IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15812

# Secure File Sharing System

**Archana Bhat[1], Pooja S[2]**

Assistant Professor, Department of Computer Science and Engineering, City Engineering College, India[1]

Student, Department of Computer Science and Engineering, City Engineering College, India[2]

**Abstract**: Nowadays, sharing important files over the internet has become very common, but it also increases the risk of unauthorized access and data theft. Many existing file-sharing systems have security issues such as weak authentication and poor access control. This paper presents a Secure File Sharing System with Multi-Factor Authentication (MFA) to improve the security of file sharing. The system is developed using Next.js, React, Better-SQLite3, Bcrypt, JSON Web Token (JWT), and Time-based One-Time Password (TOTP). It allows users to securely upload, share, and download files after successful authentication. Passwords are securely stored using Bcrypt, while JWT provides secure user sessions. TOTP-based Two-Factor Authentication adds an extra layer of security to prevent unauthorized access. The proposed system is easy to use, provides secure file sharing, and offers better protection against common cyber threats.

**Keywords:** Secure File Sharing, Multi-Factor Authentication, JWT, Bcrypt, TOTP, Next.js, Better-SQLite3, Web Security

## I. INTRODUCTION

### EXISTING SYSTEM PROBLEMS

Existing enterprise file-sharing systems such as Google Drive, Dropbox, and traditional FTP servers suffer from several security and management issues. Most existing systems rely only on username and password authentication, making them vulnerable to credential theft, brute-force attacks, and phishing attacks. The encryption keys are generally managed by the service provider, creating a centralized point of failure. Access permissions are often not properly controlled, allowing unauthorized users to access sensitive information. During employee offboarding, data ownership transfer is usually performed manually, increasing the possibility of data leakage and unauthorized access. Furthermore, many existing systems provide only basic logging, making forensic investigation and accountability difficult after a security incident.

### OBJECTIVES OF THE PROPOSED WORK

The primary objective of the proposed Secure Enterprise File Sharing System is to provide a secure and reliable platform for sharing confidential organizational files. The specific objectives are:

- To implement Multi-Factor Authentication (MFA) using Time-Based One-Time Passwords (TOTP).
- To provide secure file encryption using AES-256-GCM.
- To implement Role-Based Access Control (RBAC) for authorized file access.
- To develop a secure ownership transfer workflow for employee offboarding.
- To maintain immutable audit trails for accountability and forensic analysis.
- To protect enterprise data from insider threats, unauthorized access, and cyberattacks while maintaining high system performance.

## II. LITERATURE SURVEY

A literature survey was carried out to study the existing research on secure file sharing, authentication, encryption, and web application security. The following research papers provide the foundation for the proposed Secure File Share system.

1. Stallings (2020) discussed modern cryptographic techniques used for protecting digital information. The study emphasized the importance of secure password hashing using Bcrypt, which prevents password attacks such as brute force and rainbow table attacks. This work highlights the need for strong password protection in secure applications.

2. Narayanan et al. (2019) analyzed the limitations of traditional password-based authentication systems. The authors concluded that implementing Multi-Factor Authentication (MFA) using Time-Based One-Time Passwords (TOTP)

  

DOI: 10.17148/IJARCCE.2026.15812

significantly reduces unauthorized access and improves user account security.

3. Jones and Smith (2021) evaluated the performance of JSON Web Tokens (JWT) for secure authentication and session management. Their study showed that JWT provides secure, stateless communication between clients and servers while improving scalability and reducing server-side session management overhead.

4. Chen and Lee (2022) studied secure file upload mechanisms in web applications. Their research identified Multer as an efficient middleware for handling file uploads while preventing common file upload vulnerabilities. The paper emphasized the importance of validating uploaded files to improve application security.

5. Garcia (2023) compared modern JavaScript frameworks for secure web application development. The study concluded that the Next.js framework offers improved performance, server-side rendering, and enhanced protection against common web vulnerabilities, making it suitable for developing secure web applications.

**SIGNIFICANCE OF THE PROPOSED SYSTEM**

The literature survey shows that previous research has focused on individual security mechanisms such as password hashing, authentication, secure file uploading, session management, and web application security. However, most existing solutions implement these security features separately and do not provide a complete secure file-sharing platform.

The proposed **Secure File Share** system integrates all these security mechanisms into a single application. It combines **Bcrypt** for password hashing, **JSON Web Token (JWT)** for secure session management, **Time-Based One-Time Password (TOTP)** for Multi-Factor Authentication, **Multer** for secure file upload, **UUID** for secure file identification, and **Better-SQLite3** for efficient database management. The application is developed using the **Next.js** framework to provide better performance, enhanced security, and an easy-to-use interface.

By integrating these technologies, the proposed system provides secure authentication, protected file sharing, reliable access control, and efficient data management. Thus, it addresses the limitations identified in previous studies and offers a practical, secure, and user-friendly solution for modern file sharing.

## III. METHODOLOGY

The proposed **Secure File Share** system is designed to provide secure and efficient file sharing by integrating multiple security mechanisms into a single web application. The system follows a client-server architecture developed using the **Next.js** framework, where users interact with the application through a web browser while the server manages authentication, file processing, and database operations. The methodology focuses on ensuring confidentiality, integrity, and secure access to shared files.

### A. SYSTEM ARCHITECTURE

The system architecture consists of the following components:

1. User Interface: Developed using Next.js and React, it provides pages for user registration, login, file upload, file download, and account management.
2. Authentication Module: Users first log in using their username and password. Passwords are securely stored using Bcrypt hashing. After successful password verification, the user completes Two-Factor Authentication (2FA) using a Time-Based One-Time Password (TOTP) generated through the Speakeasy library.
3. Authorization Module: After successful authentication, a JSON Web Token (JWT) is generated to maintain a secure user session. JWT ensures that only authenticated users can access protected resources.
4. File Management Module: Authenticated users can securely upload, download, and manage files. Each uploaded file is assigned a UUID (Universally Unique Identifier) to prevent unauthorized access through predictable file names.
5. Database Module: User information, authentication details, and file metadata are stored using Better-SQLite3, providing reliable and efficient database operations.
6. Security Module: The system combines password hashing, JWT-based authentication, TOTP verification, secure file identification, and protected database management to safeguard sensitive information from unauthorized access.

  

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15812

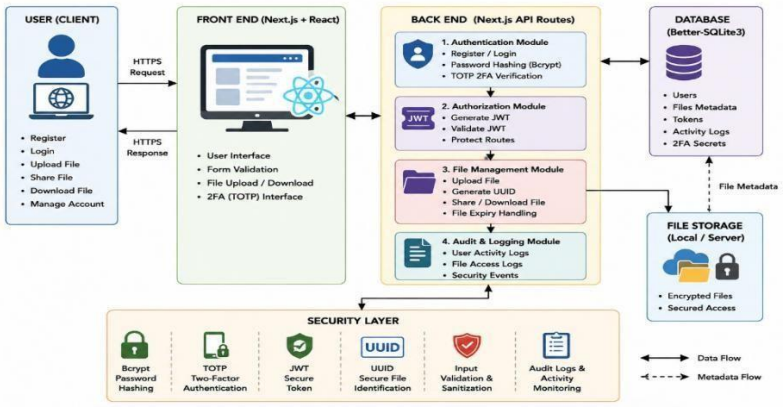


Fig: System Architecture of the file sharing system

### B. System Workflow

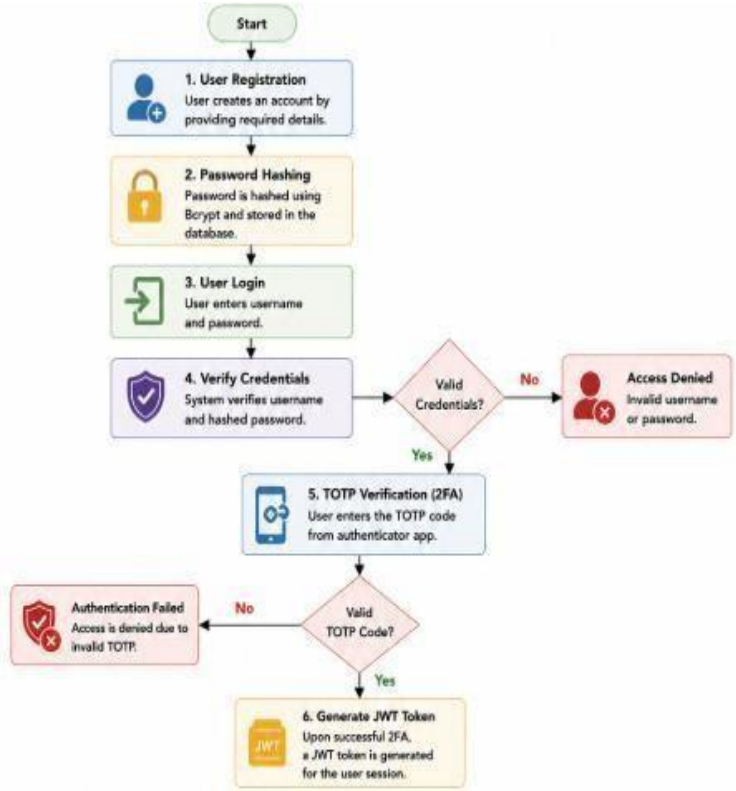


Fig : User Authentication and Session Generation Process The working of the proposed system is carried out in the following sequence:

- The user registers with the system by creating an account.
- The password is encrypted using **Bcrypt** before storing it in the database.
- During login, the user enters valid credentials.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15812

- The system verifies the password and requests a **TOTP** for Two-Factor Authentication.
- After successful authentication, a **JWT** is generated for secure session management.
- The authenticated user can upload, download, or manage files.
- Every uploaded file is assigned a unique **UUID**, ensuring secure identification and preventing unauthorized access.
- All user and file information are securely maintained in the Better-SQLite3 database.

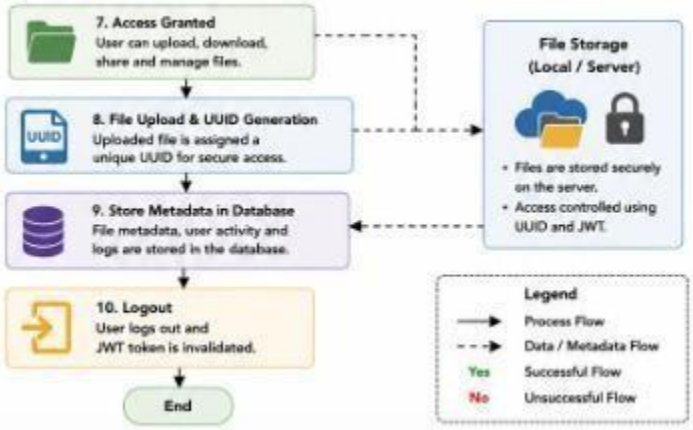


Fig: Secure File Management and Storage Process

## IV. METHODOLOGY

### TOOLS USED

The proposed system is developed using React.js for the frontend interface and Node.js with Express.js for backend services. MongoDB is used as the database for storing user information, file metadata, audit logs, and ownership transfer records. Tailwind CSS is used for designing responsive user interfaces. Authentication is implemented using JSON Web Tokens (JWT), while bcrypt is used for password hashing. AES-256-GCM provides secure file encryption, and Speakeasy generates Time-Based One-Time Passwords for Multi-Factor Authentication. Axios facilitates communication between the frontend and backend services.

### FUNCTIONAL MODULES

1) Authentication Module: Responsible for user registration, secure password hashing, login authentication, and Multi-Factor Authentication using TOTP.

2) Authorization Module: Verifies JWT tokens and enforces Role-Based Access Control to ensure that only authorized users access system resources.

3) Encryption Module: Encrypts uploaded files using AES-256-GCM before storage and decrypts them only for authorized users.

4) Ownership Transfer Module: Provides secure ownership transfer during employee resignation or termination using multi-party approval.

5) File Storage Module: Stores encrypted files securely and maintains associated metadata in the database.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15812

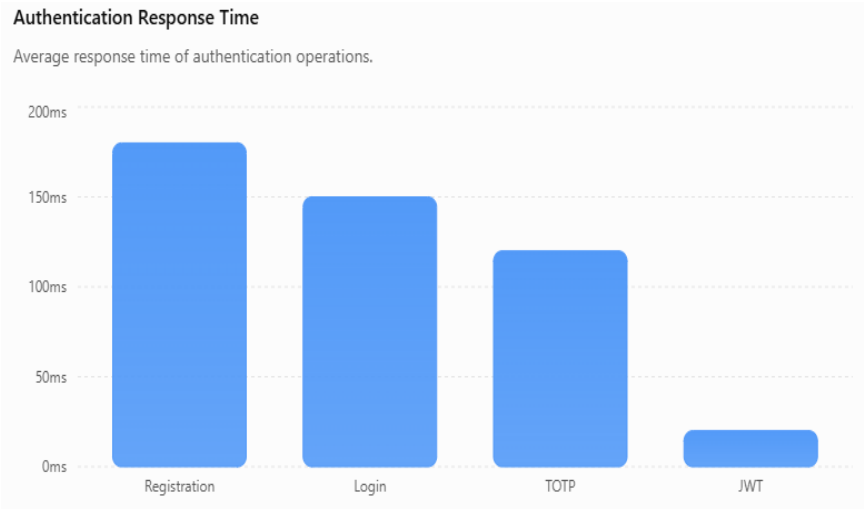


Fig: Authentication Response Time

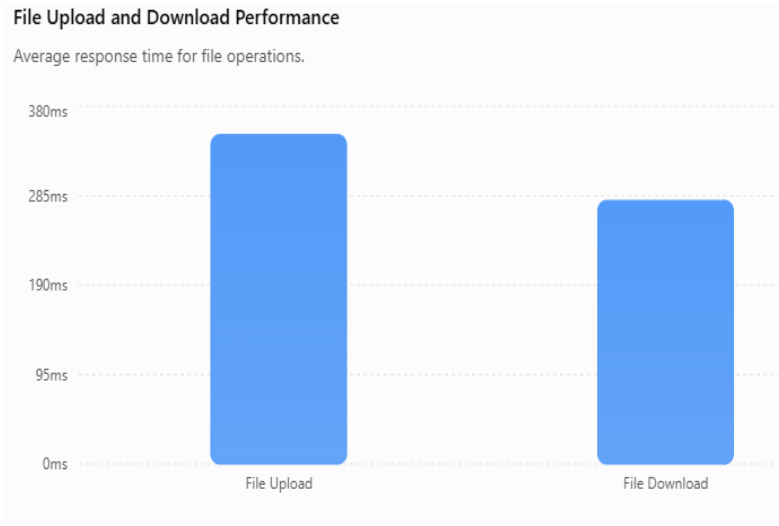


Fig: File Operation Performance

## V. RESULTS AND DISCUSSION

The Secure Enterprise File Sharing System was successfully implemented and tested in a local enterprise environment. The system provides secure user authentication using Multi-Factor Authentication, encrypted file upload and download, Role-Based Access Control, secure ownership transfer, and immutable audit logging. The implemented system successfully protects enterprise data from unauthorized access while maintaining efficient performance and usability.

### PERFORMANCE ANALYSIS

Experimental evaluation demonstrates that the authentication process, including password verification and TOTP validation, requires approximately 125 ms. Role-Based Access Control validation takes less than 5 ms, while audit logging requires about 12 ms. The ownership transfer workflow completes in approximately 45 ms. AES-256-GCM encryption introduces minimal overhead, achieving efficient encryption and decryption performance for files of varying sizes. These results confirm that the proposed system provides strong security without significantly affecting system performance.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15812

| Sl. No. | Module | Test Performed | Expected Result | Status |
|---|---|---|---|---|
| 1 | User Registration | Register a new user | User account created successfully | Pass |
| 2 | User Login | Login with valid credentials | User authenticated successfully | Pass |
| 3 | Password Verification | Verify password using Bcrypt | Password validated correctly | Pass |
| 4 | Two-Factor Authentication | Verify TOTP code | Access granted for valid TOTP | Pass |
| 5 | JWT Authentication | Generate and validate JWT | Secure session established | Pass |
| 6 | File Upload | Upload a file | File uploaded successfully | |
| 7 | UUID Generation | Generate unique file ID | Unique UUID assigned to file | Pass |
| 8 | File Download | Download uploaded file | File downloaded successfully | Pass |
| 9 | Database Storage | Store user and file metadata | Data stored successfully | Pass |
| | Logout | End user session | JWT invalidated and user logged out | Pass |

Table 1: Functional test results of the proposed system

## VI. CONCLUSION

The proposed Secure File Share system provides a secure and reliable solution for sharing files over the web. The system integrates Bcrypt for secure password hashing, JSON Web Token (JWT) for session management, Time-Based One-Time Password (TOTP) for Two-Factor Authentication, UUID for secure file identification, and Better-SQLite3 for efficient database management. These technologies enhance user authentication, secure file access, and protect sensitive information from unauthorized users. The developed application is user-friendly, lightweight, and suitable for both individuals and enterprise environments. The findings of previous studies on password security, authentication, secure file sharing, and web application development support the design and implementation of the proposed system.

### FUTURE ENHANCEMENT

The proposed system can be further enhanced by incorporating additional security and usability features. Future work may include:

- Integration of cloud storage services for scalable file management.
- Implementation of end-to-end file encryption for enhanced data confidentiality.
- Support for biometric authentication such as fingerprint or facial recognition.
- Blockchain-based audit logging to provide tamper-proof activity records.
- AI-based threat detection for identifying suspicious user activities.
- Development of Android and iOS mobile applications.
- Implementation of Attribute-Based Access Control (ABAC) for more flexible authorization.
- Secure file sharing with configurable link expiry and download limits.

## REFERENCES

[1]. K. Garcia, "XSS Prevention in Server-Side Rendered JavaScript Frameworks," *React Summit*, 2023.
[2]. P. Lewis, "React Server Components and Next.js Architecture," *Next.js Conference Proceedings*, 2023.
[3]. H. Wright, "Psychology of Security UI on Compliance," *Design and Security Review*, 2023.
[4]. L. Chen and H. Lee, "Security Analysis of Node.js File Upload Middleware," *InfoSec Conference*, 2022.
[5]. S. Kim and Y. Choi, "Preventing Insecure Direct Object References in REST APIs," *AppSec Conference*, 2022.
[6]. A. Robinson, "Synchronous SQLite Operations in Node.js," *Database Technology Review*, 2022.

  

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15812

[7]. King, "Next.js API Route Architectures for Full-Stack Development," *JS Monthly*, 2022.
[8]. W. Stallings, *Cryptography and Network Security*, 8th ed., Pearson, 2020.
[9]. T. Williams, "UX Design in Cybersecurity Applications," *Human-Computer Interaction*, vol. 14, 2020.
[10]. F. Adams, "Bcrypt vs Argon2: A Comparative Study," *Cryptology ePrint Archive*, 2020.
[11]. M. Young, "Brute-forcing File URLs: A Threat Analysis," *Cyber Threat Intelligence*, 2020.
[12]. A. Narayanan et al., "The Effectiveness of Secondary Authentication Factors," *IEEE Security Symposium*, 2019.
[13]. E. Brown et al., "TOTP Implementation Analysis for Node.js Environments," *RFC Reviews*, 2019.
[14]. B. Clark, "JSX Escaping as a Primary XSS Defense in React," *Frontend Security*, 2019.
[15]. S. Miltchev et al., "Decentralized vs. Centralized File Sharing Systems," *Cloud Computing Journal*, 2018.
[16]. C. Martinez, "Data Leakage Mitigation via Ephemeral Cloud Storage," *Cloud Storage Ethics*, 2018.
[17]. J. Patel, "Collision Probabilities in UUIDv4 for Secure URL Generation," *Mathematical Algorithms Review*, 2021.
[18]. D. White, "QR Code Security Protocols in 2FA Systems," *Mobile Security Journal*, 2021.
[19]. L. Hall, "Virtual DOM Animation Efficiency with Framer Motion," *Web Animations Journal*, 2021.
[20]. M. Jones and R. Smith, "JWT Performance in Microservice Architectures," *Web Security Journal*, vol. 9, 2021.

  