Cloud Computing Project

Team Members:

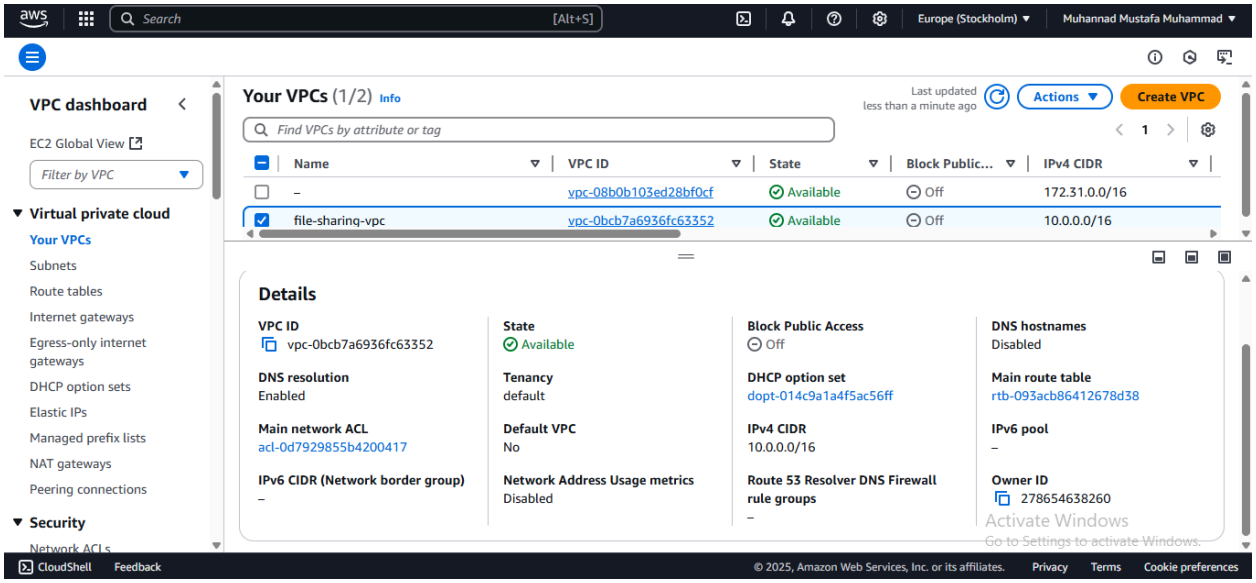
سيف الدين سامي مدكور	23012082
مهند محمد السعيد	23011569
مهند مصطفي محمد	23011571
خالد بدر المغاوري	23011258
محمد مغازي غنيم	23011495
عمر محمد عبد العال	23011386

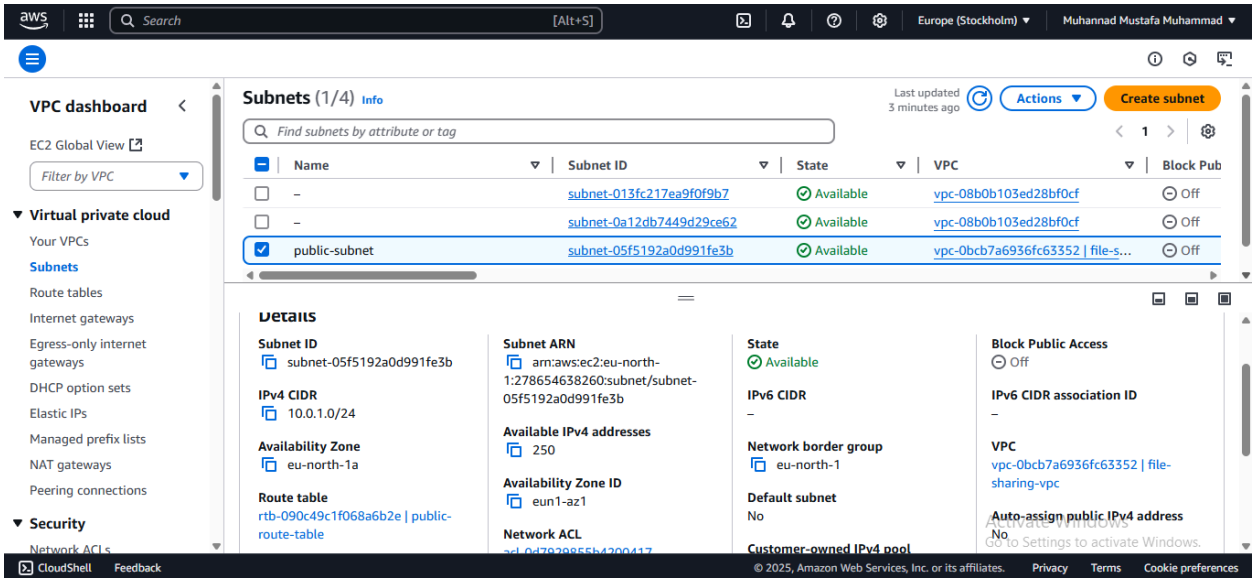
- **Creating AWS Account: مهند مصطفى محمد**

- **VPC and Testing:**

  مهند مصطفى محمد , خالد بدر المغاوري

  - **VPC Configuration**
    - VPC CIDR: `10.0.0.0/16`
    - Subnet CIDR: `10.0.1.0/24`
    - Internet Gateway attached.
    - Routing table modified to route all 0.0.0.0/0 traffic via IGW.





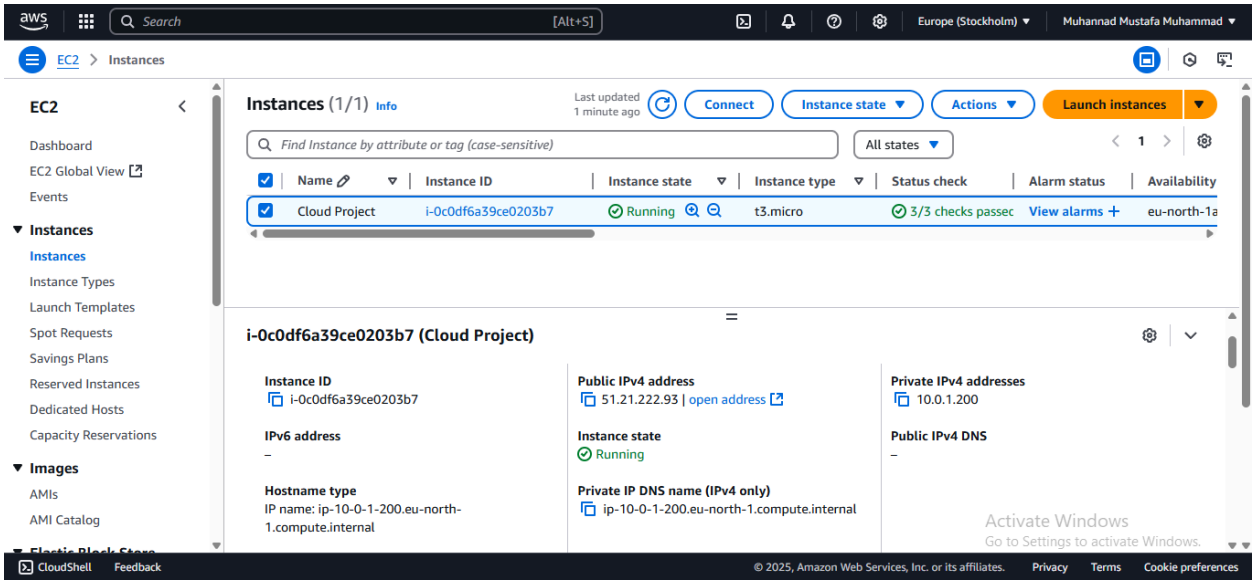
- Testing
  - The EC2 instance successfully connected to the S3 bucket using the IAM role.
  - Files could be uploaded from the EC2 terminal to S3 via CLI, proving connectivity and correct permissions.
  - Download links were generated using S3's pre-signed URL feature.

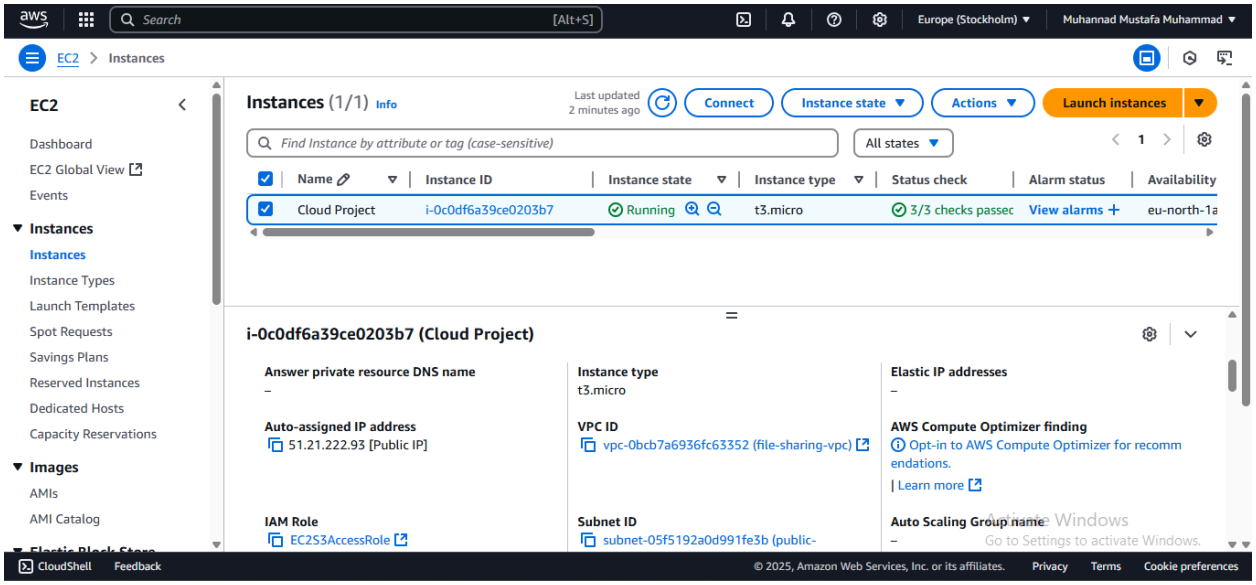
- **EC2 and IAM:**

محمد مغازي غنيم

  - **Amazon EC2**

- A virtual server (t3.micro – Free Tier eligible) was launched.
- A user data script was attached to initialize server configuration and (optionally) deploy the application automatically.
- The EC2 instance was assigned to our subnet for internet access that we created.
- Security Group was configured to open HTTP (port 80) and SSH (port 22) access.

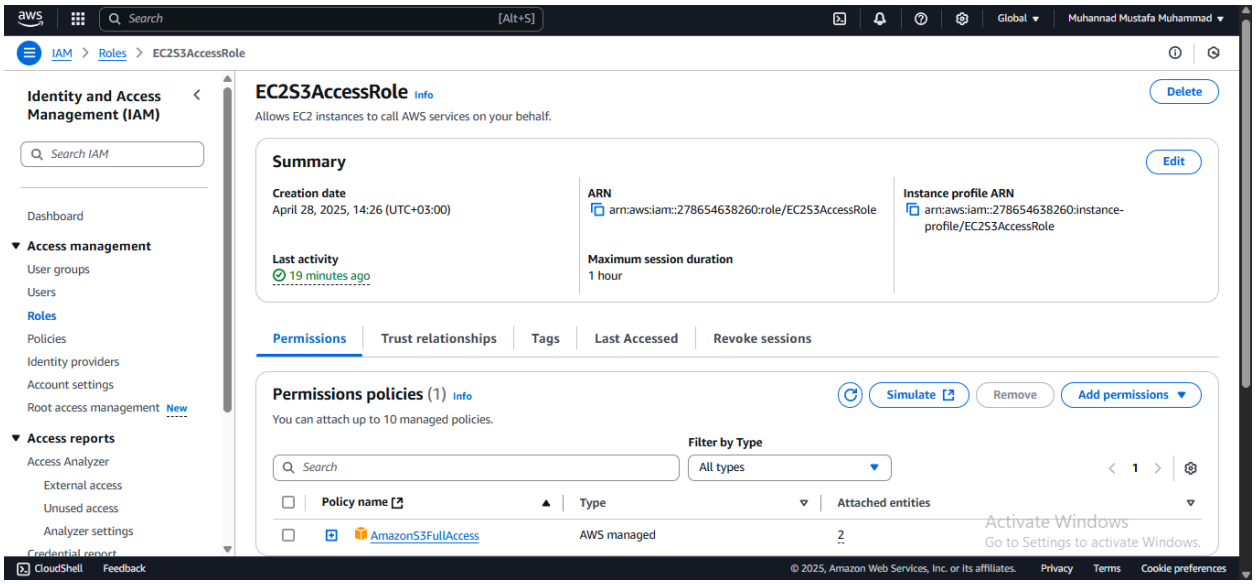


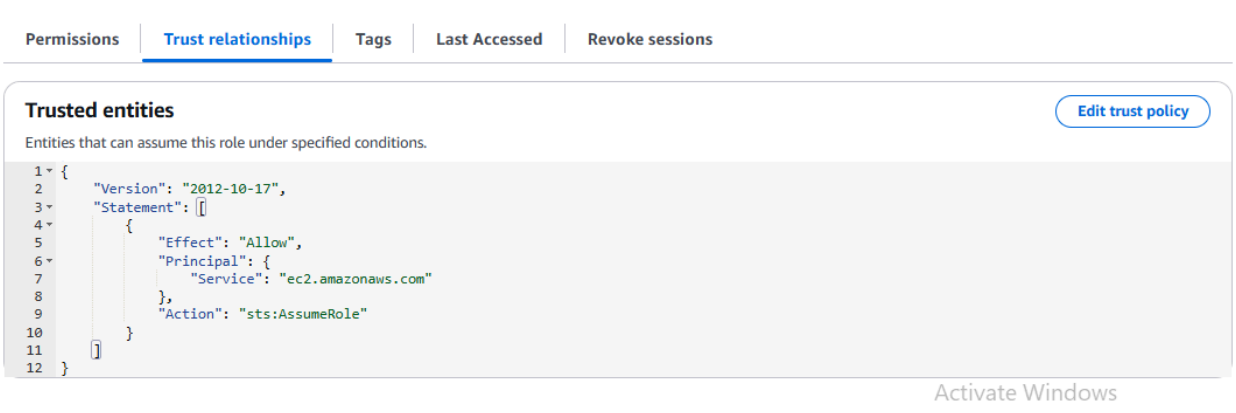


- **IAM Rule**
  - **An IAM role was created with a policy allowing access to the S3 bucket.**

- **This role was attached to the EC2 instance securely.**

- **Principle of Least Privilege was applied – the role only allowed necessary S3 actions.**

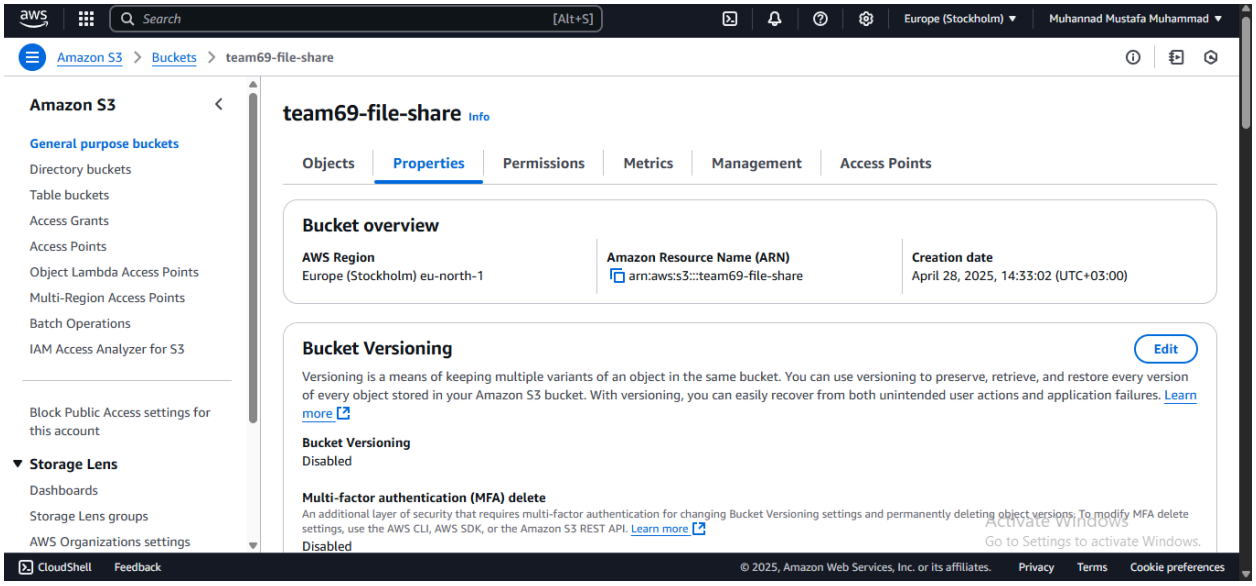


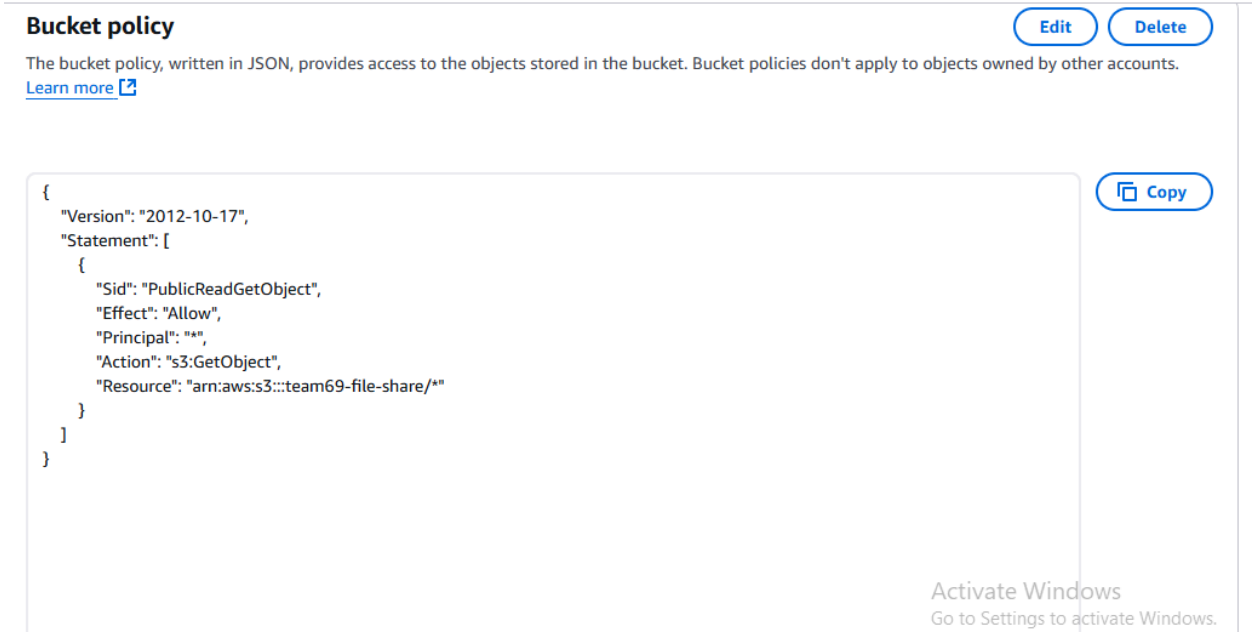


- **S3:**

مهند محمد السعيد , عمر محمد عبدالعال

- **S3 Bucket**
  - **Name: `team69-file-share`**
  - **Block Public Access: Enabled**
  - **Permissions: Set via IAM Role and Bucket Policy, Bucket policy and permissions were adjusted to allow uploads and retrieval by the application.**



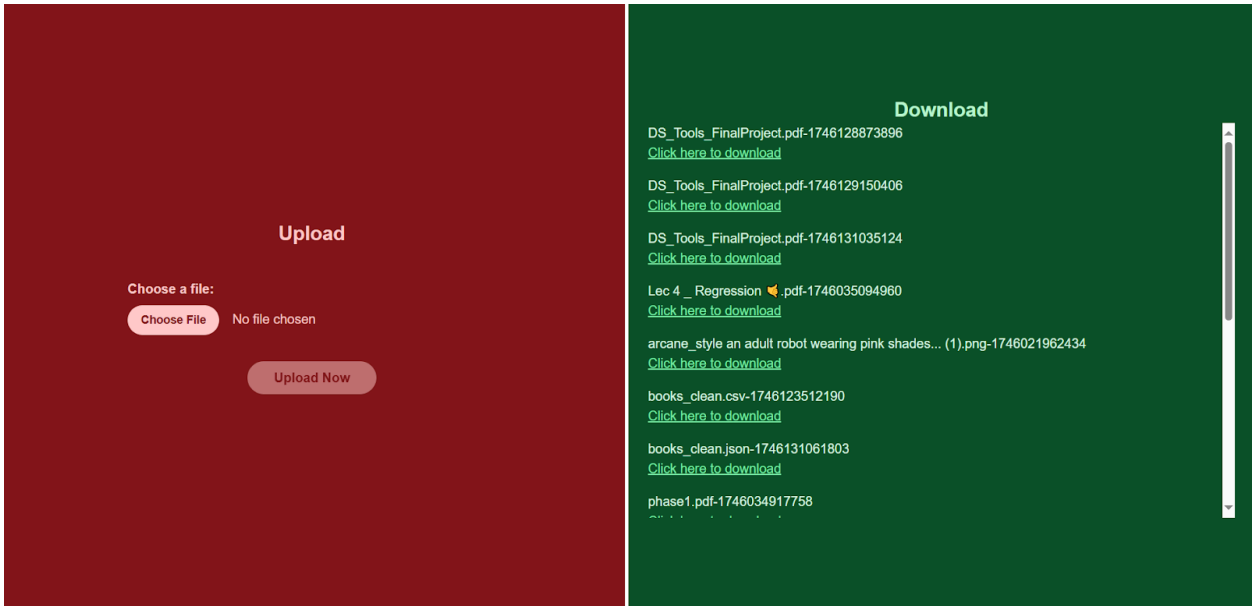


- **Web App:**

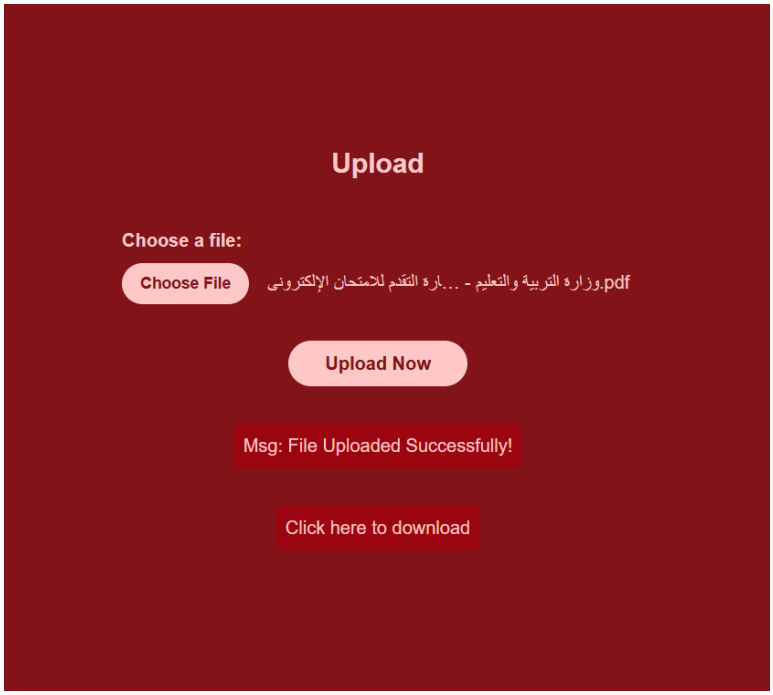
سيف الدين سامي مدكور

Using Next.JS created the web app, added UI styling using TailwindCSS, and used the API routing system in Next.JS.

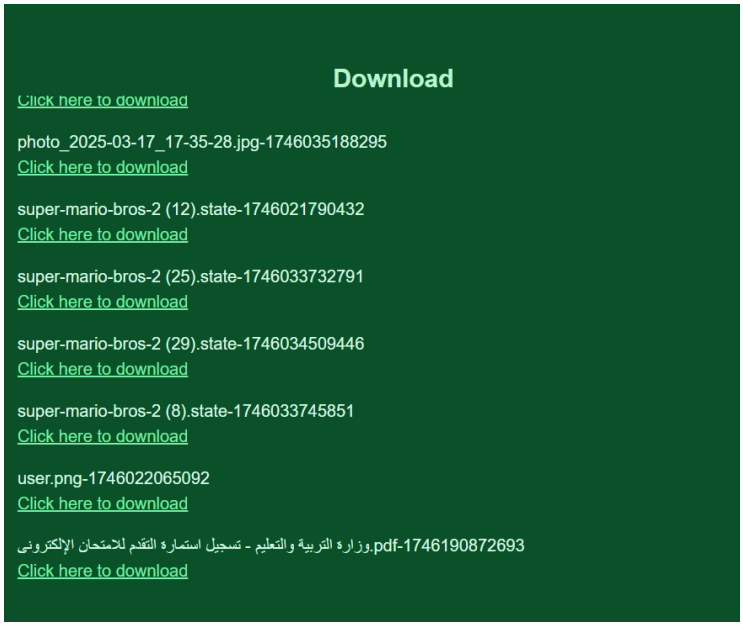
Split the screen using a flexbox to two sections Upload, and Download.



The moment you choose a file then upload it, you will get a download link immediately.



Also a list with all uploaded files will contain the new uploaded file.



A view of the API code:

```
export async function POST(req: NextRequest) {
    try {
        const formData = await req.formData();
        const file = formData.get('file');

        if (!file || !(file instanceof File)) {
            return NextResponse.json({ msg: 'Upload Error!!', err: 'Invalid or No File Found to
            Upload' }, { status: 400 });
        }

        const buffer = Buffer.from(await file.arrayBuffer());
        const fileUrl = await uploadFileToS3(buffer, file.name);

        return NextResponse.json({ msg: 'File Uploaded Successfully!!', fileUrl });
    } catch (e) {
        console.error('Error in POST /api/upload:', e);
        const errorMessage = e instanceof Error ? e.message : JSON.stringify(e);
        return NextResponse.json({ msg: 'Upload Error!!', err: errorMessage }, { status: 500 });
    }
}
```

```
export async function GET() {
    const bucketName = process.env.NEXT_PUBLIC_AWS_S3_BUCKET_NAME!

    try {
        const command = new ListObjectsV2Command({
            Bucket: bucketName,
        })

        const response = await s3.send(command)

        const files = response.Contents?.map((file) => ({
            name: file.Key!,
            url: `https://${bucketName}.s3.${process.env.NEXT_PUBLIC_AWS_REGION}.amazonaws.com/${file.Key}
            `,
        })) || []
      You, 2 days ago • project done for most …
        return NextResponse.json(files)
    } catch (error) {
        console.error('Error fetching files from S3:', error)
        return NextResponse.json({ error: 'Failed to fetch files' }, { status: 500 })
    }
}
```

All Secret access keys and important info was saved using environment variables as shown in the code to save them securely.