

**Name : Hamza Jamshaid**

**Project title :**

**Implement a Cloud-Based File Storage System**

**Introduction:**

Intern management systems often require secure storage and controlled access to sensitive documents such as joining letters, certificates, and reports. Traditional local storage solutions lack scalability, security, and remote accessibility. To address these challenges, this project implements a **Cloud-Based File Storage System** using **Microsoft Azure Blob Storage**, providing a secure, scalable, and efficient solution for document management.

**Objectives:**

The specific objectives include:

- Setting up Azure Blob Storage for cloud file management.
- Organizing documents into structured folders such as joining letters, certificates, and reports.
- Ensuring secure access to files using signed URLs (Shared Access Signatures).
- Providing controlled and temporary access to documents without exposing storage credentials.

**Project Structure:**

```
azure-file-storage-system/  
├── server.js  
├── .env  
├── controllers/  
│   └── fileController.js  
├── routes/  
│   └── fileRoutes.js  
├── package.json  
└── README.md
```

**Project Procedure:**

Below is a high-level breakdown of all procedures performed in the project. Each step mentions only actions and file names.

## 1. Create Storage account:

- Create a Storage account
- Create Container
- Get Storage Credentials

The screenshot displays the Microsoft Azure portal interface. The top navigation bar includes the 'Microsoft Azure' logo, an 'Upgrade' button, a search bar, and a user profile for 'hamzajamshaid339@g...'. The main content area shows the 'Overview' page for a deployment named 'internfilestorage\_1767848267683'. A green checkmark indicates 'Your deployment is complete'. Deployment details include the name, subscription ('Azure subscription 1'), resource group ('project2-rg'), start time ('1/8/2026, 10:00:24 AM'), and correlation ID. A 'Go to resource' button is visible. On the right, there are sections for 'Cost Management' and 'Microsoft Defender for Cloud'. Below the deployment overview, the 'Containers' view of the storage account is shown, listing two containers: 'logs' and 'intern-documents', both created on 1/8/2026.

Name	Last modified	Anonymous access level	Lease state
logs	1/8/2026, 10:01:05 AM	Private	Available
intern-documents	1/8/2026, 10:11:00 AM	Private	Available

## 2. Backend Project Setup:

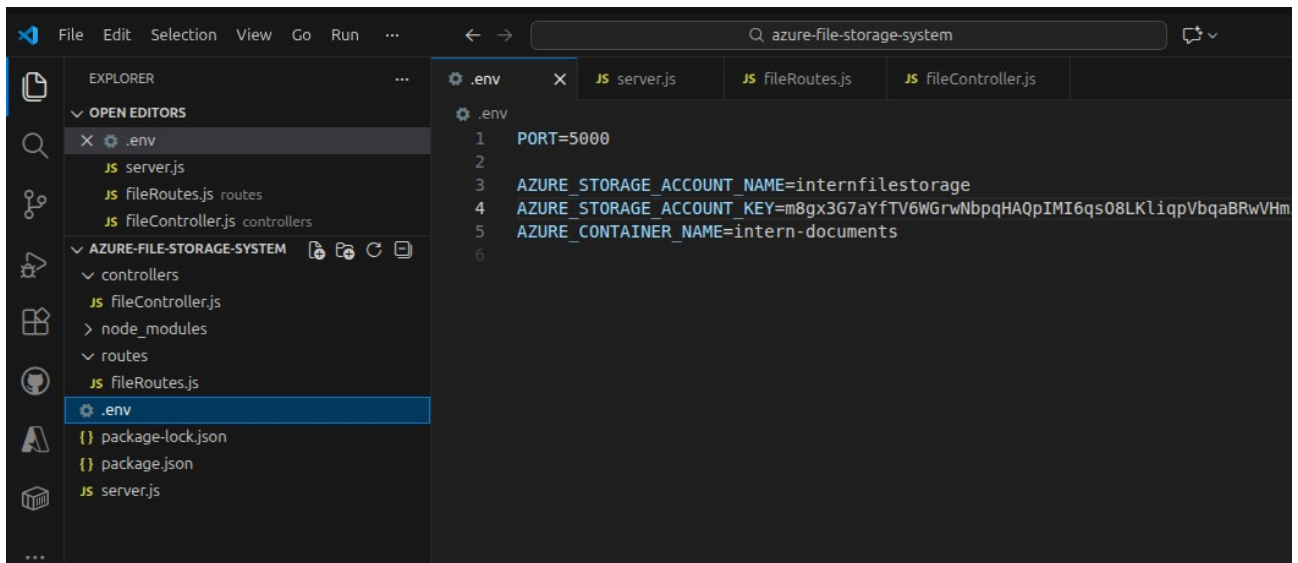
- Create Project folder
- Initialize Node.js Project

```
hamza@hamza-HP-EliteBook-840-G4:~$ mkdir azure-file-storage-system
hamza@hamza-HP-EliteBook-840-G4:~$ cd azure-file-storage-system
hamza@hamza-HP-EliteBook-840-G4:~/azure-file-storage-system$ npm init -y
Wrote to /home/hamza/azure-file-storage-system/package.json:

{
  "name": "azure-file-storage-system",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

hamza@hamza-HP-EliteBook-840-G4:~/azure-file-storage-system$
```

### 3. Environment Configuration:

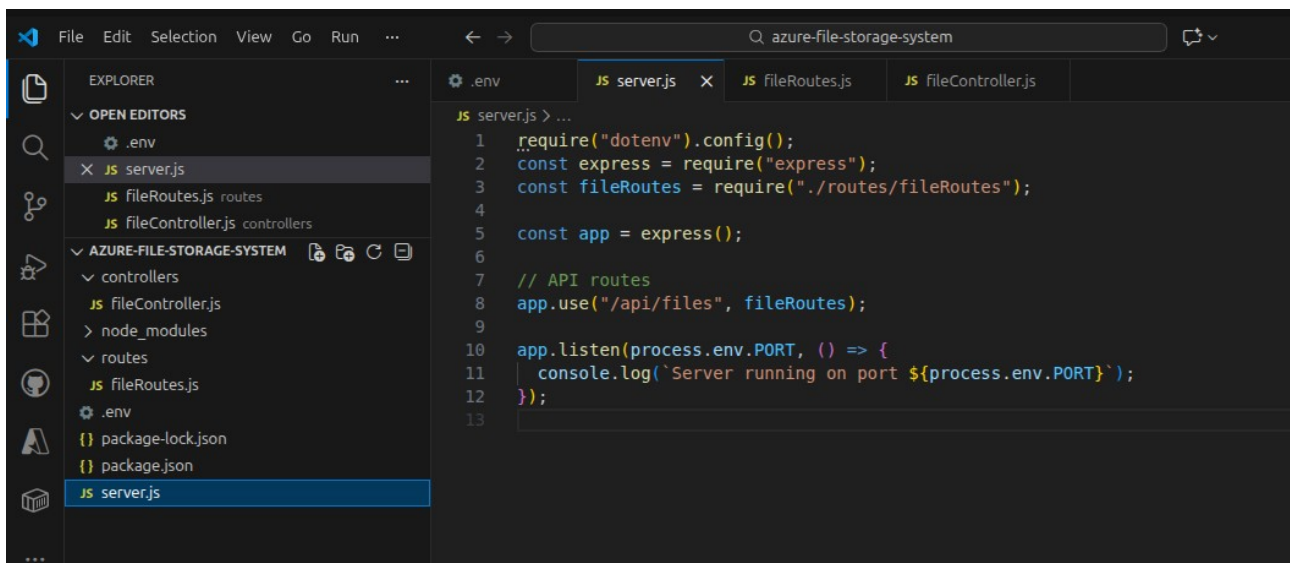


The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'AZURE-FILE-STORAGE-SYSTEM' folder is expanded, showing subfolders 'controllers' and 'routes', and files 'fileController.js', 'fileRoutes.js', and '.env'. The '.env' file is selected and its content is displayed in the main editor. The search bar at the top contains 'azure-file-storage-system'.

```
.env
1  PORT=5000
2
3  AZURE_STORAGE_ACCOUNT_NAME=internfilestorage
4  AZURE_STORAGE_ACCOUNT_KEY=m8gx3G7aYfTV6GrwNbpqHAQpIMI6qs08LKliqpVbqaBRwVHm
5  AZURE_CONTAINER_NAME=intern-documents
6
```

### 4.Backend Coding:

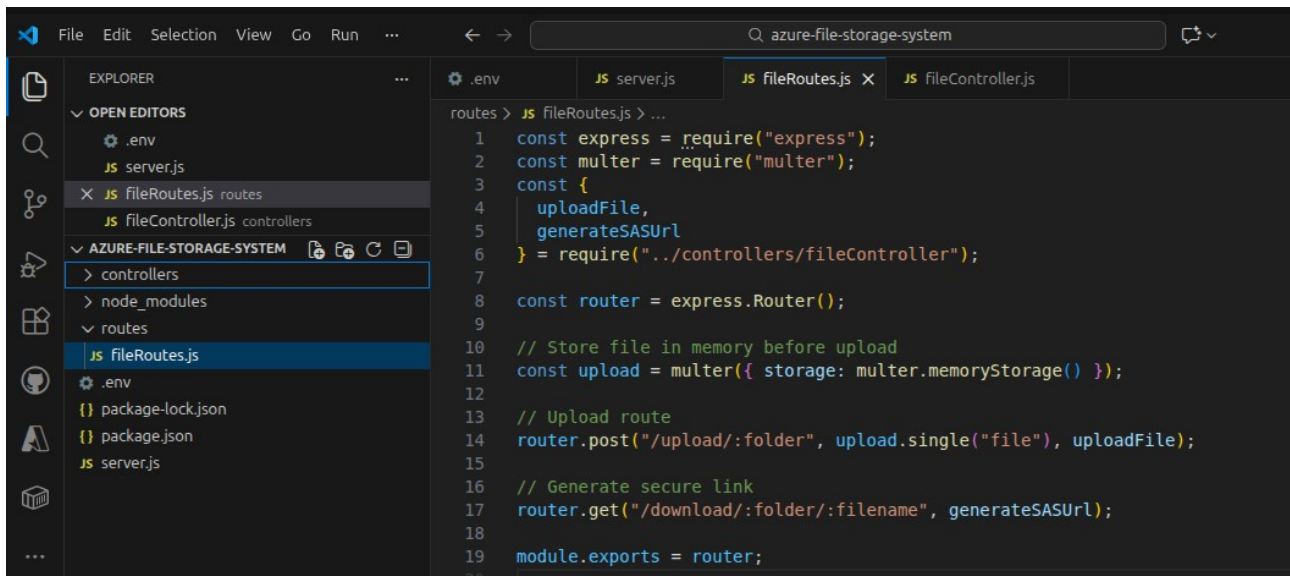
#### 4.1 Server.js:-



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'AZURE-FILE-STORAGE-SYSTEM' folder is expanded, showing subfolders 'controllers' and 'routes', and files 'fileController.js', 'fileRoutes.js', and 'server.js'. The 'server.js' file is selected and its content is displayed in the main editor. The search bar at the top contains 'azure-file-storage-system'.

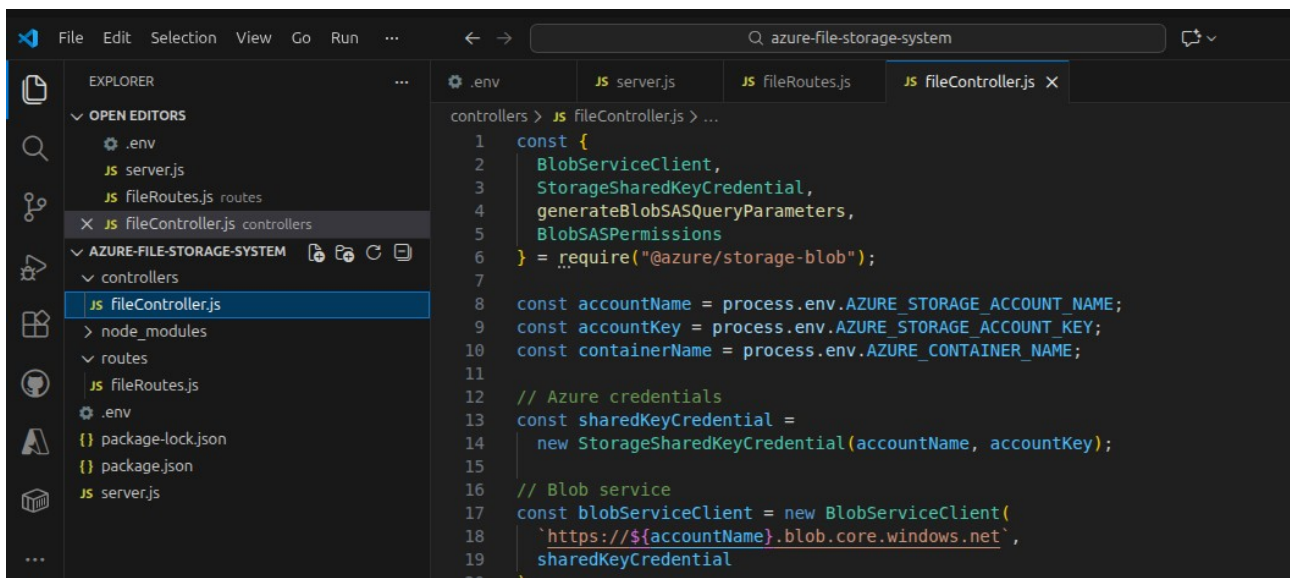
```
JS server.js > ...
1  require("dotenv").config();
2  const express = require("express");
3  const fileRoutes = require("./routes/fileRoutes");
4
5  const app = express();
6
7  // API routes
8  app.use("/api/files", fileRoutes);
9
10 app.listen(process.env.PORT, () => {
11   console.log(`Server running on port ${process.env.PORT}`);
12 });
13
```

## 4.2 routes/fileRoutes.js :-



```
1 const express = require("express");
2 const multer = require("multer");
3 const {
4   uploadFile,
5   generateSASUrl
6 } = require("../controllers/fileController");
7
8 const router = express.Router();
9
10 // Store file in memory before upload
11 const upload = multer({ storage: multer.memoryStorage() });
12
13 // Upload route
14 router.post("/upload/:folder", upload.single("file"), uploadFile);
15
16 // Generate secure link
17 router.get("/download/:folder/:filename", generateSASUrl);
18
19 module.exports = router;
```

## 4.3 controllers/fileController.js :-



```
1 const {
2   BlobServiceClient,
3   StorageSharedKeyCredential,
4   generateBlobSASQueryParameters,
5   BlobSASPermissions
6 } = require("@azure/storage-blob");
7
8 const accountName = process.env.AZURE_STORAGE_ACCOUNT_NAME;
9 const accountKey = process.env.AZURE_STORAGE_ACCOUNT_KEY;
10 const containerName = process.env.AZURE_CONTAINER_NAME;
11
12 // Azure credentials
13 const sharedKeyCredential =
14   new StorageSharedKeyCredential(accountName, accountKey);
15
16 // Blob service
17 const blobServiceClient = new BlobServiceClient(
18   `https://${accountName}.blob.core.windows.net`,
19   sharedKeyCredential
20 );
```

## 5: Run & Test :



```
1745 node server.js
```

## Result :-

The project was successfully implemented using Azure Blob Storage. Separate containers and folders were created for different types of intern documents, ensuring proper organization and easy retrieval. Files were uploaded securely to the cloud, and **signed URLs** were generated to allow temporary and permission-based access to specific documents. The system demonstrated reliable performance, secure data handling, and efficient file access from remote locations.

## **Conclusion :-**

The Cloud-Based File Storage System effectively fulfills its objective of secure and organized document storage using Azure services. By leveraging Azure Blob Storage and signed URLs, the system ensures data security, scalability, and controlled access. This solution reduces the risks associated with local storage, improves accessibility, and provides a practical foundation for enterprise-level intern or document management systems. The project highlights the importance of cloud technologies in modern application development and can be extended further with role-based access control and automation features