PROJECT REPORT

Cloud-Based Smart File Storage & Sharing System

Submitted By:
Aishwarya Kopulwar

Internship Title:
Cloud Computing Internship (Remote)

Company:
Elevate Labs, Bengaluru

Internship Duration:
20 October 2025 – 24 November 2025

Submission Date:
24 November 2025

Table of Contents

1. Abstract 1
2. Introduction 2
3. System Architecture 3
4. System Architecture Diagram 4
5. Tools & Technologies Used 5
6. Implementation 6
7. Features Implemented 7
8. System Implementation Screenshots
    - 8.1 AWS EC2 – Backend Deployment 8
    - 8.2 AWS S3 – File Storage 9
    - 8.3 AWS CloudFront – Frontend Delivery 10
    - 8.4 Amazon DynamoDB – Database Layer 11
9. Results 12
    - 9.1 User Authentication 13
    - 9.2 File Upload 14
    - 9.3 File Listing Dashboard 15
    - 9.4 File Download 16
    - 9.5 Copy Link 17
    - 9.6 Share File 18
    - 9.7 File Deletion 19
10. Conclusion 20
11. Future Enhancements 21
12. References 22

1. Abstract

This project presents a cloud-based file storage and sharing system similar to Google Drive. The objective is to allow users to securely upload, store, manage, and share files through a web interface. The system is built using Amazon Web Services (AWS) for scalability, reliability, and data durability.

The backend is developed using Python Flask, while the frontend is built using HTML, CSS, and JavaScript. AWS S3 stores user files, and DynamoDB stores metadata. Authentication is implemented using JWT tokens, ensuring only authorized users can access their files. Users can upload, list, download, share, and delete files.

This project demonstrates the use of cloud platforms to build modern applications with real-world functionality.

2. Introduction

Cloud storage platforms like Google Drive and Dropbox have become essential for managing digital files. They provide easy access, secure storage, and global availability. This project aims to implement a simplified version of such platforms using AWS services.

The objective is to develop a secure and scalable cloud-based file storage system where users can:

- Sign up and log in
- Upload files
- View/download stored files
- Share files via links
- Delete files

By integrating AWS services with a Flask backend and a user-friendly frontend, the system provides a complete cloud storage workflow.

3. System Architecture

The system consists of four components:

1. Frontend (HTML, CSS, JavaScript)

- Provides signup/login interface
- Handles file uploads and actions
- Interacts with backend APIs via HTTPS

2. Backend (Flask API)

- Manages authentication
- Handles file upload, retrieval, and deletion
- Communicates with S3 and DynamoDB
- Secures routes using JWT authentication

3. Cloud Storage (Amazon S3)

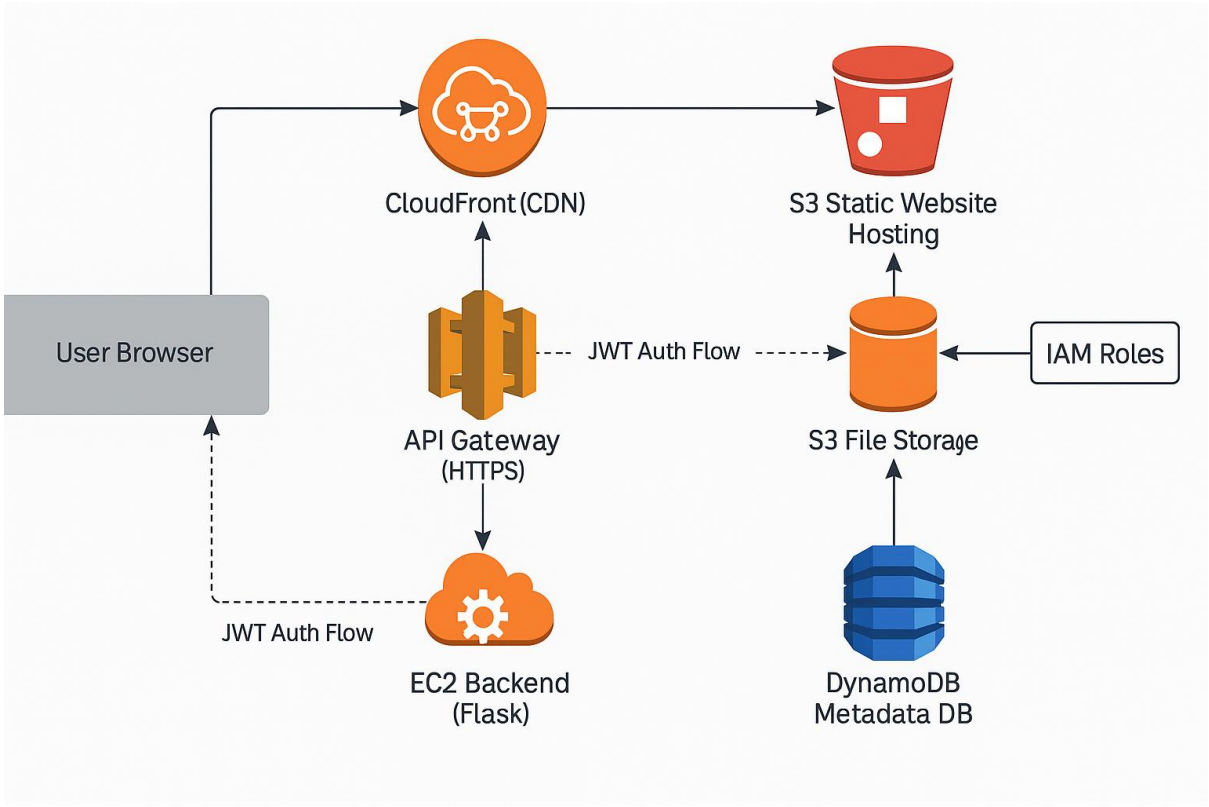
- Stores all uploaded user files
- Generates pre-signed download URLs

4. Database (DynamoDB)

- Stores user account information
- Stores file metadata (file name, owner, timestamps, etc.)

API Gateway is used to expose backend routes securely over HTTPS, and CloudFront is used as a CDN for hosting the frontend with improved speed and global accessibility.

4. System Architecture Diagram:



5. Tools & Technologies Used

AWS Services

- **EC2** – Hosts the Flask backend
- **S3** – Stores uploaded files + hosts frontend
- **API Gateway** – Provides secure HTTPS endpoints
- **DynamoDB** – Stores user data and file metadata
- **CloudFront** – CDN for secure and fast content delivery

Programming & Frameworks

- Python Flask (Backend)
- HTML, CSS, JavaScript (Frontend)
- JWT Authentication
- Boto3 (AWS SDK for Python)

6. Implementation

Backend Development

- Built REST APIs for signup, login, upload, list, and delete.
- Implemented JWT authentication to protect private routes.
- Integrated S3 for file uploads and pre-signed URLs.
- Saved file details in DynamoDB.

Frontend Development

- Designed a clean, responsive UI for all user interactions.
- Implemented JavaScript logic for interacting with API Gateway.
- Added upload, copy link, share, and delete functionalities.

Cloud Setup

- Configured S3 bucket for frontend hosting.
- Created IAM roles and permissions for EC2 → S3 → DynamoDB access.
- Designed DynamoDB tables for Users and Files.
- Deployed Flask backend on EC2 instance.
- Added API Gateway for HTTPS secure communication.
- Integrated CloudFront for fast global deployment.

7. Features Implemented

- User Signup & Login
- JWT Based Authentication
- File Upload to S3
- File Download using Pre-Signed URLs
- File Delete Functionality
- File Sharing through Shareable Links
- Metadata Storage in DynamoDB
- Clean & Simple UI
- HTTPS-Secured Backend & Frontend
- CloudFront CDN Integration

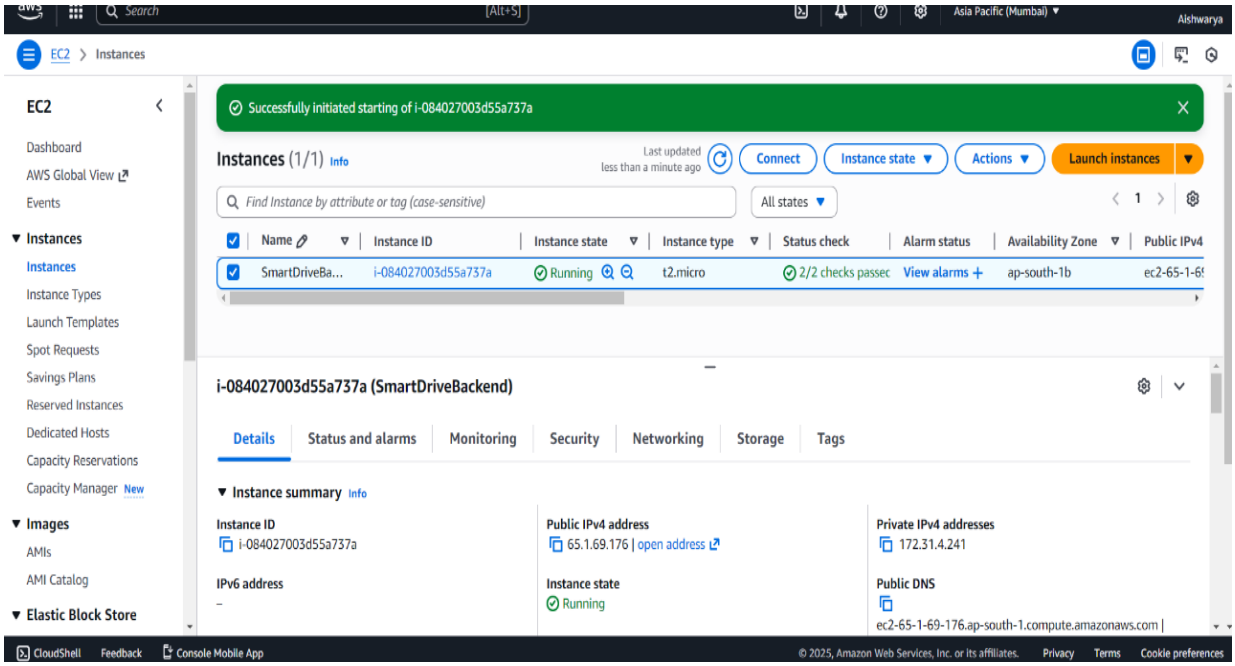
8. System Implementation Screenshots

This section presents the essential implementation screenshots of the cloud-based file storage system. Only the required visuals are included to demonstrate the architecture, deployment, and functionality of the system.

8.1 AWS EC2 – Backend Deployment

EC2 Instance Running the Flask API

The backend is deployed on an Amazon EC2 instance running Amazon Linux.
It hosts the Python Flask application that handles authentication, file operations, and API interactions.



Flask Backend Service Running

The backend service is started using Python and listens on port **5000** for API requests.

```
ec2-user@ip-172-31-4-241:~/smart-drive/backend
HP@LAPTOP-SPP29OTG MINGW64 ~/Downloads
$ ssh -i "Smart-drive-backend-keypair.pem" ec2-user@ec2-65-1-69-176.ap-south-1.c
ompute.amazonaws.com

A newer release of "Amazon Linux" is available.
  Version 2.023.9.20251110:
  Version 2.023.9.20251117:
Run "/usr/bin/dnf check-release-update" for full release and version update info
       #_
   ~\_  ####_          Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___    https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Nov 22 02:03:58 2025 from 117.222.6.0
[ec2-user@ip-172-31-4-241 ~]$ cd smart-drive/backend
[ec2-user@ip-172-31-4-241 backend]$ python3 app.py
/home/ec2-user/.local/lib/python3.9/site-packages/boto3/compat.py:84: PythonDeprecationWarning: Boto3 will no longer support Python 3.
 and security updates please upgrade to Python 3.10 or later. More information can be found here: https://aws.amazon.com/blogs/develop
  warnings.warn(warning, PythonDeprecationWarning)
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.4.241:5000
Press CTRL+C to quit
 * Restarting with stat
/home/ec2-user/.local/lib/python3.9/site-packages/boto3/compat.py:84: PythonDeprecationWarning: Boto3 will no longer support Python 3.
 and security updates please upgrade to Python 3.10 or later. More information can be found here: https://aws.amazon.com/blogs/develop
  warnings.warn(warning, PythonDeprecationWarning)
 * Debugger is active!
 * Debugger PIN: 136-659-529
3.111.110.38 - - [23/Nov/2025 09:06:07] "OPTIONS /login HTTP/1.1" 200 -
3.7.10.48 - - [23/Nov/2025 09:06:07] "POST /login HTTP/1.1" 200 -
3.111.111.111 - - [23/Nov/2025 09:06:09] "OPTIONS /files?email=test@gmail.com HTTP/1.1" 200 -
3.7.10.18 - - [23/Nov/2025 09:06:09] "GET /files?email=test@gmail.com HTTP/1.1" 200 -
3.7.10.192 - - [23/Nov/2025 09:06:54] "OPTIONS /login HTTP/1.1" 200 -
3.111.111.219 - - [23/Nov/2025 09:06:54] "POST /login HTTP/1.1" 200 -
3.111.110.38 - - [23/Nov/2025 09:07:11] "OPTIONS /files?email=test@gmail.com HTTP/1.1" 200 -
3.7.10.217 - - [23/Nov/2025 09:07:11] "GET /files?email=test@gmail.com HTTP/1.1" 200 -
3.7.10.162 - - [23/Nov/2025 09:07:34] "OPTIONS /delete/b8e280e7-3843-4a8c-95c2-381bce5a170d?email=test@gmail.com HTTP/1.1" 200 -
3.111.110.196 - - [23/Nov/2025 09:07:34] "DELETE /delete/b8e280e7-3843-4a8c-95c2-381bce5a170d?email=test@gmail.com HTTP/1.1" 200 -
3.111.110.94 - - [23/Nov/2025 09:07:36] "OPTIONS /files?email=test@gmail.com HTTP/1.1" 200 -
3.7.10.162 - - [23/Nov/2025 09:07:36] "GET /files?email=test@gmail.com HTTP/1.1" 200 -
3.111.111.111 - - [23/Nov/2025 09:12:04] "OPTIONS /upload HTTP/1.1" 200 -
3.111.110.94 - - [23/Nov/2025 09:12:04] "POST /upload HTTP/1.1" 200 -
3.111.110.38 - - [23/Nov/2025 09:12:19] "OPTIONS /files?email=test@gmail.com HTTP/1.1" 200 -
3.7.10.43 - - [23/Nov/2025 09:12:19] "GET /files?email=test@gmail.com HTTP/1.1" 200 -
3.111.111.219 - - [23/Nov/2025 09:14:40] "OPTIONS /delete/b89914a7-7a39-4e9f-a6ec-9e4b515711bd?email=test@gmail.com HTTP/1.1" 200 -
3.111.111.105 - - [23/Nov/2025 09:14:41] "DELETE /delete/b89914a7-7a39-4e9f-a6ec-9e4b515711bd?email=test@gmail.com HTTP/1.1" 200 -
3.111.111.51 - - [23/Nov/2025 09:15:04] "OPTIONS /files?email=test@gmail.com HTTP/1.1" 200 -
3.111.111.111 - - [23/Nov/2025 09:15:04] "GET /files?email=test@gmail.com HTTP/1.1" 200 -
```

Backend Health Response

A browser request to the EC2 public IP confirms the backend API is active.

Not secure 65.1.69.176:5000

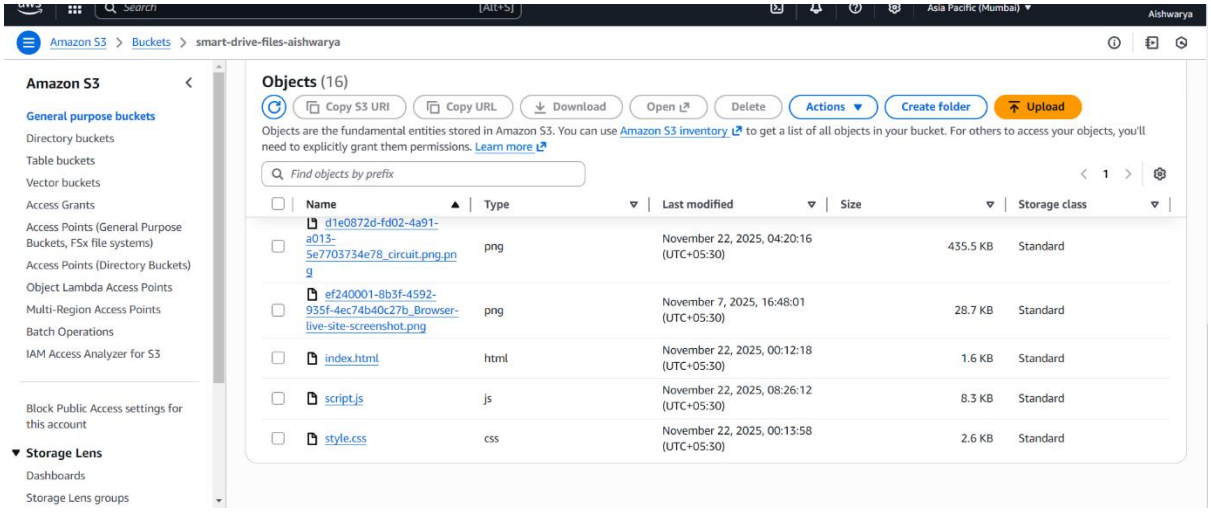
Gmail YouTube Deep Learning-Base... Maps A Whisker Away (Fu... Course M

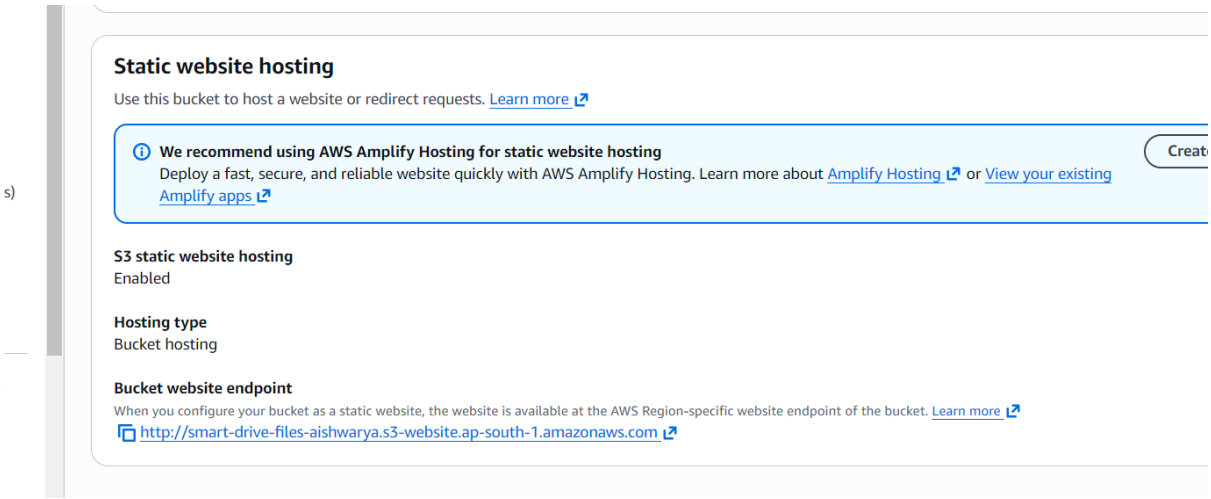
```
{
  "message": "Smart Drive Backend is running successfully!"
}
```

8.2 AWS S3 – File Storage

S3 Bucket Overview

The system uses an S3 bucket to store user-uploaded files.
The screenshot shows the bucket created for this project.

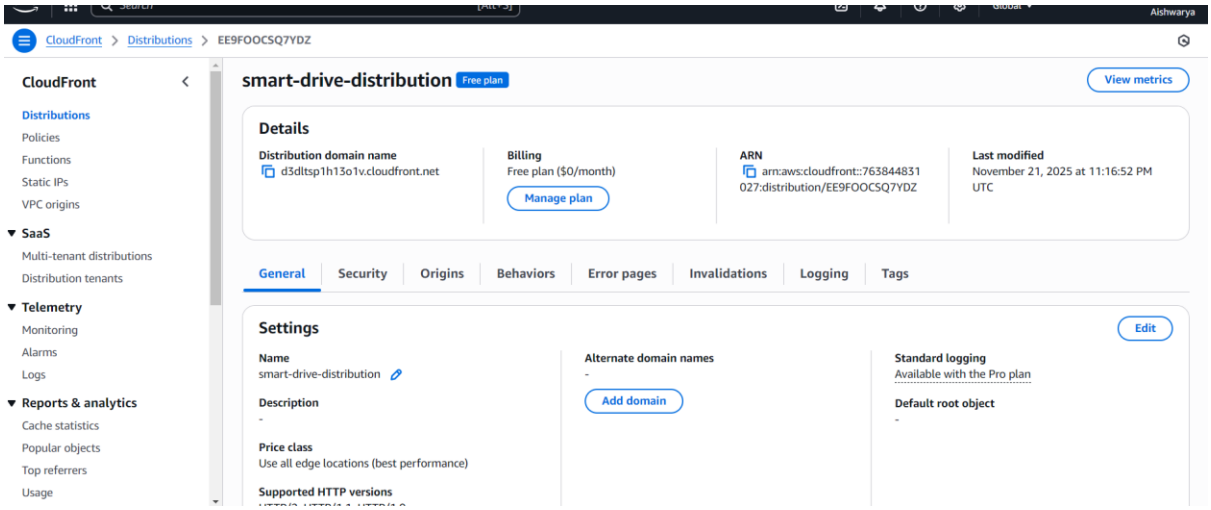




8.3 AWS CloudFront – Secure Frontend Delivery

CloudFront Distribution Configuration

The frontend is hosted on S3 and delivered securely via AWS CloudFront over HTTPS.

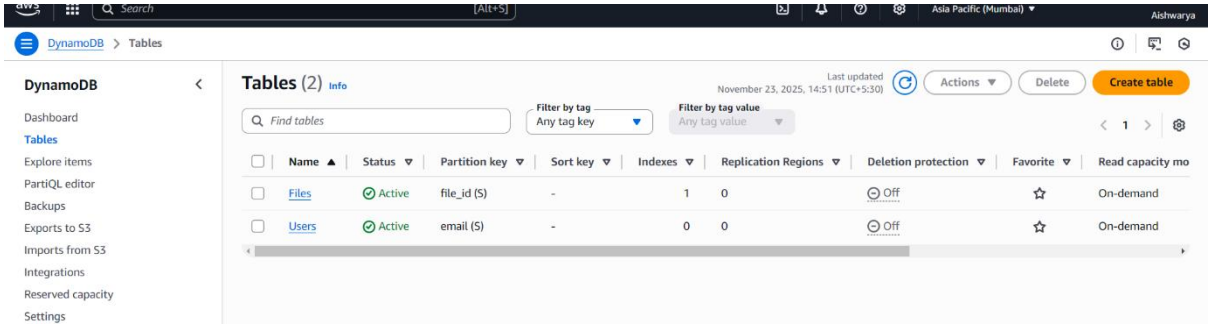


8.4 Amazon DynamoDB – Database Layer

DynamoDB Tables Overview

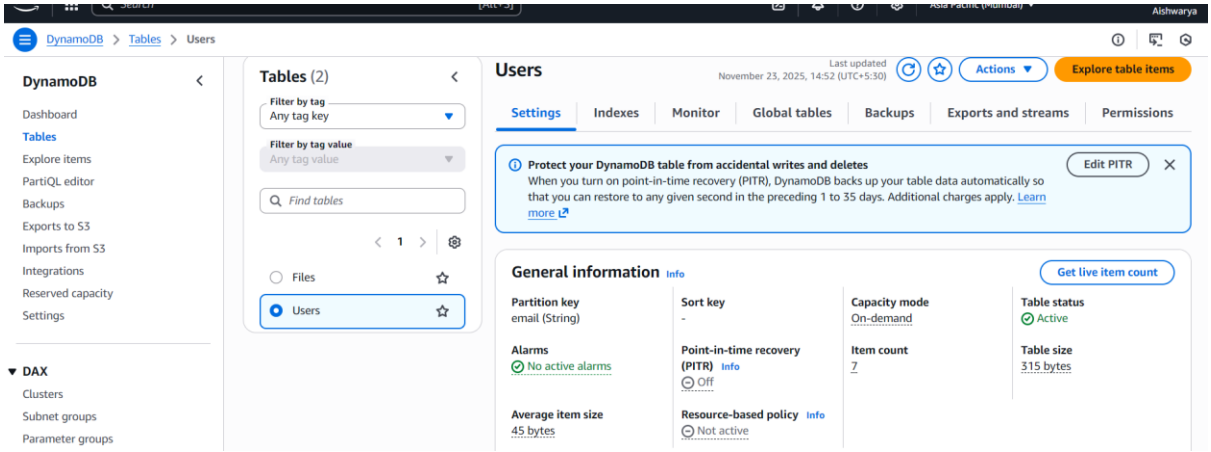
Two DynamoDB tables are used:

- **Users** table for storing user credentials
- **Files** table for storing file metadata



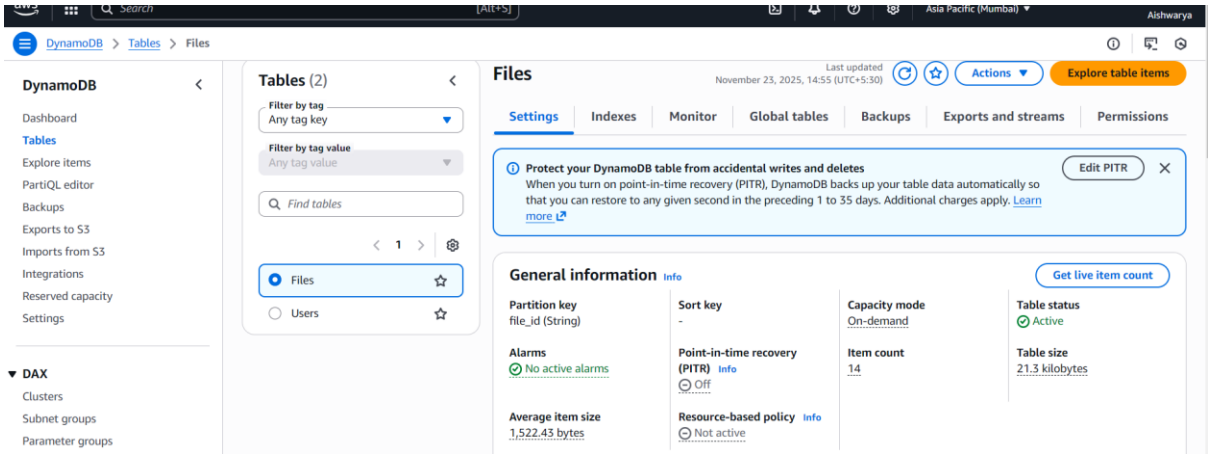
Users Table Structure

Displays the table schema with **email** as the primary key.



Files Table Structure

Displays the table schema with **file_id** as the primary key.



9. Results

The system successfully allows users to upload, download, share, and delete files securely. User and file data is stored in AWS cloud services ensuring reliability. The website loads over HTTPS through CloudFront, and all backend interactions occur securely via API Gateway.

Frontend URLs:

- **CloudFront (Primary – HTTPS):**

  https://d3dltsp1h13o1v.cloudfront.net/

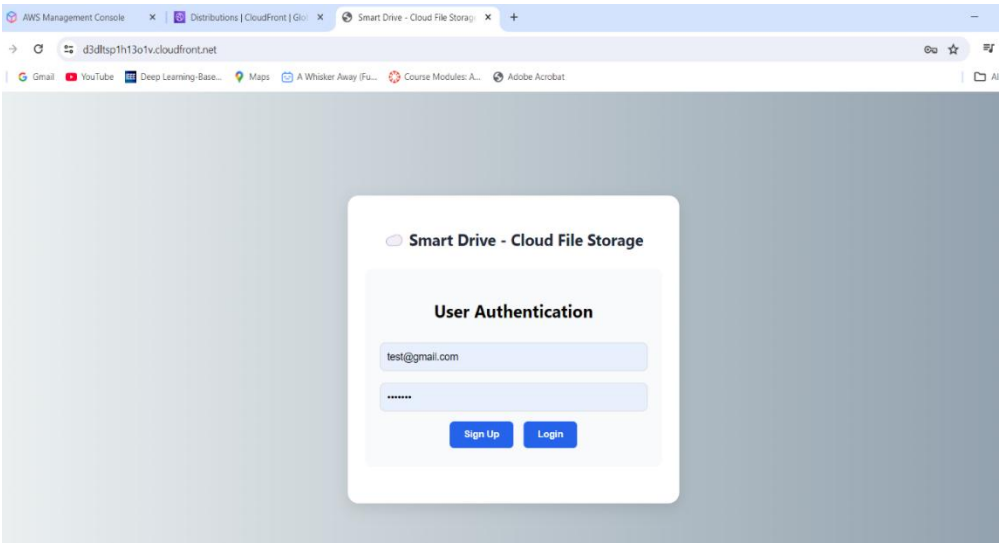
- **S3 Static Website (Optional – HTTP Only):**
  http://smart-drive-files-aishwarya.s3-website.ap-south-1.amazonaws.com

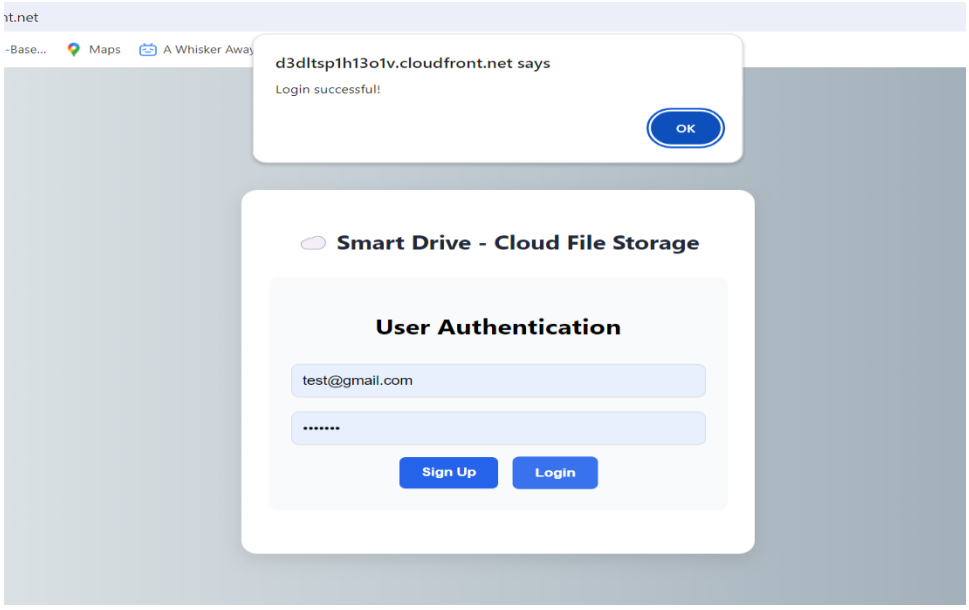
Backend API (API Gateway – HTTPS):
https://c2bnbljfi9.execute-api.ap-south-1.amazonaws.com/

The developed Cloud-Based Smart File Storage & Sharing System provides a fully functional interface for authenticated file management. The following results demonstrate how the system appears to the user and how each major feature works.

9.1 User Authentication

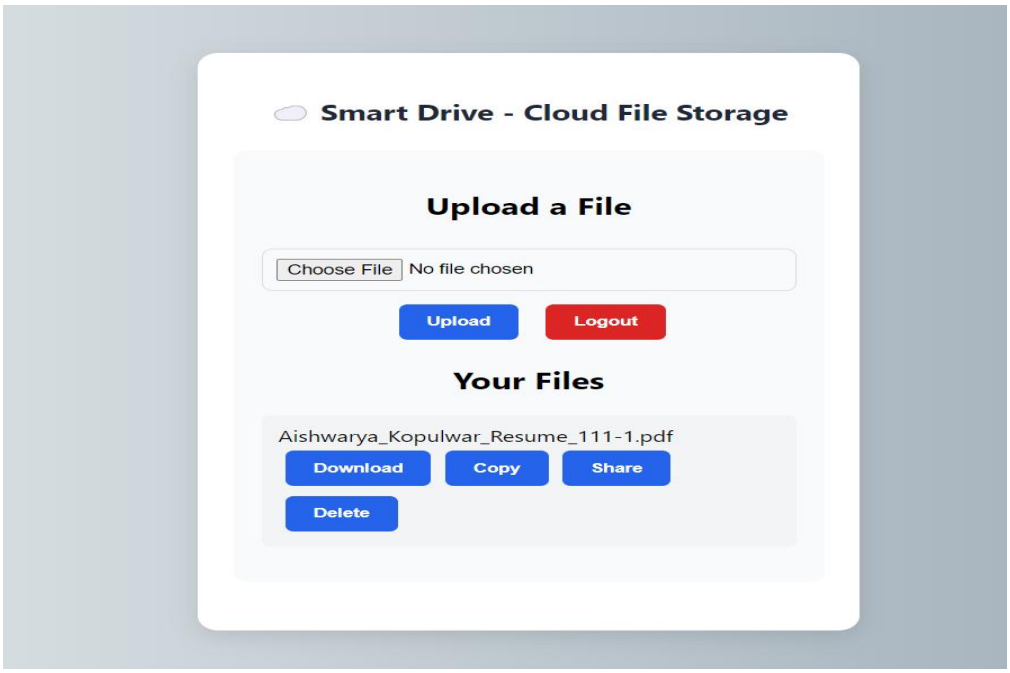
The system begins with a secure login/signup screen.
Users must authenticate before accessing the dashboard.

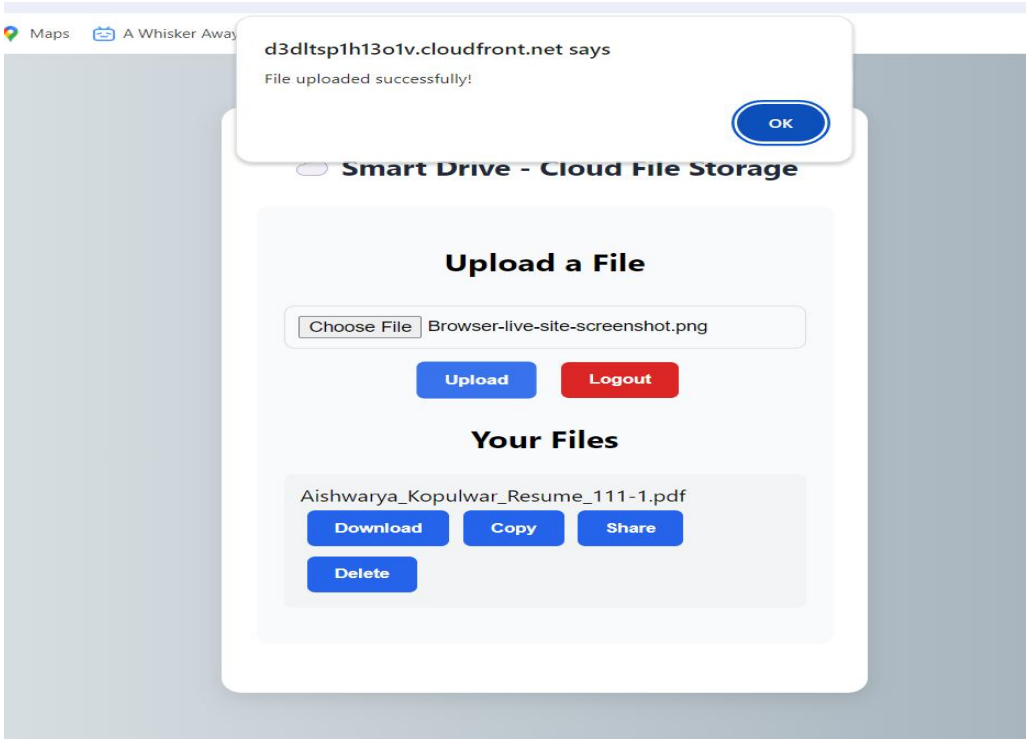




9.2 File Upload

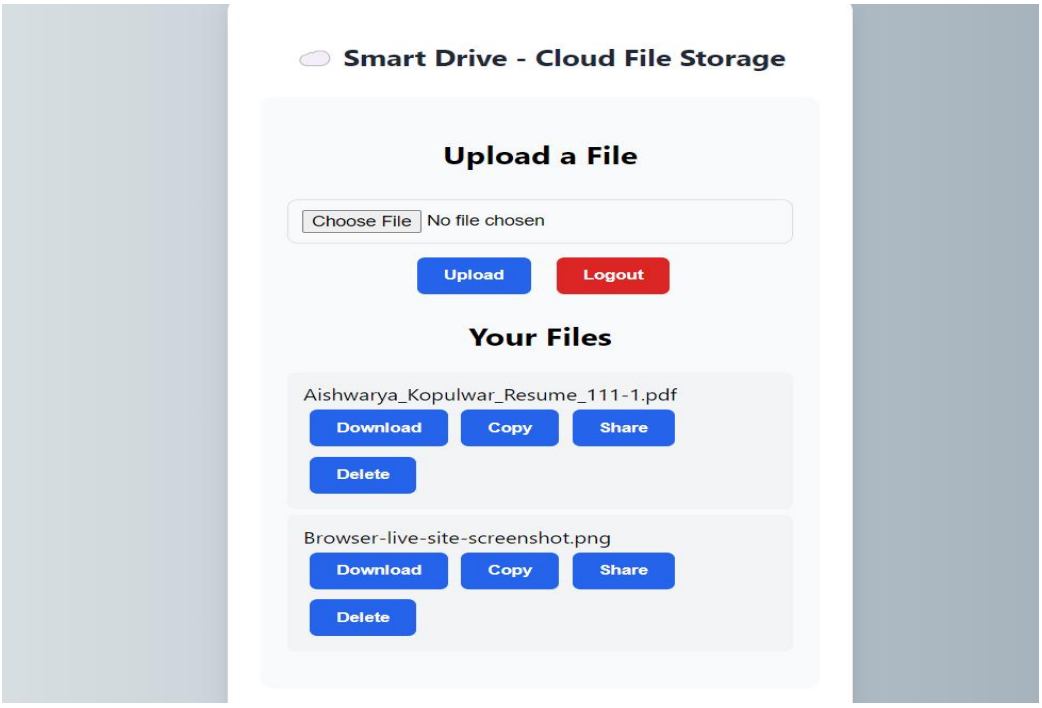
After logging in, users are redirected to the dashboard where they can upload files of any type.
On successful upload, the user receives a confirmation message, and the file is displayed in the list.





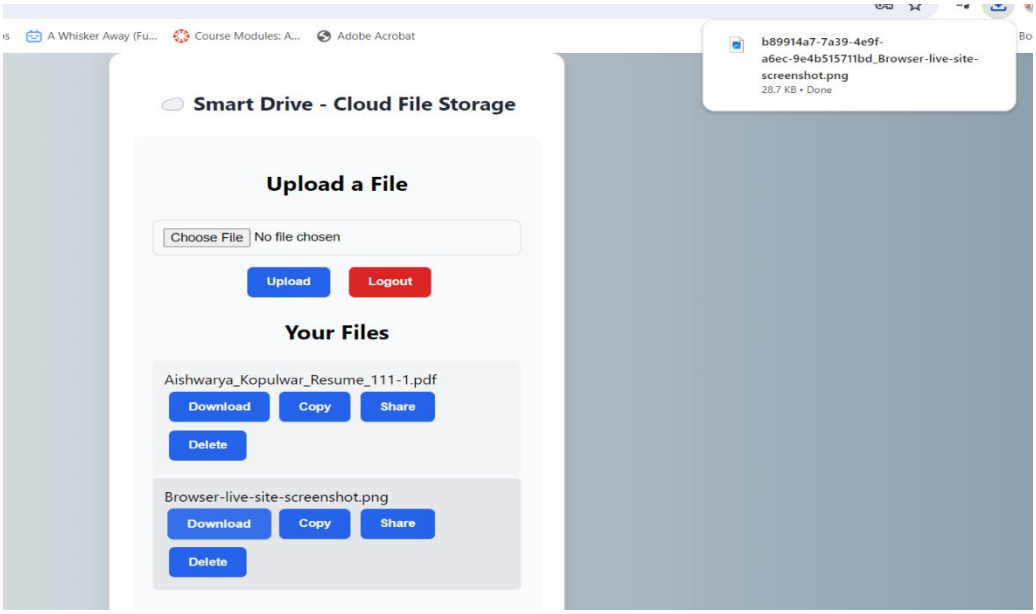
9.3 File Listing Dashboard

Uploaded files appear under Your Files, showing filename and available actions.



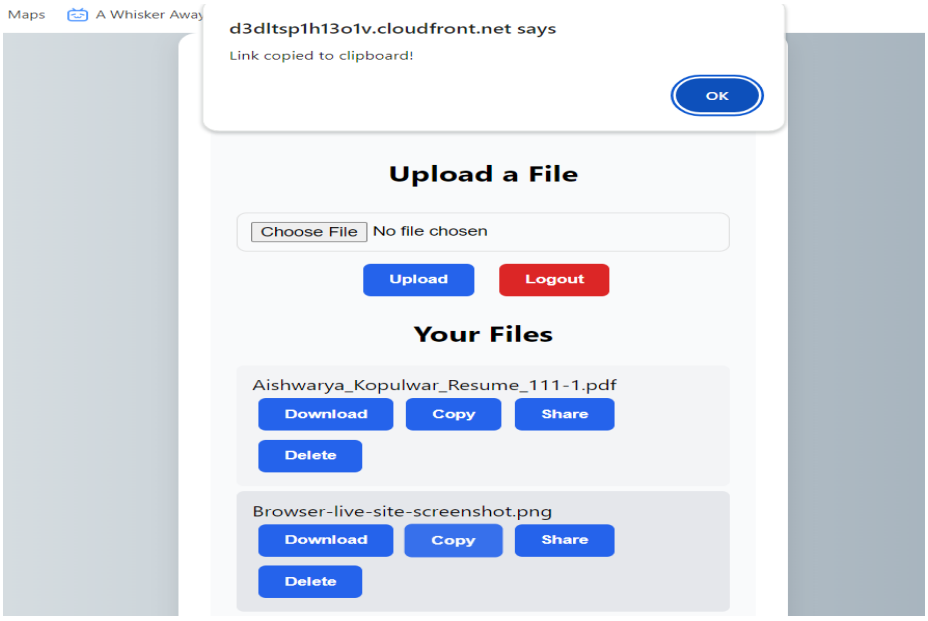
9.4 File Download

Users can download their uploaded files using secure pre-signed URLs.



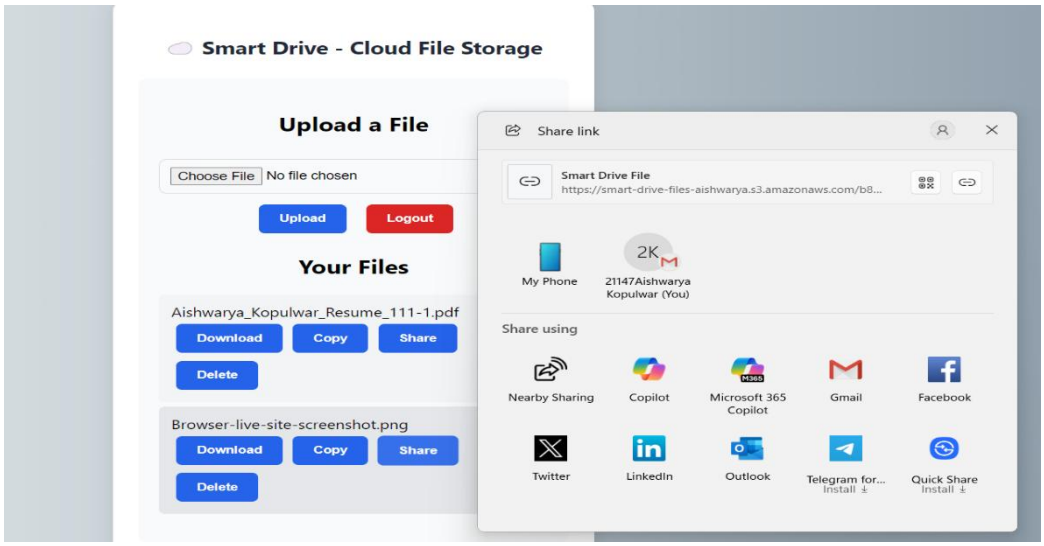
9.5 Copy Link

The "Copy" button instantly copies the secure file URL to the clipboard for easy sharing.



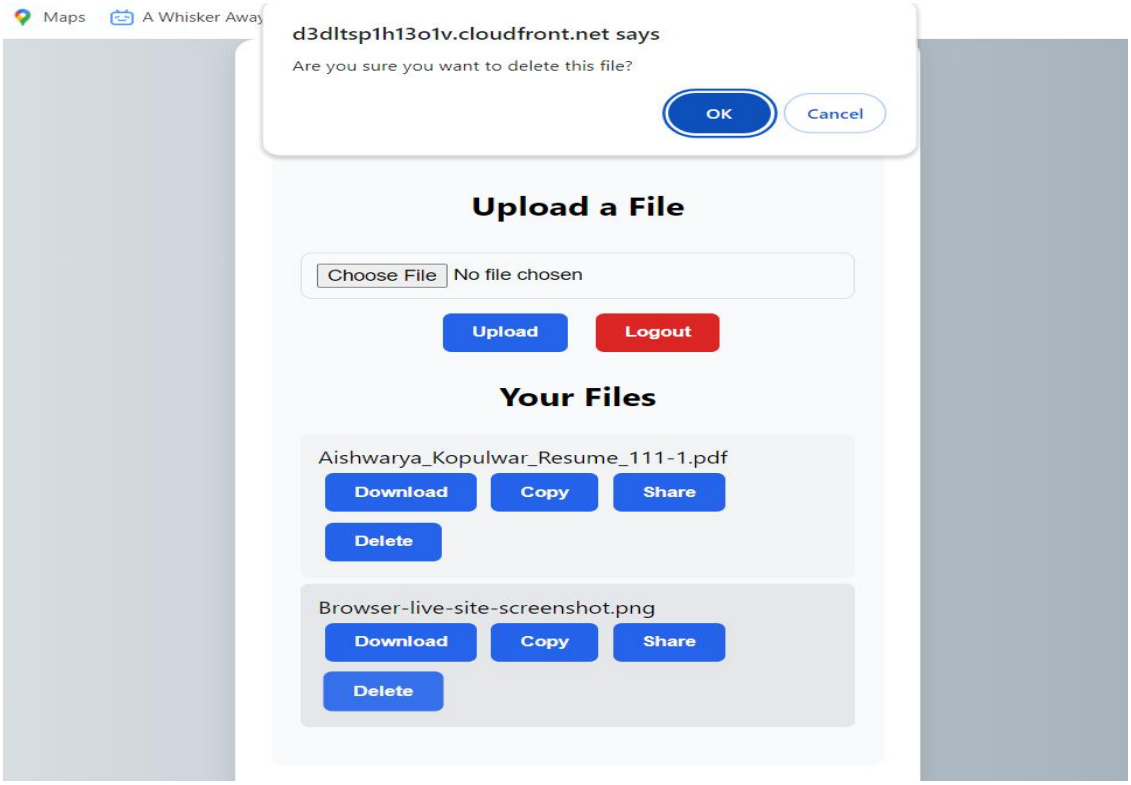
9.6 Share File

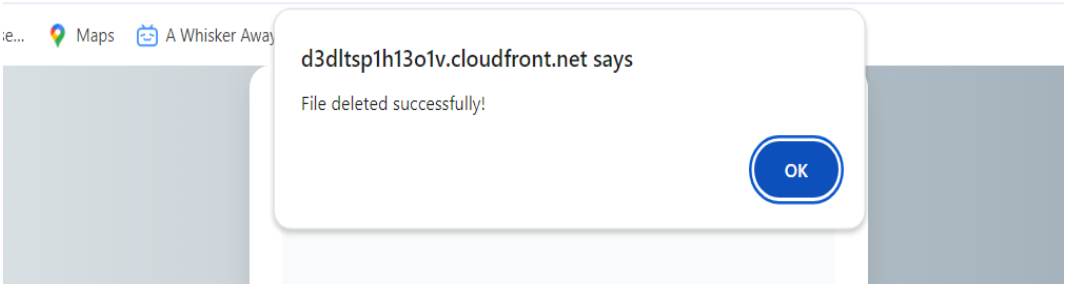
The "Share" button opens the system's native share dialog, allowing sharing through Gmail, LinkedIn, WhatsApp, etc.



9.7 File Deletion

The user can delete files, and the system confirms both the action and the successful deletion.




Maps
A Whisker Away
d3dltsp1h13o1v.cloudfront.net says
File deleted successfully!
OK

10. Conclusion

This project deepened my understanding of how cloud platforms can be used to build real-world, scalable applications. I learned how to integrate multiple AWS services and build a fully functional web application with authentication and file management.

The project strengthened my skills in:

- Cloud Computing
- Backend Development
- AWS Architecture
- Full-Stack Integration
- Secure Web Application Design

11. Future Enhancements

- Email-based file sharing
- Folder system for organization
- File previews (images, PDFs, videos)
- Password hashing for improved security
- User profile and settings page
- File versioning
- Usage analytics and download counters

12. References

- AWS Official Documentation
- Flask Official Documentation
- JWT Documentation
- W3Schools (HTML/CSS/JS)