Project Name: Suraksha Cloud

Suraksha Cloud
A Secure Server-Side Rendered File Sharing Platform using AWS S3 & Cloudflare CDN

---

1. Project Overview

Suraksha Cloud is a secure and scalable file-sharing platform developed using **Node.js, Express, EJS, and MongoDB**. The application uses **server-side rendering (SSR)** to generate pages on the server and ensures better performance and security.

Files uploaded by users are stored securely in **AWS S3**, and file previews and downloads are delivered through **Cloudflare CDN**. This helps in faster file access, reduced server load, and improved user experience. The project is designed to reflect real-world cloud-based systems used in the industry.

---

2. Problem Statement

Many traditional student file-sharing systems depend on local storage or publicly accessible cloud links. This causes several problems such as: - Poor scalability when users increase - Security risks due to public file access - Slow file access for users from different locations - High bandwidth usage on the main server

Suraksha Cloud solves these problems by using **private cloud storage** and a **CDN-based delivery system** with proper access control.

---

3. Objectives

- To build a secure file-sharing platform using server-side rendering
- To store files safely using AWS S3
- To deliver files faster using Cloudflare CDN
- To control file access using permissions and expiry-based sharing

- To follow clean coding practices and proper project structure

---

4. Why This Architecture Was Chosen

4.1 Server-Side Rendering (SSR with EJS)

Why - Faster page load for the first request - Better SEO for file-sharing pages - Less work on the client side - Suitable for dashboards and content-based applications

Reason
In a file-sharing system, user authentication and authorization are important. SSR allows the server to verify the user before rendering the page.

Implementation - Express renders EJS templates for each request - Controllers fetch data from MongoDB before rendering - Pages are generated dynamically using server-side logic

---

4.2 AWS S3 for File Storage

Why - Very high durability and reliability - Can handle a large number of files - Cost-effective for storage - Widely used in real-world applications

Reason
Storing files on the application server is not secure or scalable. AWS S3 helps separate file storage from application logic.

Implementation - A private S3 bucket is used with public access blocked - Files are uploaded from Node.js using the AWS SDK - Only file information (metadata) is stored in MongoDB

4.3 Cloudflare CDN for File Delivery

Why - Faster file delivery for users worldwide - Reduces load on AWS S3 - Protects against DDoS attacks - Improves caching and performance

Reason
Serving files directly from S3 for every request can be slow and expensive. Cloudflare stores cached files closer to users.

Implementation - Cloudflare is placed in front of AWS S3 - CDN URLs are stored in the database - All file previews and downloads use the CDN URL

4.4 MongoDB for Metadata Storage

Why - Flexible schema - Easy to scale - Fast data access

Reason
File data like permissions and sharing options may change often. MongoDB allows easy updates without complex changes.

Implementation - Separate collections for Users, Files, and Share Links - Indexing is used for better performance

---

4.5 Docker & CI/CD

Why - Same environment for development and production - Faster and safer deployments - Easy collaboration

Reason
Automation helps reduce manual errors and improves deployment reliability.

Implementation - Docker is used to containerize the application - GitHub Actions handles build and deployment

---

5. Technology Stack

Backend

- Node.js
- Express.js
- EJS (Server-Side Rendering)
- MongoDB (Mongoose ODM)

Cloud & DevOps

- AWS S3 (Private Bucket)
- Cloudflare CDN
- Docker
- GitHub Actions

Security

- Session-based authentication
- Secure file access rules
- Environment variables for secrets

---

6. System Architecture

High-Level Flow

User → Node.js SSR Application → AWS S3 (Storage)
↓
Cloudflare CDN

- Files are uploaded to private S3 buckets
- Cloudflare caches and serves files
- MongoDB stores only file metadata

---

7. Application Features

User Features

- User registration and login
- Upload files (images, PDFs, documents)
- Preview files using CDN
- Download files securely
- Share files with expiry links
- Organize files using folders

Admin Features

- Manage users
- Monitor storage usage
- Moderate uploaded files
- Track user activity

---

8. Database Design

User Collection

```
{
  name,
  email,
  password,
  role,
  storageUsed,
  createdAt
}
```

File Collection

```
{
  ownerId,
  originalName,
  s3Key,
  cdnUrl,
  mimeType,
  size,
  isPublic,
  downloadCount,
  createdAt
}
```

Share Link Collection

```
{
  fileId,
  token,
  expiresAt,
  password
}
```

---

9. Security Considerations

- S3 bucket access is fully private
- Files are accessed only through CDN or secure links
- Cloudflare provides protection against attacks
- Sensitive data is stored in environment variables
- Middleware protects private routes

---

10. Folder Structure

```
src/
  ├── controllers/
  ├── routes/
  ├── models/
  ├── services/
  │     └── s3.service.js
  ├── views/
  ├── public/
  ├── middlewares/
  └── app.js
```

---

11. Deployment Strategy

- Application is containerized using Docker
- CI/CD pipeline is implemented with GitHub Actions
- Deployed on AWS EC2
- Environment variables managed using `.env`

---

12. Future Enhancements

- Virus scanning using ClamAV
- File versioning support
- Cloudflare Workers for better access control
- Storage limit per user
- Audit and activity logs

---

13. Conclusion

Suraksha Cloud is a practical and real-world file-sharing system that combines server-side rendering, cloud storage, and CDN delivery. This project helped in understanding secure file handling, cloud services, and deployment practices, making it suitable for academic and beginner industry-level learning.

---

14. Project Repository

(To be added)

Prepared By: Saroj Kumar Tharu
Technology Used: Node.js, EJS, MongoDB, AWS S3, Cloudflare CDN