Link-Load

(Secure Upload & Accessing Platform)

Project Report Submitted
in partial fulfillment for the award of

Master of Computer Applications

By

Anvesh Avinash Gawhane

Master of Computer Application

Roll Number (2024011090120)

Department of Master of Computer Applications
Jawaharlal Nehru Engineering College
MGM University, Chh. SambhajiNagar (Oct- Nov 2025)

PROJECT APPROVAL SHEET

This project report entitled Link-Load by Anvesh Avinash Gawhane is approved for the partial fulfillment of the award of SYMCA degree of MGM University, Chh. SambhajiNagar (M.S.)

Project In charge

____

Dr. Sheetal M Chavan

External Examiner

____

____

Date: / /

Place: Jawaharlal Nehru Engineering College, MGM University Chh. SambhajiNagar (M.S.) – 431005

CERTIFICATE

This is to be certify that, the project work

“Lik-Load”

Submitted by

Anvesh Avinash Gawhane

is a bonafide work completed under my supervision and guidance in partial fulfilment for award of Master of Computer Applications degree of MGM University and is being submitted to Jawaharlal Nehru Engineering College, Chh.SambhjiNagar.

Prof. **Ashwini Nandure** — **Dr. Sonal S. Deshmukh**

Project Guide — **HOD MCA**

Dr. V.B. Musande

Principal

Jawaharlal Nehru Engineering College, Chh.SambhjiNagar (M.S.).
MGM University

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Anvesh Avinash Gawhane

Contents

List of abbreviations : **Sample**

ABBREVATION	FULL SPELLING
Admin	Administrator
Amt	Amount
c/s	Client/ server
B/S	Browser/Server
S/w	Software
h/w	Hardware
m/c	Machine

List of Figures:

Fig. No.	Name of figure	Page No.
1	Deployment Diagram	4
2	Activity Diagrams	6-17
3	Use Case Diagrams	18-25
4	Sequence Diagram	26-32

1. INTRODUCTION

In the modern digital age, data has become the most valuable asset for individuals, organizations, and businesses. Managing this data efficiently, securely, and in a scalable manner is one of the biggest challenges faced by the IT industry. The rapid evolution of cloud computing has revolutionized the way digital data is stored, processed, and accessed. Users no longer need to depend on local storage or dedicated servers — instead, they can access data from anywhere.

The project **“Link-Load”** is a web-based cloud platform designed to provide a secure uploading and accessing system for users who want to store and manage their files online. It is built using Amazon Web Services (AWS), a leading cloud computing platform, which ensures reliability, data protection, and global accessibility. The main aim of Link-Load is to create a user-friendly and serverless environment where users can upload, download, and delete files securely, while all data is automatically synchronized with cloud storage and a metadata database.
The system integrates major AWS services like Amazon S3 (Simple Storage Service) for storing files, Amazon DynamoDB for storing file information, AWS Lambda for backend serverless processing, Amazon Cognito for secure user authentication, and API Gateway for linking the frontend with backend services. This integration provides a fully automated, scalable, and cost-efficient architecture.

In today’s context, where data breaches and cyberattacks are common, Link-Load emphasizes security and reliability. The system ensures that only authenticated users can access or modify their data. Moreover, it promotes an eco-friendly digital solution by reducing the need for physical storage systems and traditional server infrastructure.

1.1 Background

With the rapid advancement of technology, the need for reliable data storage systems has grown significantly. Traditional methods of data storage, such as local hard drives, USBs, and centralized servers, often face limitations like restricted access, higher maintenance costs, risk of data loss, and lack of scalability. As organizations and individuals produce enormous volumes of digital content every day, traditional systems fail to offer the flexibility and global accessibility required in modern computing.

Cloud computing emerged as the ideal solution, offering on-demand access to computing resources through the internet. It allows users to store, manage, and retrieve their files without depending on physical infrastructure. Cloud-based platforms like AWS, Google Cloud, and Microsoft Azure have transformed how users handle data, making it possible to create highly efficient and serverless systems.

In this context, the Link-Load project is developed as a cloud-based file storage system that addresses the issues of traditional systems. It combines data security, scalability, and cost-effectiveness using AWS services. The system architecture uses AWS Lambda functions to handle backend processes without any physical servers, ensuring minimal operational cost. User authentication is handled by Amazon Cognito, while files and their metadata are securely stored in S3 and DynamoDB respectively.

This system benefits both individual users and small organizations by providing a reliable digital storage platform that is accessible from any location and device. In short, Link-Load bridges the gap between user convenience and secure cloud computing technology.

1.2 Objectives

The primary objective of Link-Load is to design and implement a secure, serverless, and cloud-based file management system that allows users to upload, access, and manage their files efficiently. The system's design focuses on achieving seamless integration between frontend and cloud services using AWS.

The specific objectives of the project are as follows:

1. To provide **user authentication and authorization** through Amazon Cognito, ensuring only valid users can access their data.

2. To implement **secure file uploads** to Amazon S3 using temporary credentials, ensuring data privacy and preventing unauthorized access.

3. To **store and manage file metadata** such as name, type, size, and upload date using Amazon DynamoDB.

4. To develop a **responsive and interactive web interface** using HTML, CSS, and JavaScript for easy user interaction.

5. To establish **API integration** through AWS API Gateway and Lambda functions for communication between frontend and backend.

6. To allow users to **view, download, and delete their uploaded files** directly through the web interface, with automatic reflection in S3 and DynamoDB.

7. To demonstrate the use of **serverless architecture**, improving system performance, scalability, and reliability without the need for manual server management.

8. To ensure **data integrity and availability**, reducing the risks of data loss or corruption. The project thus full-fills both academic and practical goals by applying cloud computing concepts in a real-world use case.

1.3 Purpose

The purpose of developing Link-Load is to provide an innovative and practical solution for secure cloud-based file management. In an era where digital transformation is reshaping industries, individuals and organizations require fast, secure, and reliable access to their data. Traditional file storage methods often lead to problems such as accidental deletion, hardware failure, limited accessibility, and unauthorized access. The Link-Load system solves these challenges by leveraging AWS's serverless architecture, which provides elasticity, scalability, and reduced operational cost.

The project also serves as a learning and implementation model for understanding real-world cloud deployment. It demonstrates how various AWS components can work together seamlessly to form a complete software ecosystem. For students and developers, this project highlights the benefits of adopting modern technologies such as API Gateway, Lambda, S3, and DynamoDB, and teaches how to secure cloud applications using Cognito authentication.
In a broader sense, the purpose is to empower users with a digital platform where they can store and manage their data without worrying about infrastructure. The project's significance lies in promoting data protection, efficient storage, and seamless cloud interaction — making it a step toward sustainable digital transformation.

1.4 Scope

The Link-Load project focuses on creating a comprehensive, cloud-integrated system that can be extended and scaled in the future. The system's scope covers design, development, and deployment on AWS infrastructure using modern web technologies.
The major components included within the project's scope are:

- **Frontend Interface:** Designed using HTML, CSS, and JavaScript, hosted on Amazon S3 as a static website for user interaction.
- **Authentication Module:** Integration of Amazon Cognito for secure user registration, login, and session management.
- **File Management:** Upload, retrieve, delete, and access files from S3 storage via signed URLs using secure AWS SDK methods.
- **Database Management:** Storing metadata (file name, type, description, upload time, etc.) in Amazon DynamoDB for structured access.
- **Serverless Processing:** AWS Lambda functions used for saving metadata, retrieving files, and deleting records without any dedicated servers.
- **API Integration:** AWS API Gateway acts as a bridge between the frontend and backend Lambda functions, ensuring smooth communication.
- **Security and Compliance:** Ensuring that each action — upload, delete, or access — is authorized by Cognito and logged for accountability.

-The system currently targets **individual users** and can later be expanded to support multiple user roles, collaborative sharing, and analytics. It provides a real-world demonstration of building an efficient **serverless cloud application** using industry-standard tools.

2. EXISTING SYSTEM

2.1 Existing System

In existing file management systems, users usually store and share data through local drives, USB devices, shared networks, or basic cloud storage like Google Drive or Dropbox. These platforms provide only limited control, low customization, and often depend on centralized servers that need maintenance and monitoring.
Most organizations use traditional client–server models, where users access files from internal servers. Such systems are often restricted to specific networks, making remote access difficult. Moreover, scalability is limited -when data increases, performance drops unless hardware is upgraded.
These systems lack automation, security, and integration with cloud technologies. Users cannot track uploads, analyze usage, or ensure strong data protection. As digital collaboration grows, these limitations make traditional storage inefficient and outdated.

2.2 Disadvantages

1. Limited Accessibility: Users must rely on local networks or manual file transfers.
2. High Maintenance Cost: Physical servers require frequent upgrades and monitoring.
3. Low Security: Files are often unencrypted and lack authentication control.
4. Poor Scalability: Storage cannot expand automatically with data growth.
5. Manual Backup: Data recovery and version control are unreliable.
6. Lack of Cloud Integration: No real-time synchronization or serverless features.

-Because of these issues, existing systems fail to meet modern needs for secure, automated, and globally accessible data management..!

2.3 Proposed System

The proposed project, Link-Load, is a secure cloud-based file uploading and accessing platform developed using Amazon Web Services (AWS). It overcomes the drawbacks of existing systems through a serverless and scalable design.

Files uploaded by users are stored in Amazon S3, while their details (metadata) are saved in Amazon DynamoDB. AWS Lambda handles backend operations such as saving and deleting metadata through API Gateway routes, and Amazon Cognito manages secure user authentication.

The system's interface, built with HTML, CSS, and JavaScript, allows users to easily upload, access, or delete files categorized as images, videos, documents, or audio. Any changes made on the frontend automatically reflect in AWS services.

Key Benefits:

- Secure and authenticated cloud access
- Automatic scalability and no maintenance
- Real-time synchronization between storage and database
- Cost-efficient, pay-per-use architecture

Thus, **Link-Load** transforms traditional storage into a modern, fully automated cloud solution, combining security, accessibility, and efficiency.

3. SYSTEM DEVELOPMENT

The system development of Link-Load involves designing, implementing, and deploying a secure, serverless file uploading and accessing platform using Amazon Web Services (AWS). The project uses a combination of web technologies (HTML, CSS, JavaScript) for the frontend and AWS cloud services for backend operations like authentication, data storage, and metadata management.

The development process includes:

- Frontend Design: Creating responsive web pages for file upload, user login, and data access.
- Backend Logic: Using AWS Lambda for serverless functions to handle metadata operations.
- Database Management: Storing file metadata securely in DynamoDB.
- Cloud Integration: Connecting all AWS services (S3, Cognito, API Gateway, Lambda, DynamoDB).
- Testing and Deployment: Hosting the web interface through S3's static website hosting.

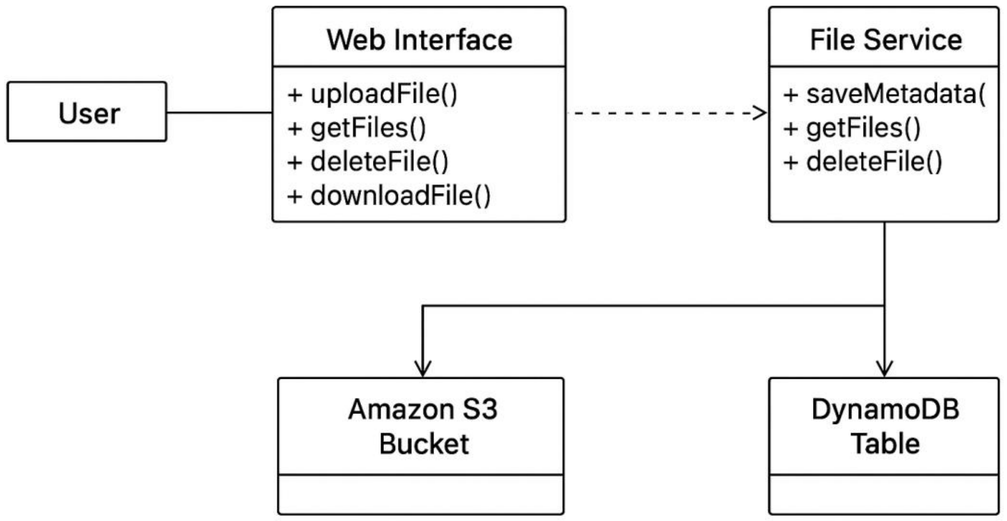
Hardware Requirements

Component	**Specification-**
Processor	Intel i3/i5 or equivalent (2.0 GHz or above)
RAM	Minimum 4 GB (8 GB recommended)
Storage	Minimum 20 GB free disk space
Display	1366×768 resolution or higher
Internet	Stable broadband connection for AWS connectivity
Peripherals	Keyboard, mouse, and optional external storage

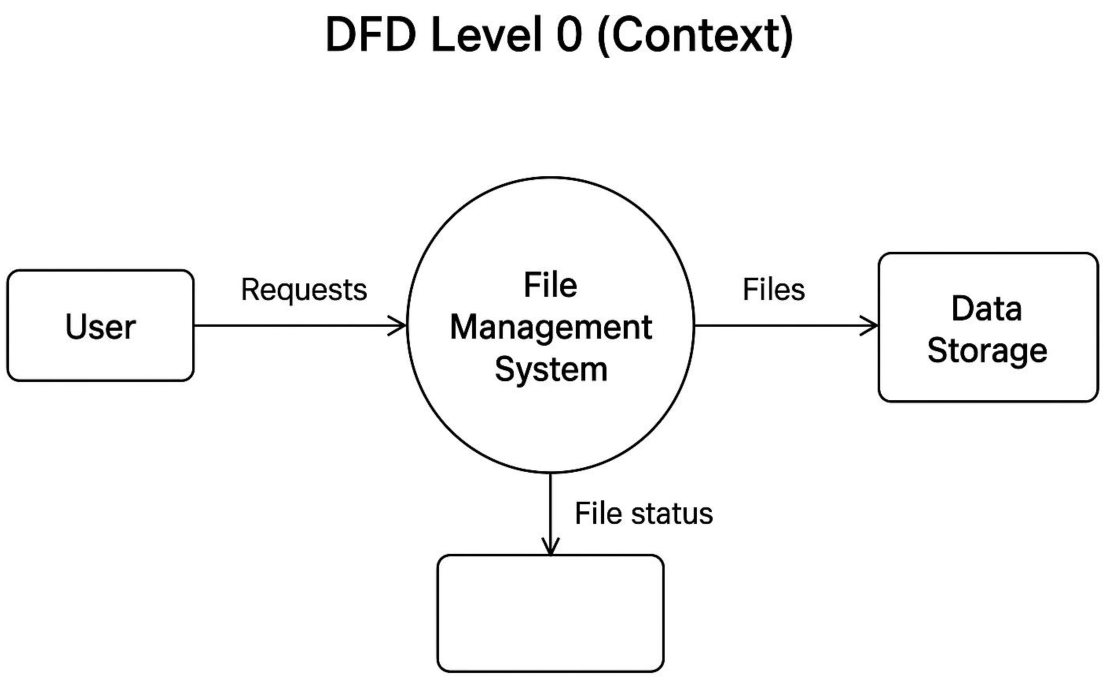
Software Requirements

Category	Software / Service	Purpose
Operating System	Windows 10 / 11	Development environment
Code Editor	Visual Studio Code	Writing and editing HTML, CSS, JS, and Lambda functions
Frontend Languages	HTML5, CSS3, JavaScript	Web interface design and logic
Backend Framework	AWS Lambda (Node.js Runtime 22.x)	Serverless backend functions
Database Service	Amazon DynamoDB	Stores uploaded file metadata
Storage Service	Amazon S3	Stores actual uploaded files and static web pages
Authentication	Amazon Cognito	Manages user sign-up, login, and secure access
API Management	AWS API Gateway	Connects frontend with Lambda backend securely
Cloud Monitoring	Amazon CloudWatch	Tracks function logs and performance
Web Hosting	S3 Static Website Hosting	Hosts index.html, access.html, and related assets
Testing Tools	Browser Console & AWS Lambda Test Events	Debugging and validating API requests

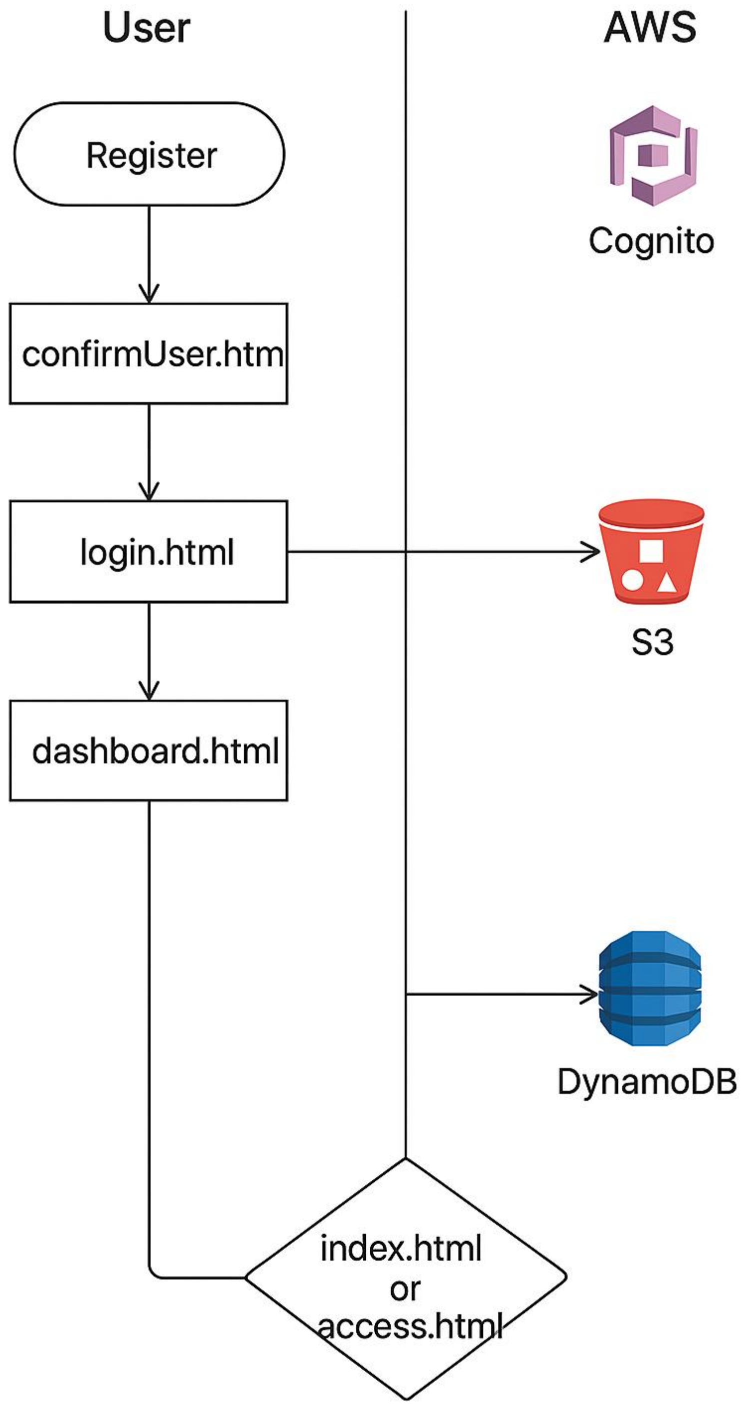
UML Diagram



DFD level 0 (context) Diagram :-



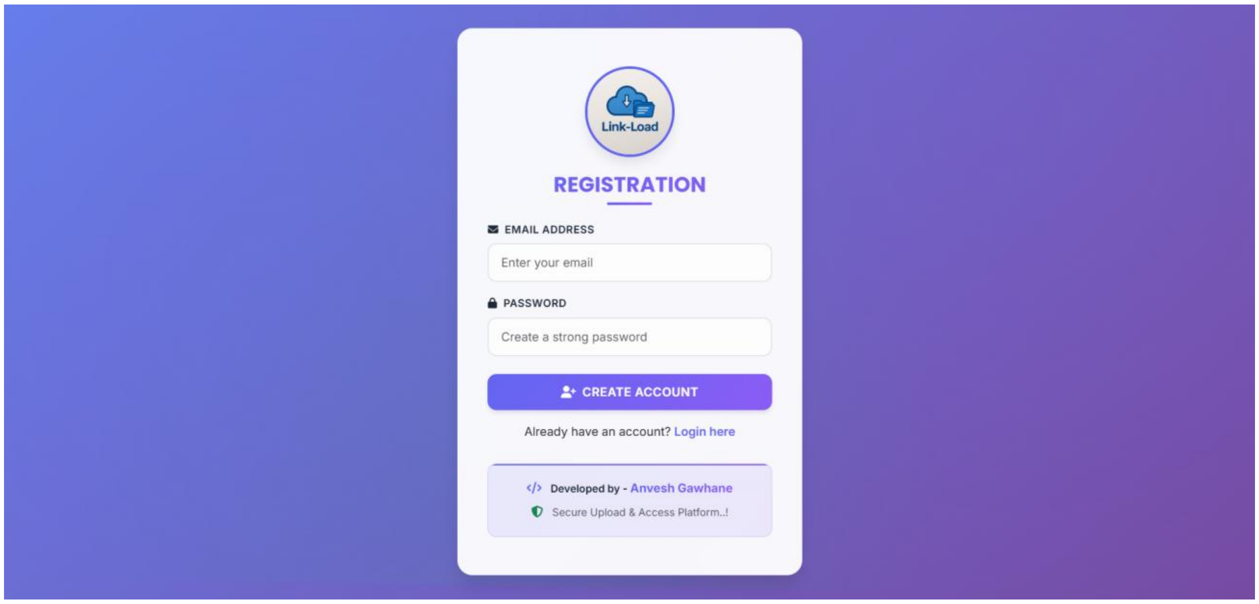
Activity Diagram :-



4. PERFORMANCE ANALYSIS

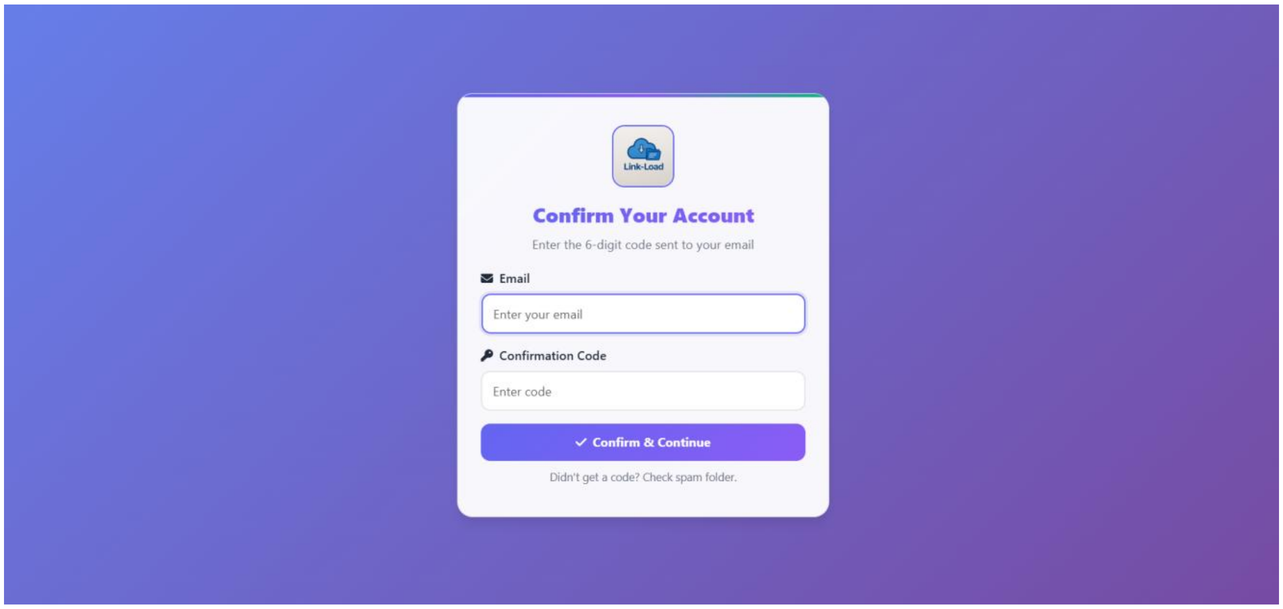
- **Register.html Page :-**

This page allows a new user to create an account in the Link-Load system using AWS Cognito authentication. The registration form securely collects user details and initiates the verification process. Once the user submits the form, Cognito sends a confirmation code to their registered email, ensuring secure and verified access.


Link-Load
REGISTRATION
EMAIL ADDRESS
Enter your email
PASSWORD
Create a strong password
CREATE ACCOUNT
Already have an account? Login here
Developed by - Anvesh Gawhane
Secure Upload & Access Platform..!

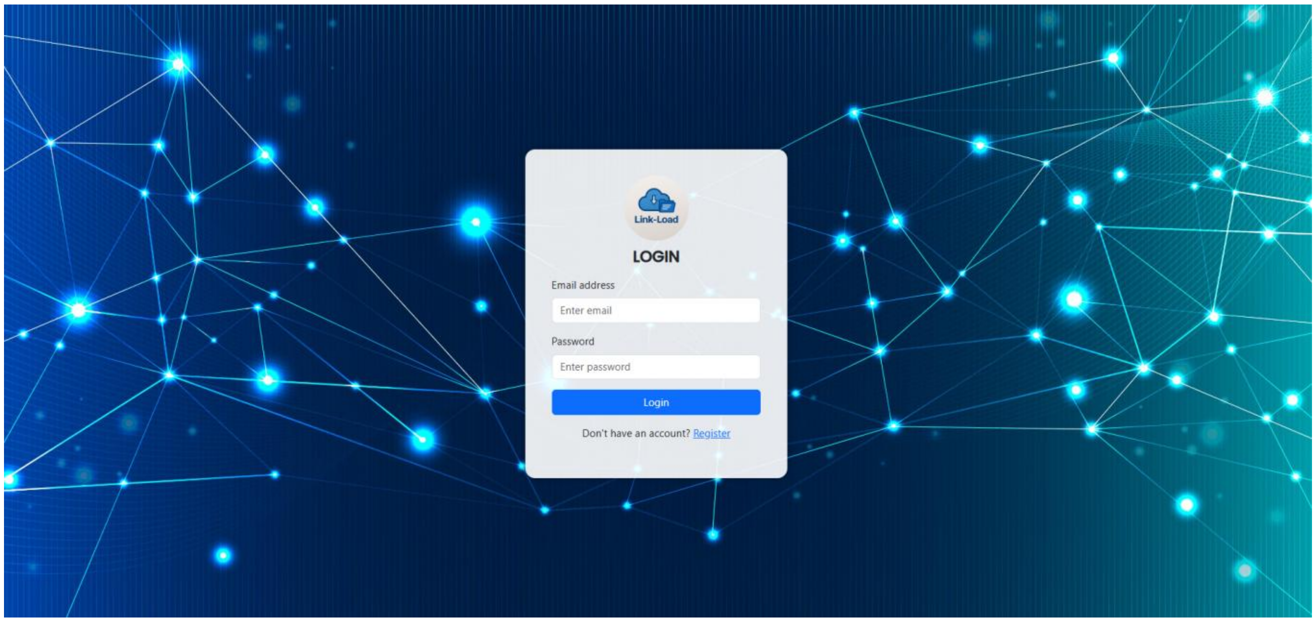
- **ConfirmUser.html Page :-**

After registration, the user is directed to this confirmation page to verify their account using the OTP sent via email. This ensures that only authorized and valid users gain access to the system. It uses AWS Cognito APIs to confirm user identity and activate their account for login.



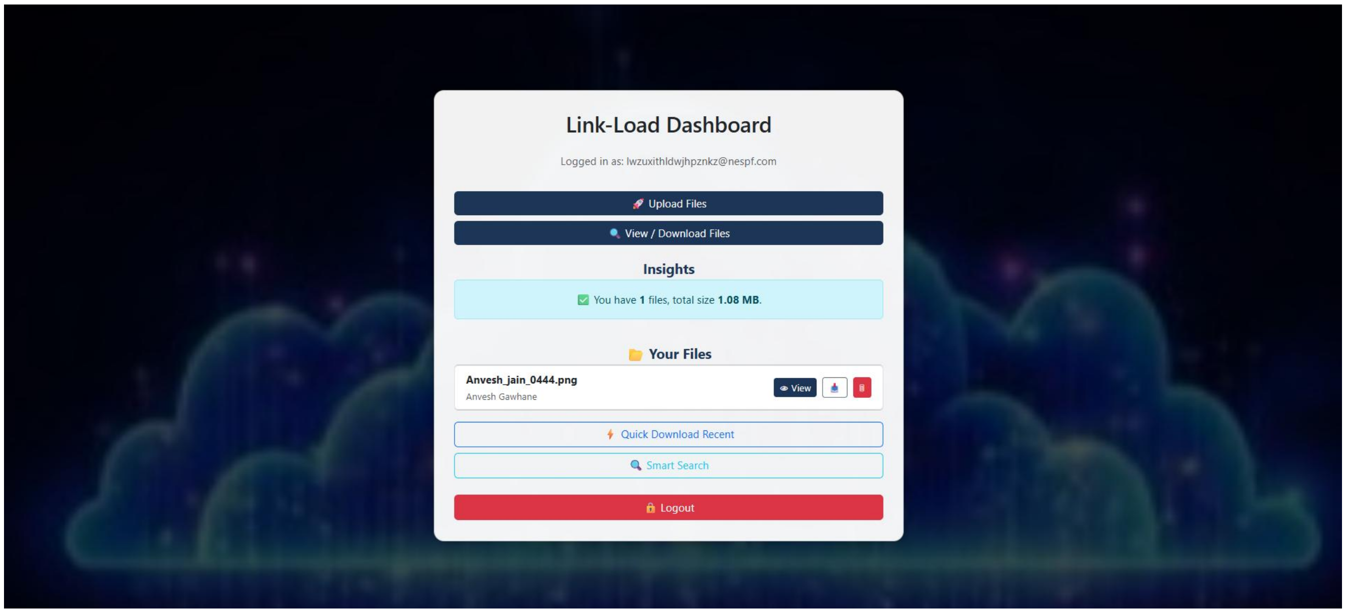
- **Login.html Page :-**

This page provides the secure login interface for users to access their accounts. It authenticates credentials through AWS Cognito and generates session tokens for authorized access. Once validated, users are redirected to the dashboard for file management operations.



- **Dashboard.html Page :-**

The dashboard acts as the main control panel for the user after successful login. It displays key navigation options such as file upload, file access, and logout. The page provides a simple and intuitive design for managing and monitoring user activities within the Link-Load system.


Link-Load Dashboard

Logged in as: lwzuxithldwjhpznkz@nespf.com

Insights

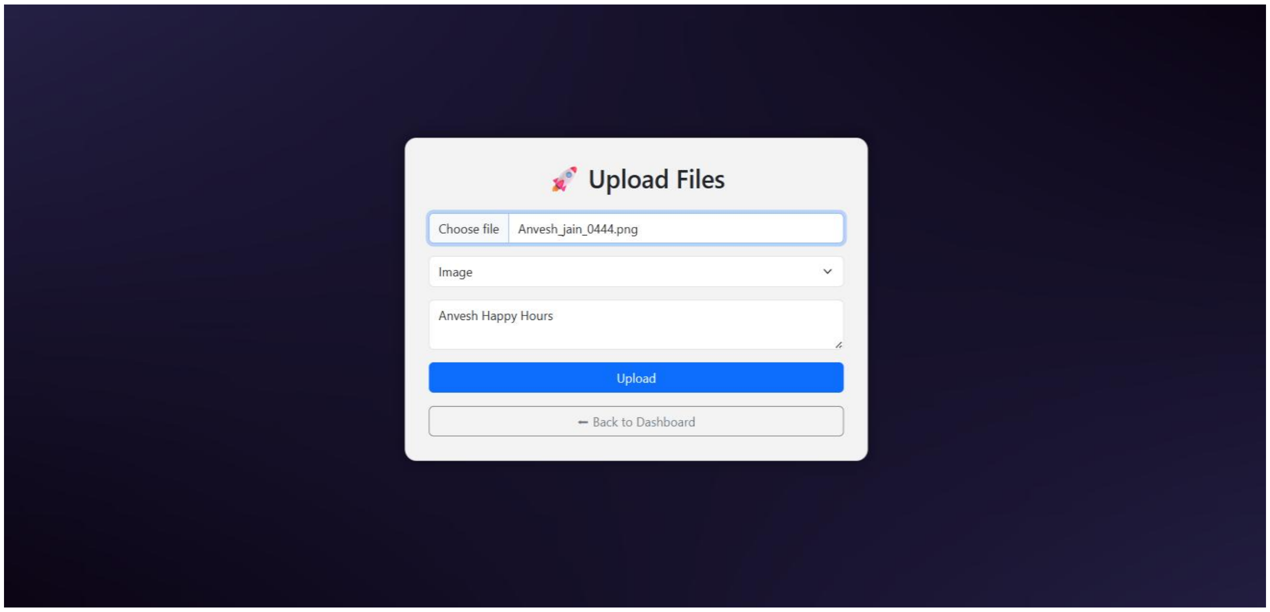
✅ You have **1** files, total size **1.08 MB**.

📁 Your Files

Anvesh_jain_0444.png
Anvesh Gawhane

- **Index.html Page :-**

This is the main file upload page of the Link-Load system. Users can select files, specify their type (image, video, audio, document), and add descriptions before uploading. The page connects with AWS S3 via Lambda and API Gateway to securely store uploaded files in the cloud, showing progress and AI-based file analysis feedback.


Upload Files

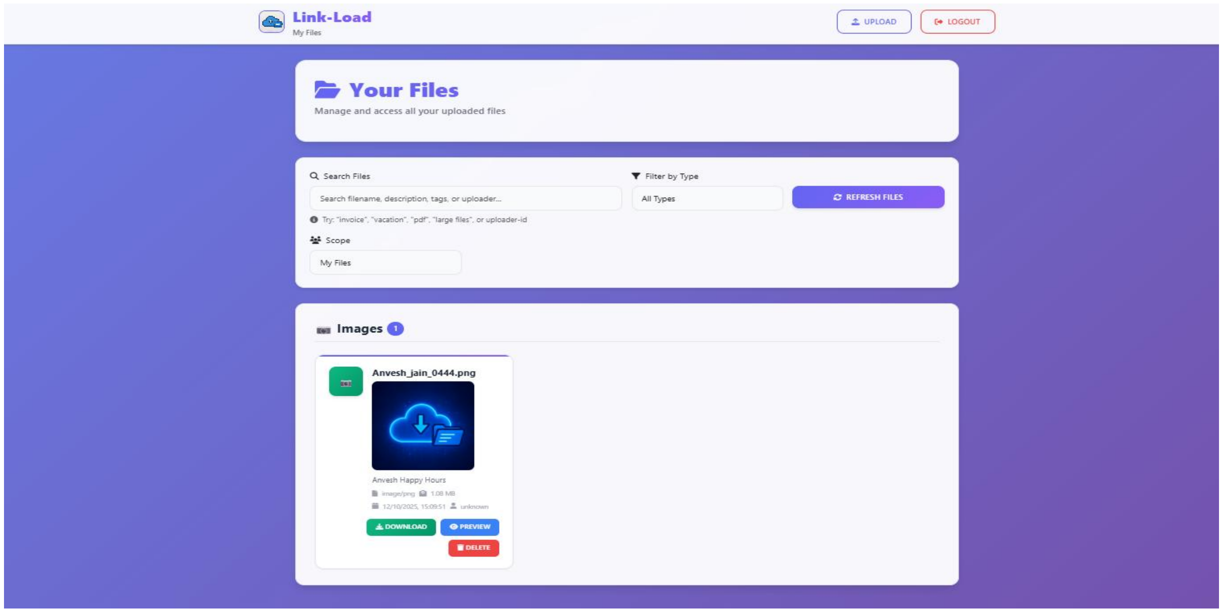
Choose file: Anvesh_jain_0444.png

Image

Anvesh Happy Hours

- **Access.html Page :-**

This page enables users to view, download, and delete their uploaded files. It retrieves file metadata from AWS DynamoDB using API Gateway and Lambda functions. The page categorizes data by type and displays detailed information like file size, upload date, and description, ensuring full user control over stored content.


Link-Load
My Files

Your Files
Manage and access all your uploaded files

Search Files: Search filename, description, tags, or uploader...
Filter by Type: All Types
Try: "invoice", "vacation", "pdf", "large files", or uploader-id
Scope: My Files

Images 1

Anvesh_jain_0444.png
Anvesh Happy Hours
image/png 1.08 MB
12/10/2025, 15:09:51 unknown

Sample Lambda Function Code (index.mjs) :-

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, PutCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({ region: "ap-southeast-2" });
const docClient = DynamoDBDocumentClient.from(client);

export const handler = async (event) => {
  try {
    const body = JSON.parse(event.body);
    const { userId, fileName, fileType, description, size, uploadDate, s3Key, s3Url } = body;

    // Prepare item for DynamoDB
    const item = { userId, fileName, fileType, description, size, uploadDate, s3Key, s3Url };

    // Save metadata to DynamoDB table
    await docClient.send(new PutCommand({
      TableName: "LinkLoadFiles",
      Item: item
    }));

    return {
      statusCode: 200,
      body: JSON.stringify({ message: "File metadata saved successfully!" }),
    };
  } catch (error) {
    return {
      statusCode: 500,
      body: JSON.stringify({ error: error.message }),
    };
  }
};
```

5. CONCLUSION

5.1 Conclusion

In conclusion, this project bridges the gap between secure file management and user-friendly accessibility. By leveraging AWS services, the system eliminates traditional server dependencies and manual configuration efforts. The implementation of AWS Cognito ensures that only verified users can access the system, while S3 bucket policies and Lambda-based logic provide strong data protection.

"Link-Load" reflects how modern cloud solutions can empower individuals and organizations to handle large volumes of data with reliability, low latency, and minimal maintenance. The project outcome validates that serverless architectures are the future of web-based file handling systems.

5.2 Limitations of the System

While the system is functional and efficient, there are certain limitations identified during testing:

1. The platform depends entirely on AWS internet connectivity — it cannot function offline.
2. File upload speed may vary depending on the user's network quality and file size.
3. The system currently lacks role-based access (e.g., admin vs. normal users).
4. There is limited real-time collaboration or file-sharing functionality between users.
5. AWS service costs may increase if the system scales with high traffic or large data volumes.

5.3 Future Scope of the Project

There is significant potential for extending the "Link-Load" system with additional features and optimizations:

1. Role-Based Access Control (RBAC): Introduce admin-level controls and multi-user permissions.
2. AI-Based File Analysis: Integrate AWS Rekognition or Comprehend for intelligent tagging and automatic content classification.
3. File Sharing Links: Allow users to generate secure temporary links for sharing files externally.
4. Version Control: Implement file version history to restore or track previous uploads.
5. Mobile App Integration: Extend functionality to Android and iOS using AWS Amplify.
6. Cost Optimization: Introduce automatic archiving for unused files using Amazon S3 Glacier.

References :

1. Rajkumar Buyya, Christian Vecchiola, and Thamarai Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*, McGraw Hill Education, 2013.

2. Stuart Scott, *AWS Certified Cloud Practitioner Study Guide*, Sybex, 2021.

3. Ben Piper and David Clinton, *AWS Certified Solutions Architect Official Study Guide*, Wiley, 2020.

4. Genie Ashwani -
youtube link- https://youtu.be/_LIuNAu5Ktc?si=xvWgIlkjmNDdIxMB

ACKNOWLEDGEMENT

The completion of any inter-disciplinary project depends upon cooperation, co-ordination and combined efforts of several sources of knowledge.

I wish to record my heartfelt gratitude and sincere thanks to Prof. Ashwini Nandure Asst. Professor, Department of MCA, for his/her kind support, immense guidance and inspiration given till the end of my project.

I thank Dr.Sonal S Deshmukh Head, Department of MCA, for their constant support and even willingness to give us valuable advice and direction

I take this opportunity to extend my gratitude to Dr. V.B. Musande, Principal, NEC, MGM University for their valuable suggestions.

Last but not least, I also thankful to all the staff members for their immense cooperation and motivation of completing out the project

Anvesh Avinash Gawhane