

Name : Shreyash Santosh Sahu
Sem : 7th Sem CSE-A
Roll no. : 62

TA-1

Aim :

To create a file upload system in Next.js that utilizes the AWS S3 service, enabling users to upload files and maintain a centralized data repository for their needs.

Github repository link : <https://github.com/1-Shreyash/ccta>

PERFORMANCE PICS :

Here's a step-by-step guide to set up AWS S3 for file uploads, including configuring bucket policies and roles.

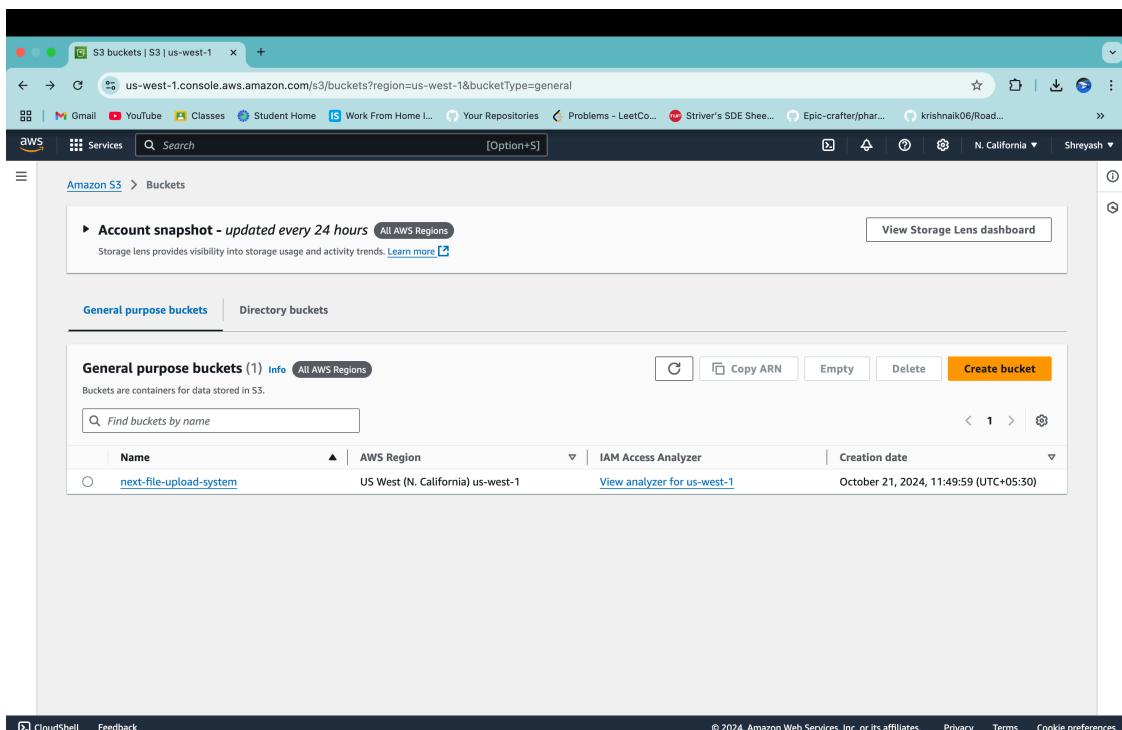
Step 1: Set Up an AWS S3 Bucket

1. Sign in to AWS:

- Go to the [AWS Management Console](#).
- If you don't have an AWS account, create one.

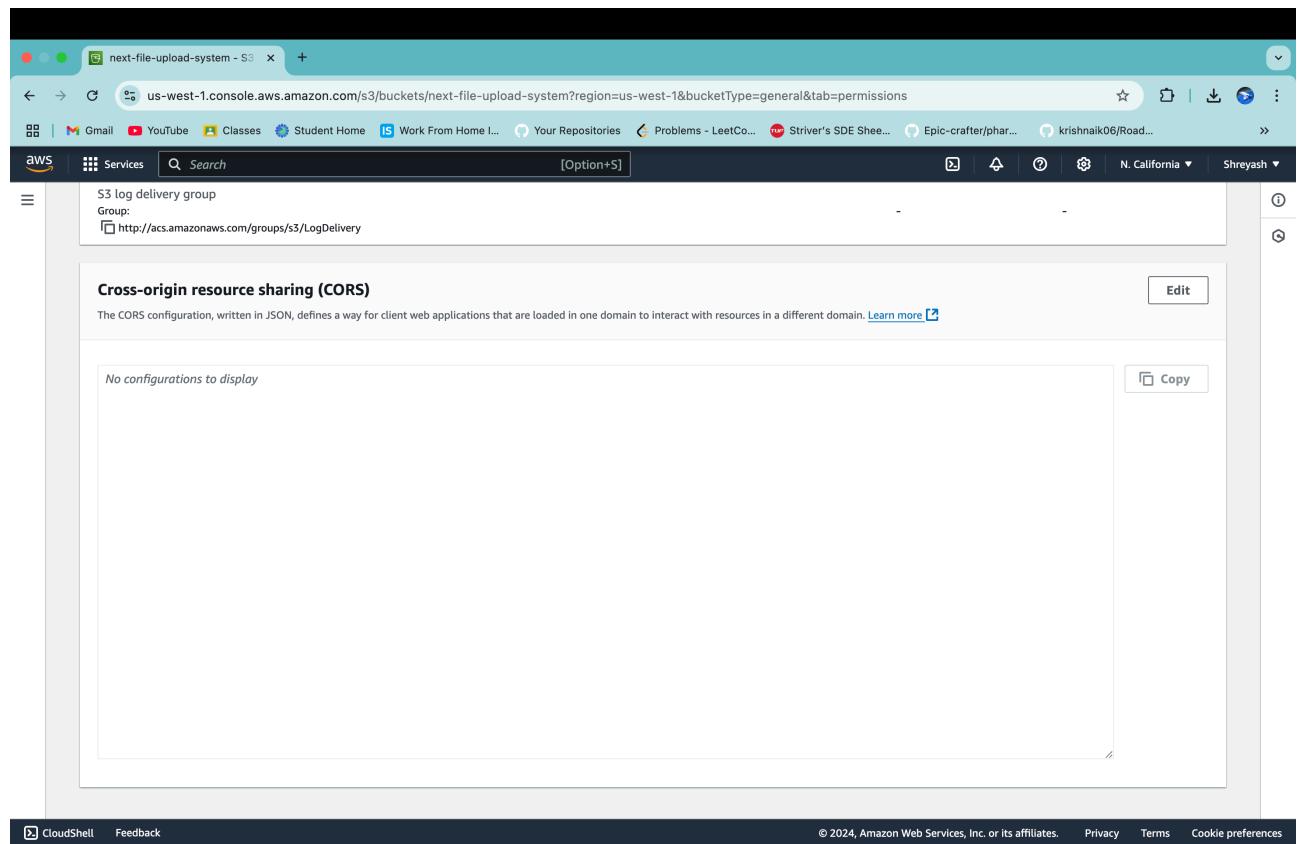
2. Open the S3 Service:

- In the AWS Management Console, search for "S3" in the search bar at the top.
- Click on **S3** to open the S3 dashboard.



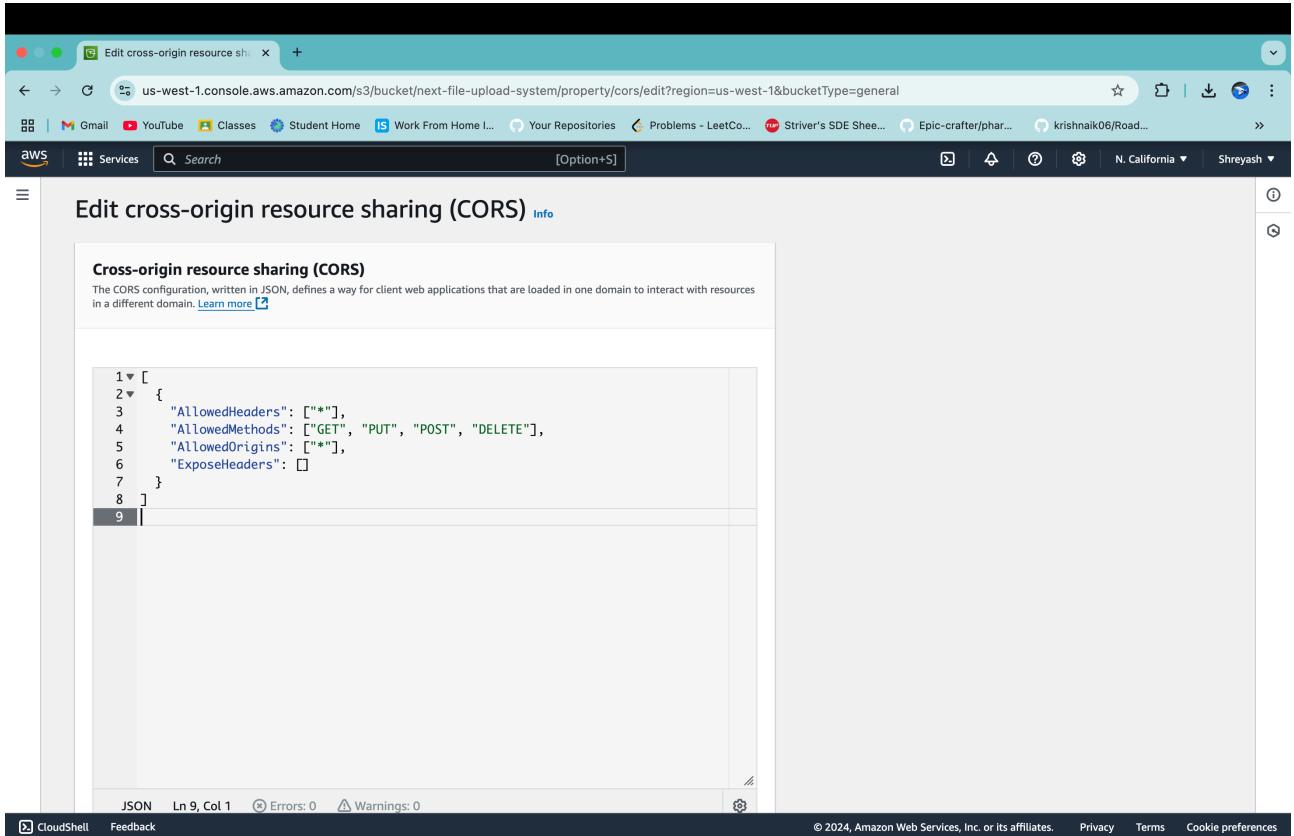
3. Create a New Bucket:

- Click the **Create bucket** button.
- **Bucket Name:** Provide a globally unique name for your bucket (e.g., `my-nextjs-file-uploads`).
- **Region:** Choose the AWS region where you want the bucket to reside (e.g., `us-east-1`).
- **Bucket settings:** You can leave most settings as default.
- **Block Public Access:** Ensure public access is blocked if you don't want files to be publicly accessible. Otherwise, allow public access based on your project needs.
- Click **Create Bucket**.



4. Enable CORS Configuration (if your app makes requests directly to S3):

- In the S3 bucket dashboard, click on your bucket.
- Go to the **Permissions** tab.
- Scroll down to **CORS Configuration** and click **Edit**.
- Add the following CORS policy to allow file uploads:



The screenshot shows the AWS CloudShell interface with a tab titled "Edit cross-origin resource sharing (CORS)". The main content area displays a JSON-based CORS policy configuration. The policy is defined as follows:

```
1▼ [  
2▼ {  
3  "AllowedHeaders": ["*"],  
4  "AllowedMethods": ["GET", "PUT", "POST", "DELETE"],  
5  "AllowedOrigins": ["*"],  
6  "ExposeHeaders": []  
7 }  
8 ]  
9 |
```

Below the code editor, status information is shown: "JSON" and "Ln 9, Col 1". At the bottom of the interface, there are links for "CloudShell", "Feedback", "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

Click **Save changes**.

Step 2: Set Bucket Policy

To allow uploading files to the S3 bucket, configure the bucket policy.

1. Navigate to Bucket Policy:

- On your bucket's **Permissions** tab, scroll down to the **Bucket policy** section.
- Click **Edit**.

2. Add Bucket Policy:

- Use the following JSON policy template to grant public access or restrict access based on your needs. Here's an example that allows uploading from specific IAM roles or users:

The screenshot shows the 'Edit bucket policy' page in the AWS Management Console. The URL is <https://us-west-1.console.aws.amazon.com/s3/bucket/next-file-upload-system/property/policy/edit?region=us-west-1&bucketType=general>. The page title is 'Edit bucket policy'. The policy content is a JSON template:

```
1▼ {
2  "Version": "2012-10-17",
3  "Statement": [
4    {
5      "Effect": "Allow",
6      "Principal": "*",
7      "Action": "s3:PutObject",
8      "Resource": "arn:aws:s3:::your-bucket-name/*"
9    }
10  ]
11}
12|
```

On the right, there's a sidebar with 'Edit statement' and 'Select a statement' sections, and a button '+ Add new statement'.

Replace `your-bucket-name` with your actual bucket name.
This policy allows all users to upload objects. You can restrict this by specifying specific users/roles under "Principal" if needed.

The screenshot shows the 'Create user' wizard in the AWS Management Console. The URL is <https://us-east-1.console.aws.amazon.com/iam/home?region=us-west-1#/users/create>. The step is 'Step 1: Specify user details'. The 'User details' section shows a 'User name' field with 'file-upload-user' entered. A note says: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = . @ _ - (hyphen)'. There's an optional checkbox 'Provide user access to the AWS Management Console - optional' with a note: 'If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.' At the bottom are 'Cancel' and 'Next' buttons.

Save the Policy.

Step 3: Set Up IAM Roles and Permissions

To securely upload files, create a role or user with access to the S3 bucket.

1. Go to IAM Service:

- In the AWS Console, search for **IAM** (Identity and Access Management).

2. Create a New IAM User or Role:

- Click on **Users** (for user-specific access) or **Roles** (for service access like Lambda, EC2, etc.).
- Click on **Add User** or **Create Role**.

The screenshot shows the AWS IAM service interface. On the left, there's a navigation sidebar with sections like 'Identity and Access Management (IAM)', 'Dashboard', 'Access management' (with 'User groups', 'Users', 'Roles', 'Policies', 'Identity providers', and 'Account settings'), 'Access reports' (with 'Access Analyzer', 'External access', 'Unused access', 'Analyzer settings', 'Credential report', 'Organization activity', and 'Service control policies'), and 'CloudShell' and 'Feedback' buttons at the bottom. The main content area is titled 'Users (1) Info' and contains a table with one row for 'file-upload-user'. The table columns include 'User name' (with a checkbox), 'Path' (containing '/'), 'Group' (with a dropdown menu), 'Last activity' (with a timestamp), 'MFA' (with a dropdown menu), 'Password age' (with a dropdown menu), and 'Console last used' (with a dropdown menu). There are also 'Create user' and 'Delete' buttons at the top right of the table area. The URL in the browser is 'us-east-1.console.aws.amazon.com/iam/home?region=us-west-1#/users'.

3. Set Permissions:

- For the user/role, under **Permissions**, attach the following policy:
 - Click on **Attach existing policies directly**.
 - Search for **AmazonS3FullAccess** or create a custom policy:

The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, a sidebar navigation includes 'Identity and Access Management (IAM)', 'Dashboard', 'Access management' (with 'User groups' and 'Users' selected), 'Policies', 'Identity providers', 'Account settings', 'Access reports' (with 'Access Analyzer', 'External access', 'Unused access', 'Analyzer settings', 'Credential report', 'Organization activity', and 'Service control policies'), and 'CloudShell' and 'Feedback' buttons at the bottom.

The main content area displays the details for the user 'file-upload-user'. The 'Summary' section shows the ARN (arn:aws:iam::010438474149:user/file-upload-user), Console access status (Disabled), and an Access key (Access key 1, with a 'Create access key' link). It also shows the creation date (October 21, 2024, 11:58 (UTC+05:30)) and the last console sign-in (None).

The 'Permissions' tab is selected, showing one attached policy: 'AmazonS3FullAccess' (AWS managed, Directly). There are buttons for 'Remove' and 'Add permissions'.

At the bottom, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

- Replace **your-bucket-name** with your actual bucket name.
Click **Next**, review, and create the user or role.

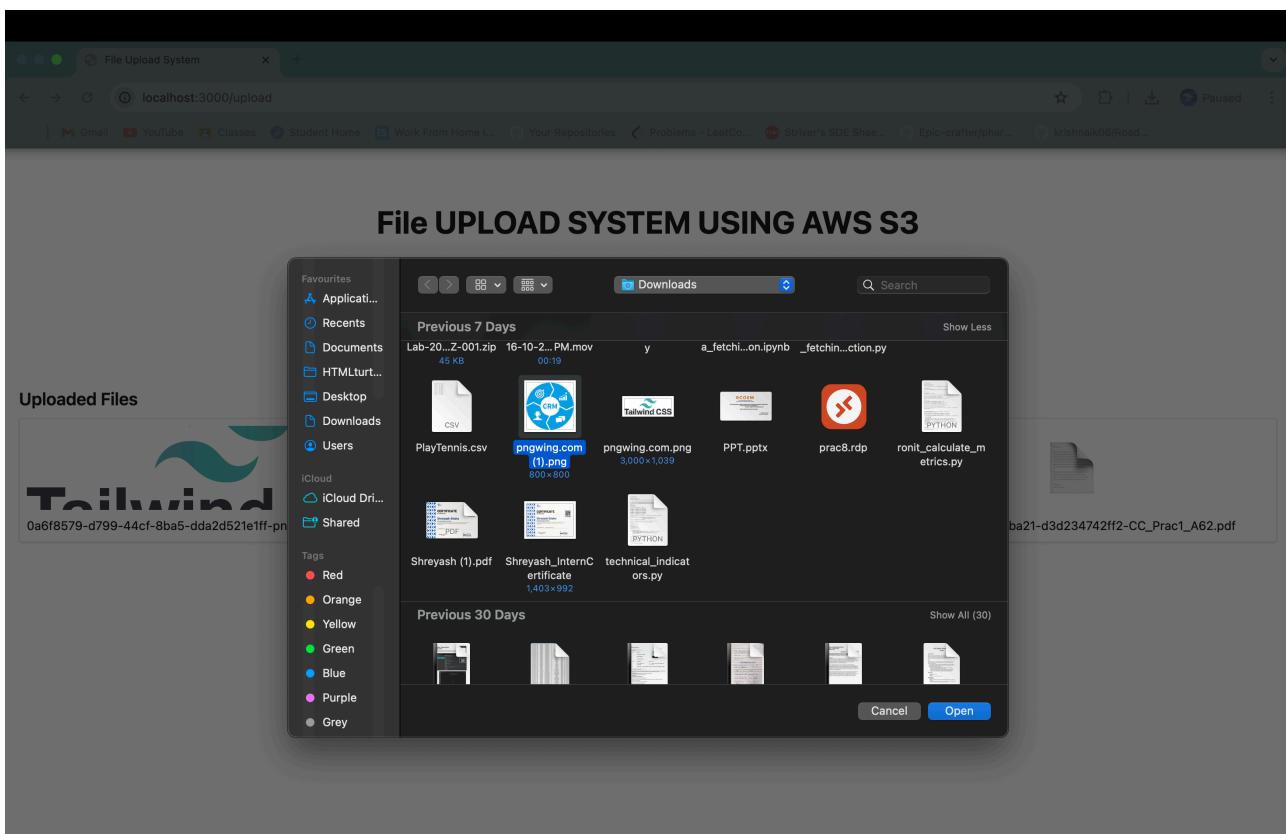
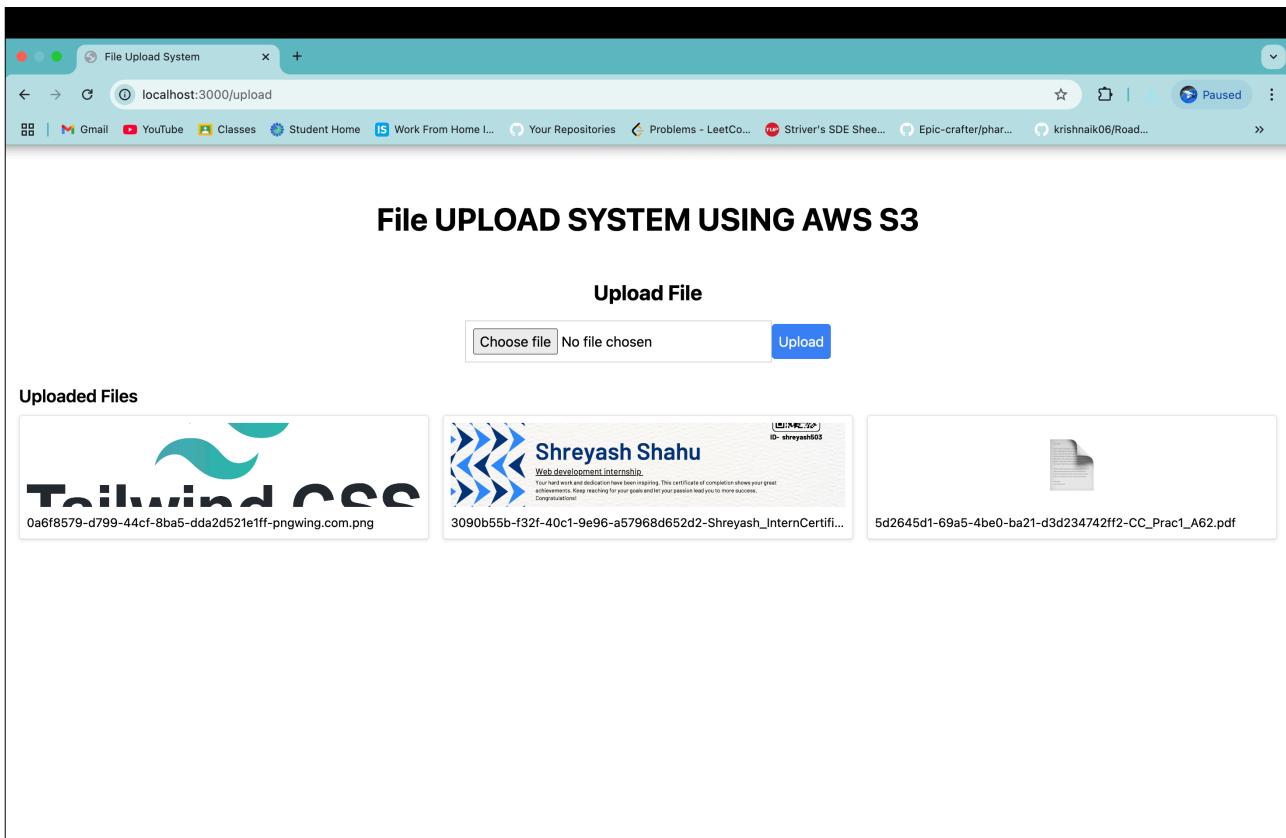
4. Save Access Key and Secret Key:

- For users: After creating the user, you'll receive an **Access Key ID** and **Secret Access Key**.
- Save these credentials as you'll need them in the `.env.local` file of your Next.js app.

The screenshot shows the AWS IAM 'Create access key' page. At the top, there's a green success message: 'Access key created' with a note: 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below this, the breadcrumb navigation shows: IAM > Users > file-upload-user > Create access key. The main section is titled 'Retrieve access keys' with a sub-section 'Access key'. It contains two tables: 'Access key' and 'Secret access key'. The 'Access key' table has one row with values: AKIAQE3ROPGSVMNSV27P and a 'Hide' link. The 'Secret access key' table has one row with values: O3q89Ui/E1GGkJovpkqtET9CM7jAla5mXWVLJVn and a 'Hide' link. Below these tables is a 'Access key best practices' section with a bulleted list: 'Never store your access key in plain text, in a code repository, or in code.', 'Disable or delete access key when no longer needed.', 'Enable least-privilege permissions.', and 'Rotate access keys regularly.' A note at the bottom of this section says: 'For more details about managing access keys, see the [best practices for managing AWS access keys](#)'. At the bottom right of the page are 'Download .csv file' and 'Done' buttons. The footer includes links for CloudShell, Feedback, Copyright notice (© 2024, Amazon Web Services, Inc. or its affiliates), Privacy, Terms, and Cookie preferences.

Application :

GitHub link : <https://github.com/1-Shreyash/ccta>



All Files getting stored in S3 :

Name	Type	Last modified	Size	Storage class
0a6f8579-d799-44cf-8ba5-dda2d521e1ff-pngwing.com.png	png	October 21, 2024, 13:36:59 (UTC+05:30)	188.1 KB	Standard
3090b55b-f32f-40c1-9e96-a57968d652d2-Shreyash_InternCertificate.jpg	jpg	October 21, 2024, 13:39:21 (UTC+05:30)	354.7 KB	Standard
5d2645d1-69a5-4be0-ba21-d3d234742ff2-CC_Prac1_A62.pdf	pdf	October 21, 2024, 12:44:36 (UTC+05:30)	2.0 MB	Standard
b65bc366-f1d4-409b-8a3a-42230c299425-pngwing.com (1).png	png	October 21, 2024, 13:46:51 (UTC+05:30)	73.6 KB	Standard