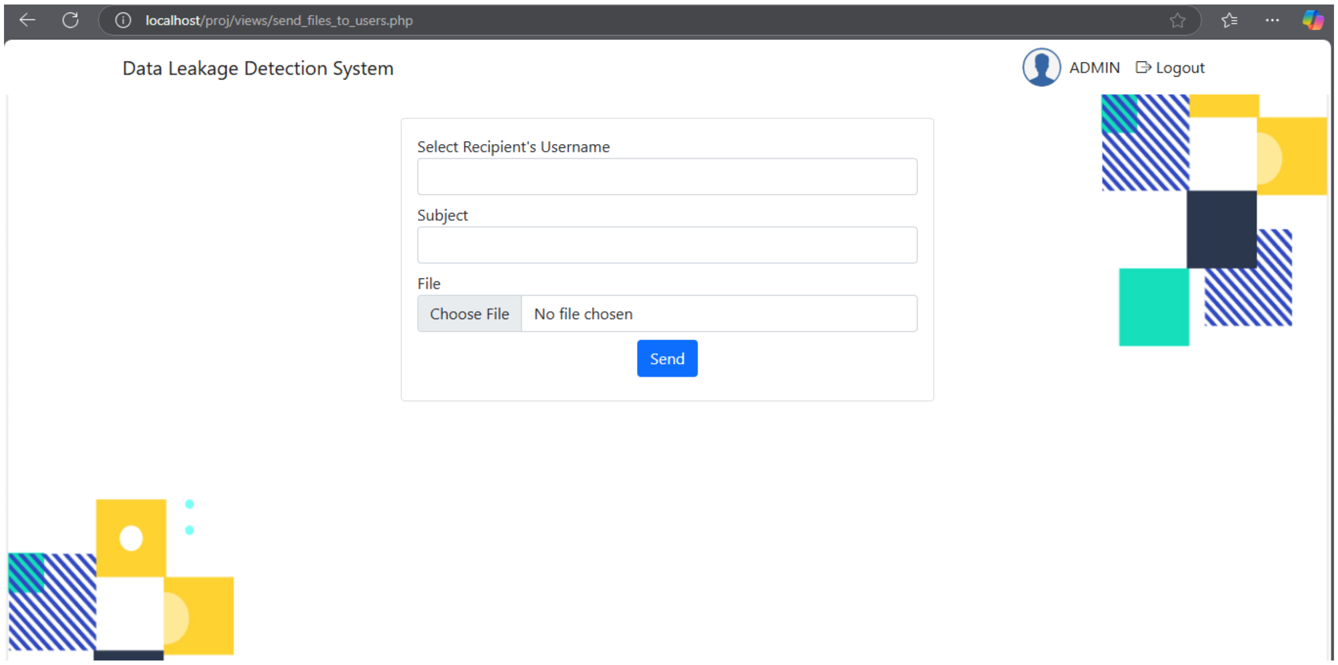
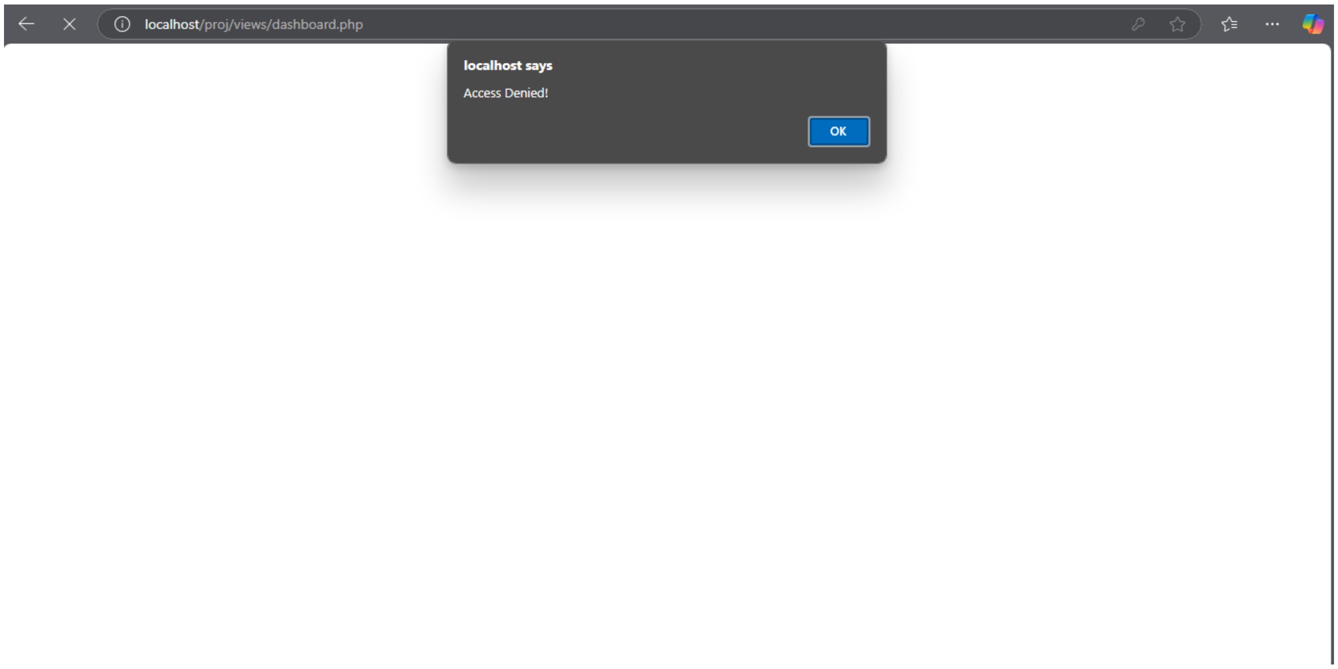

localhost/proj/views/send_files_to_users.php
Data Leakage Detection System
ADMIN
Logout
Select Recipient's Username
Subject
File
Choose File
No file chosen
Send

Fig 6(g): Send File



localhost says

Access Denied!

Fig 6(h): Blocked User

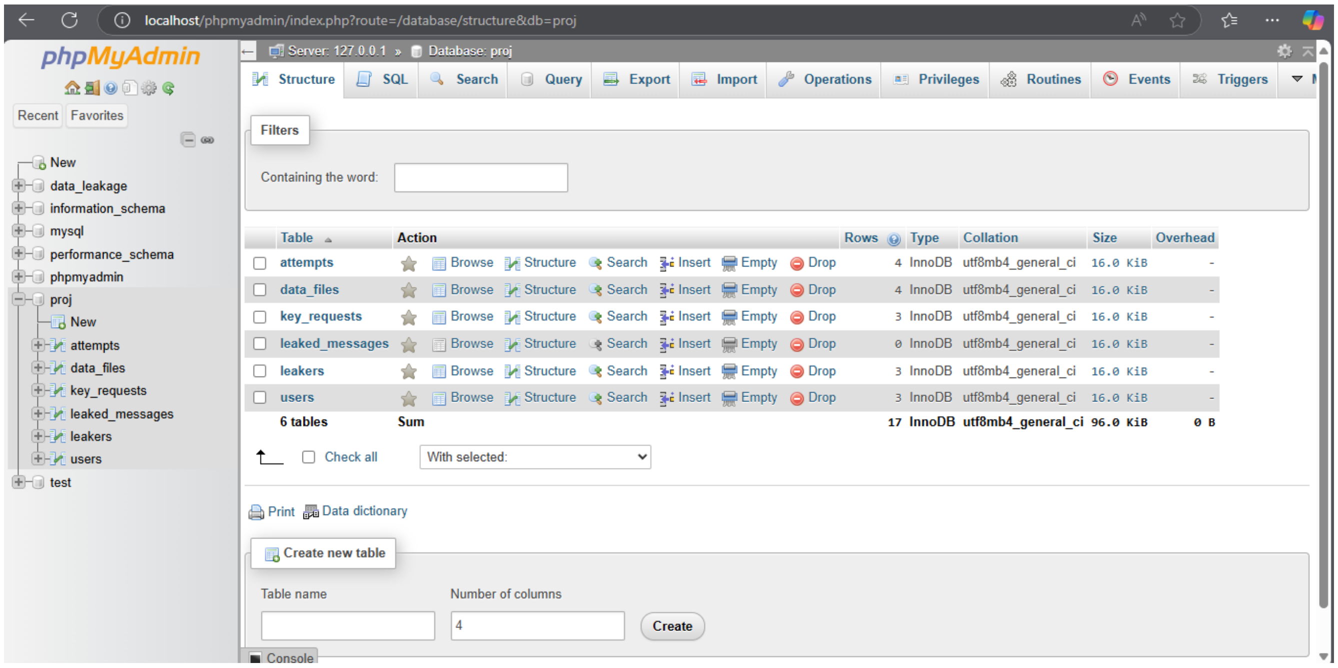
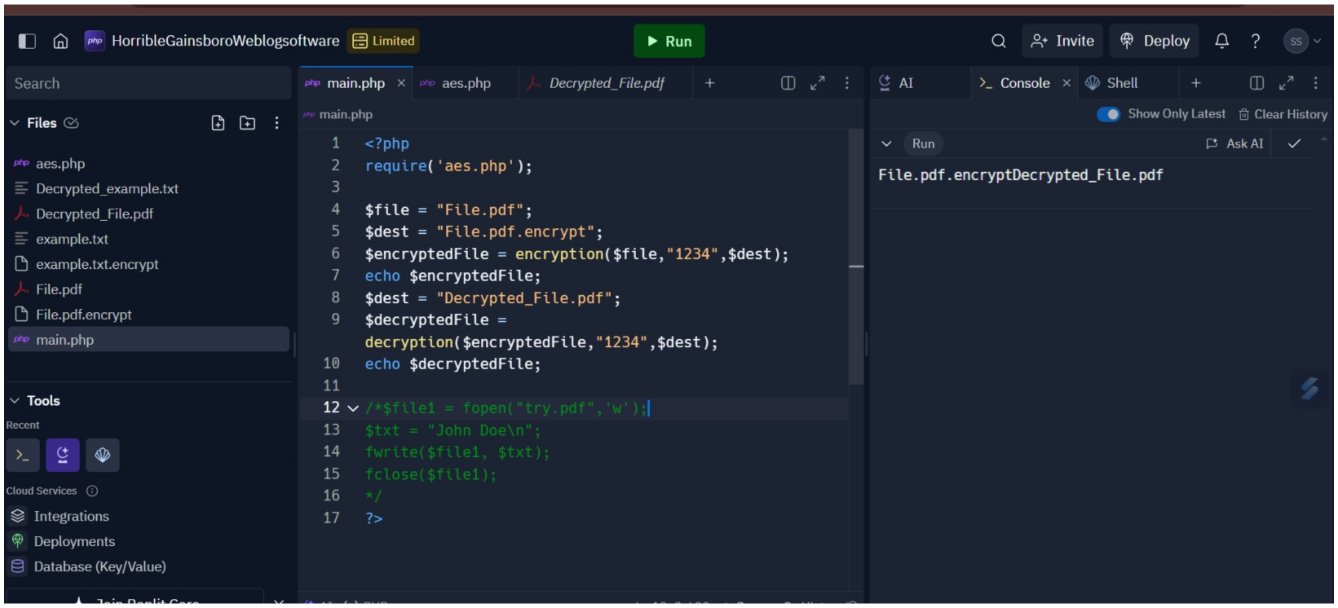


Table	Action	Rows	Type	Collation	Size	Overhead
attempts	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
data_files	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
key_requests	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
leaked_messages	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
leakers	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
users	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
6 tables	Sum	17	InnoDB	utf8mb4_general_ci	96.0 KiB	0 B

Fig 6(i): phpMyAdmin



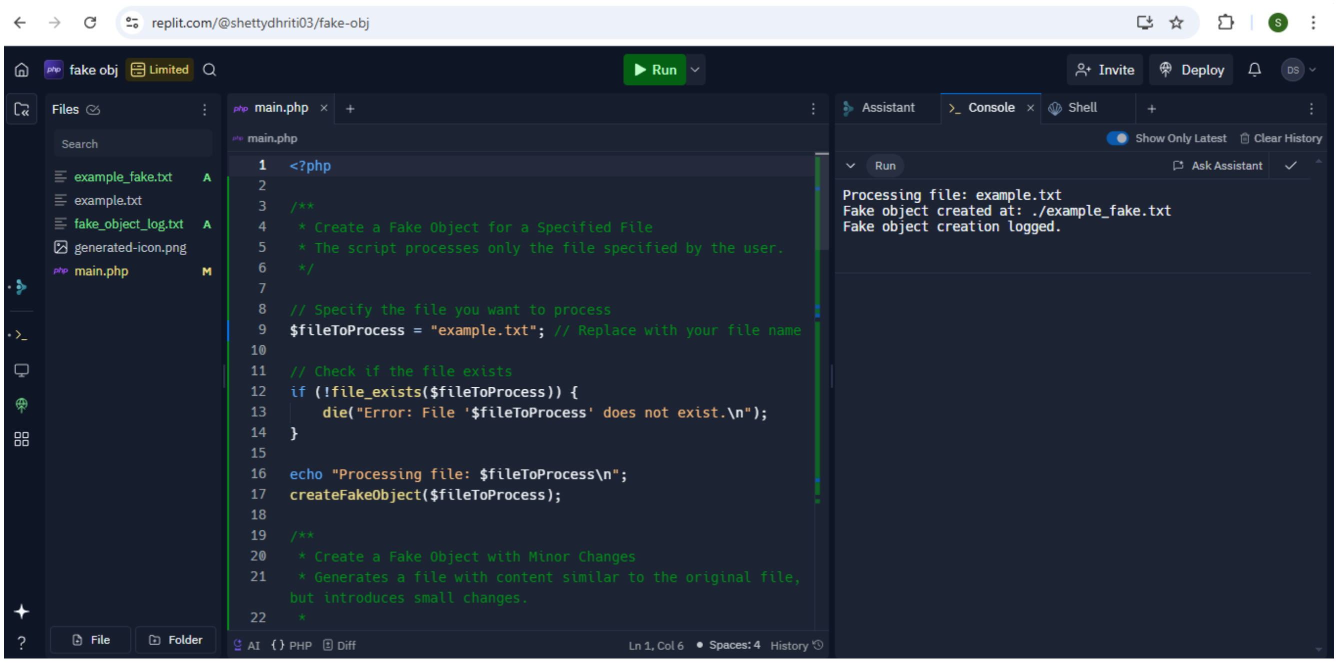
```php
<?php
require('aes.php');

$file = "File.pdf";
$dest = "File.pdf.encrypt";
$encryptedFile = encryption($file,"1234",$dest);
echo $encryptedFile;
$dest = "Decrypted_File.pdf";
$decryptedFile =
decryption($encryptedFile,"1234",$dest);
echo $decryptedFile;

/*$file1 = fopen("try.pdf",'w');
$txt = "John Doe\n";
fwrite($file1, $txt);
fclose($file1);
*/
?>
```

```
File.pdf.encryptDecrypted_File.pdf
```

Fig 6(j): AES Algorithm



replit.com/@shettydhriti03/fake-obj

```
<?php

/**
 * Create a Fake Object for a Specified File
 * The script processes only the file specified by the user.
 */

// Specify the file you want to process
$fileToProcess = "example.txt"; // Replace with your file name

// Check if the file exists
if (!file_exists($fileToProcess)) {
    die("Error: File '$fileToProcess' does not exist.\n");
}

echo "Processing file: $fileToProcess\n";
createFakeObject($fileToProcess);

/**
 * Create a Fake Object with Minor Changes
 * Generates a file with content similar to the original file,
but introduces small changes.
 *
```

```
Processing file: example.txt
Fake object created at: ./example_fake.txt
Fake object creation logged.
```

Fig 6(k): Fake Objection Insertion Algorithm

Chapter 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In this phase, we successfully integrated the watermarking algorithm with the portal system, allowing for seamless embedding of unique metadata within the files uploaded through the portal. This integration enhances the security of the system by ensuring traceability and detecting unauthorized access. Meanwhile, the other algorithms continue to run independently, contributing additional layers of security. With this progress, we have strengthened the foundation of our system, and in the next phase, we aim to integrate all algorithms into a unified system, improving both its functionality and security.

7.2 Future Scope

The future scope of this project includes expanding data leakage detection capabilities through the integration of advanced machine learning algorithms and artificial intelligence. By analyzing patterns and anomalies in data access and usage, these technologies will enhance the identification of sophisticated threats and improve the accuracy of fake object detection. This advancement could enable more proactive measures in detecting and preventing potential breaches, thereby reducing the need for manual monitoring and intervention. Incorporating real-time analytics and automated response systems will further streamline the management of detected threats, enhancing the overall security of cloud storage environments.

Another significant area for future development involves increasing the system's applicability across various cloud platforms and services. By enhancing compatibility with different cloud environments and storage solutions, the technology will become more versatile and widely applicable. This will ensure that the data leakage detection system can be effectively used in diverse settings, providing robust security measures regardless of the specific cloud infrastructure.

Further research could also focus on optimizing encryption processes and watermarking techniques to reduce performance impacts while maximizing security. Investigating new methods to enhance these technologies will help maintain the system's effectiveness against evolving threats. By continuously improving these aspects, the system will be able to uphold the highest standards of data protection and confidentiality, ensuring it remains resilient and effective in the face of emerging challenges.

Chapter 8

REFERENCES

[1] Secure Data Storage and Sharing Techniques for Data Protection in Cloud Environments: A Systematic Review, Analysis, and Future Directions 1. Ishu Gupta, 2. Ashutosh Kumar Singh, 3. Chung-Nan Lee, 4. Rajkumar Buyya
Date of Publication: 4 July 2022 | Date Added to IEEE Xplore: 11 July 2022

[2] Avoiding the data leakage and providing privacy to data in Networking 1. Madhavi Suryawanshi, 2. Prof . Sarita Patil
Date of Conference: 12-13 August 2016 | Date Added to IEEE Xplore: 23 February 2017

[3] A Survey on Data Breach Challenges in Cloud Computing Security: Issues and Threats 1. R.Barona, 2. E.A.Mary Anita
Date of Conference: 20-21 April 2017 | Date Added to IEEE Xplore: 19 October 2017

[4] Dynamic Data Leakage using Guilty Agent Detection over Cloud 1. Govinda. K, 2. Divya Joseph
Date of Conference: 07-08 December 2017 | Date Added to IEEE Xplore: 21 June 2018

[5] An Analysis of Data Leakage and Prevention Techniques in Cloud Environment 1.T. Brindha, 2. R. S. Shaji
Date of Conference: 18-19 December 2015 | Date Added to IEEE Xplore: 23 May 2016

[6] Data Leakage Detection 1. Panagiotis Papadimitriou, 2.Hector Garcia-Molina
Date of Publication: 17 June 2010

[7] Detection of Data Leakage in Cloud Computing Environment 1. Neeraj Kumar, 2. Vijay Katta, 3. Himanshu Mishra, 4. Hitendra Garg
Date of Conference: 14-16 November 2014 | Date Added to IEEE Xplore: 26 March 2015