Centurion University	School: .... Campus: ....
	Academic Year: .... Subject Name: .... Subject Code: ....
	Semester: ...... Program: .... Branch: .... Specialization: ....
	Date: ....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : Store with IPFS – Decentralized File Upload

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

1. Open Brave Browser and visit the Pinata website.
2. Log in to your Pinata account.
3. Click on "Get Started" to access your dashboard.
4. Generate an API Key (optional if uploading manually).
5. Now Create a Folder in your system (eg. backend)
6. Now Open the folder With VS code and create .env and addfile.js file
7. Now Write the code which is redirect to pinata and upload files directly

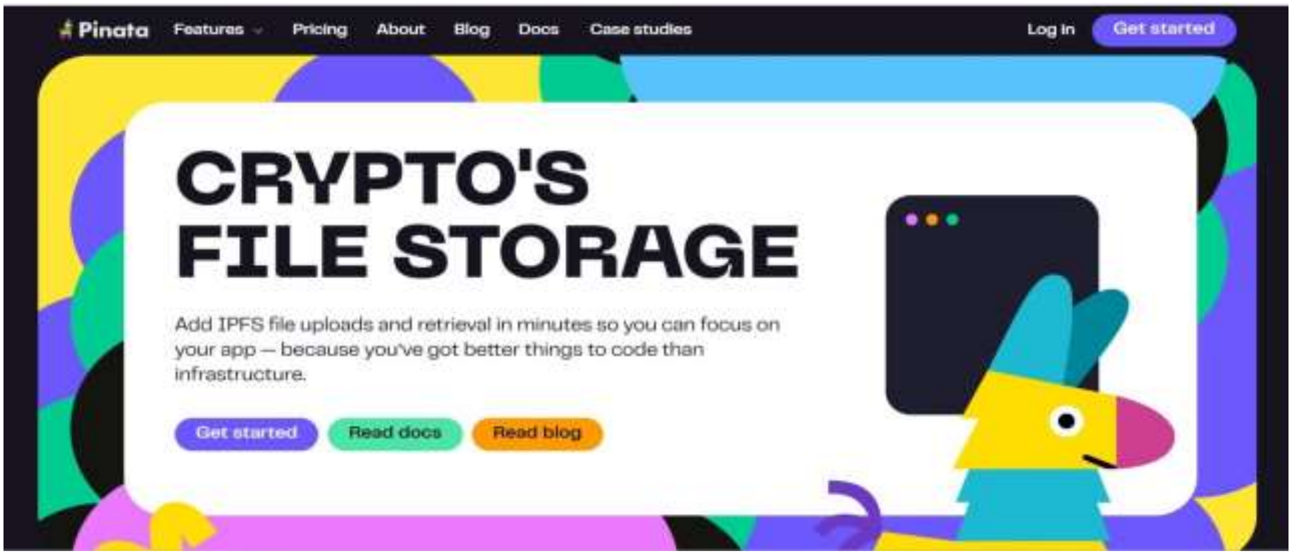
*Softwares required:

1.Brave Browser
2.Pinata
3.Vs Code

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

Open Brave Browser and go to https://www.pinata.cloud, which is a platform for uploading and pinning files on the IPFS (InterPlanetary File System).

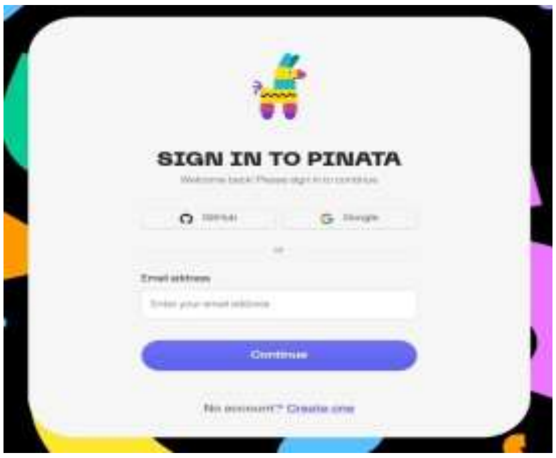

Pinata Features Pricing About Blog Docs Case studies Log in Get started

CRYPTO'S FILE STORAGE

Add IPFS file uploads and retrieval in minutes so you can focus on your app — because you've got better things to code than infrastructure.

Get started Read docs Read blog

Click on "Login" and sign in using your credentials. If you're new, sign up and verify your email.


SIGN IN TO PINATA

Welcome back! Please sign in to continue

GitHub Google

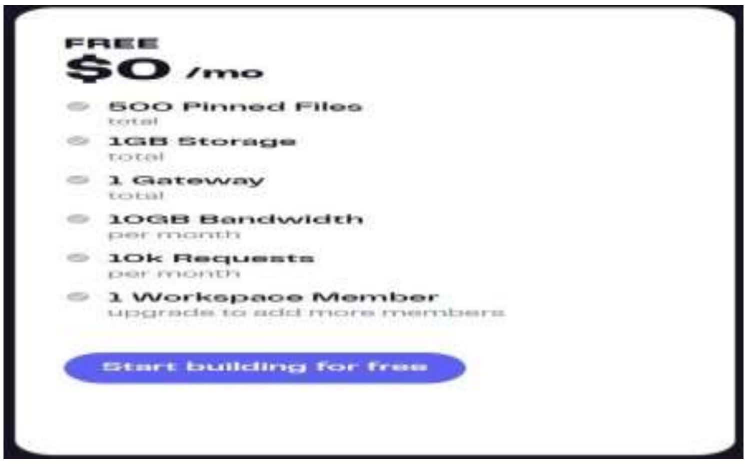
or

Email address

Enter your email address

Continue

No account? Create one

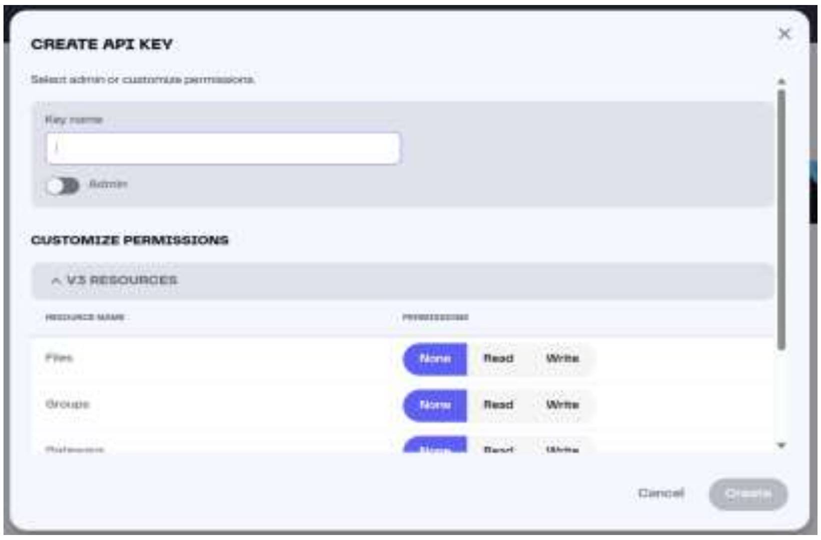

FREE

$0 /mo

- 500 Pinned Files total
- 1GB Storage total
- 1 Gateway total
- 10GB Bandwidth per month
- 10k Requests per month
- 1 Workspace Member upgrade to add more members

Start building for free

If needed for API uploads, go to the API Keys section and click on "New Key" to generate an API key..


CREATE API KEY

Select admin or customize permissions.

Key name

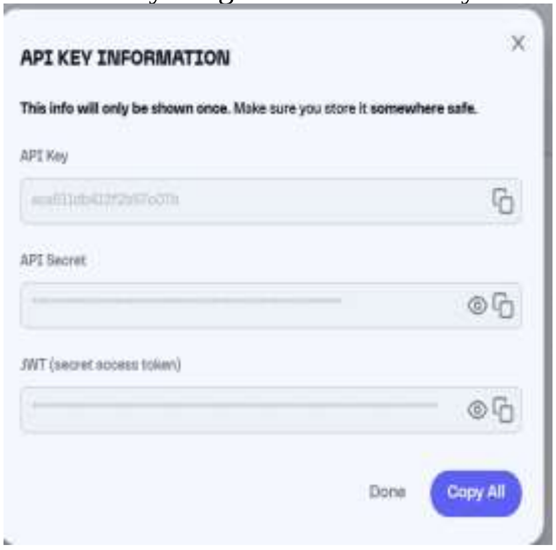
Admin

CUSTOMIZE PERMISSIONS

V3 RESOURCES

RESOURCE NAME	PERMISSIONS
Files	[✓ None] Read Write
Groups	[✓ None] Read Write
Gateways	[✓ None] Read Write

Cancel Create


API KEY INFORMATION

This info will only be shown once. Make sure you store it somewhere safe.

API Key

[illegible]

API Secret

JWT (secret access token)

Done Copy All

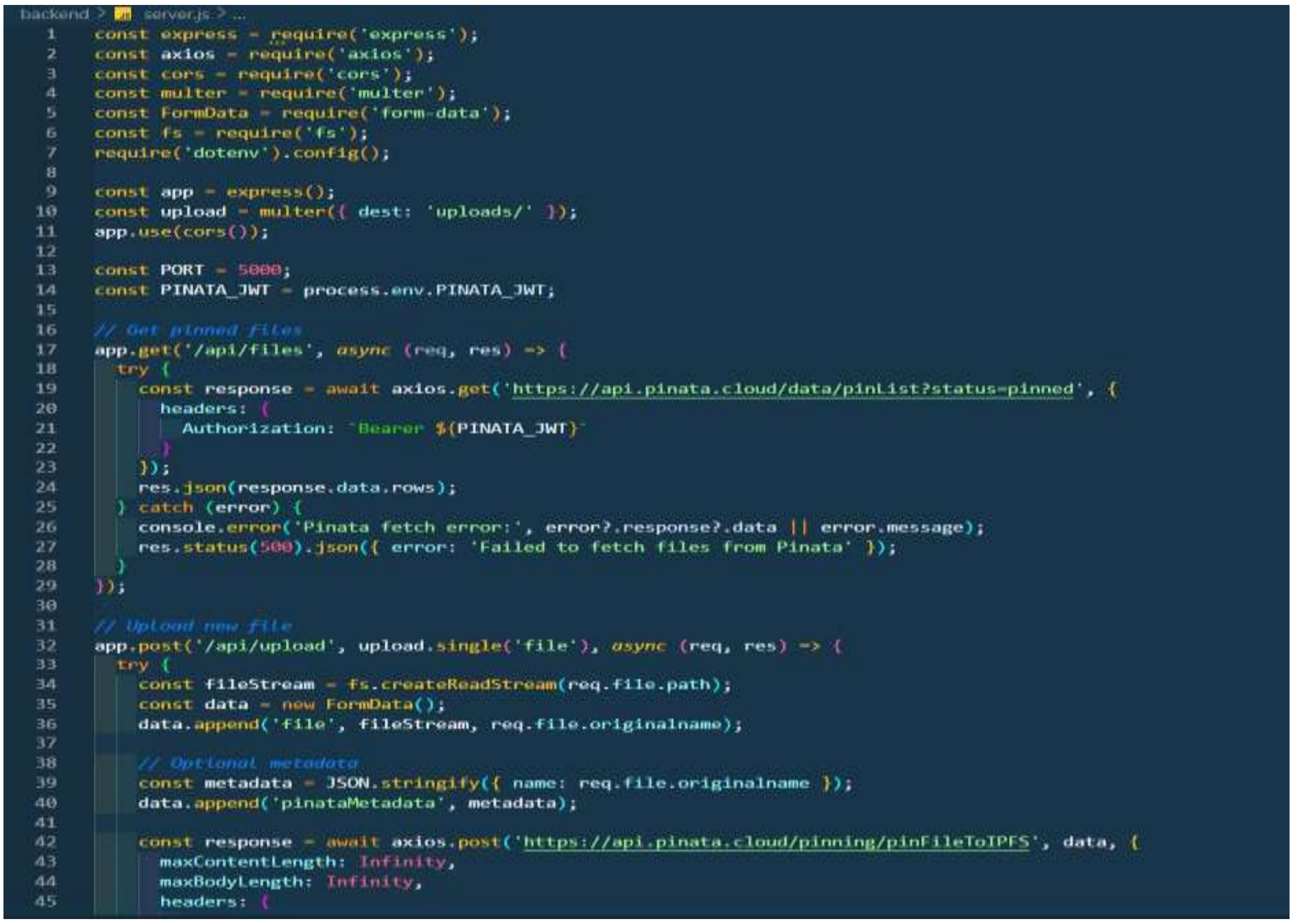
*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

Create a folder in your system and open with vs code and in this folder create a .env and a addfile.js file to write your code .

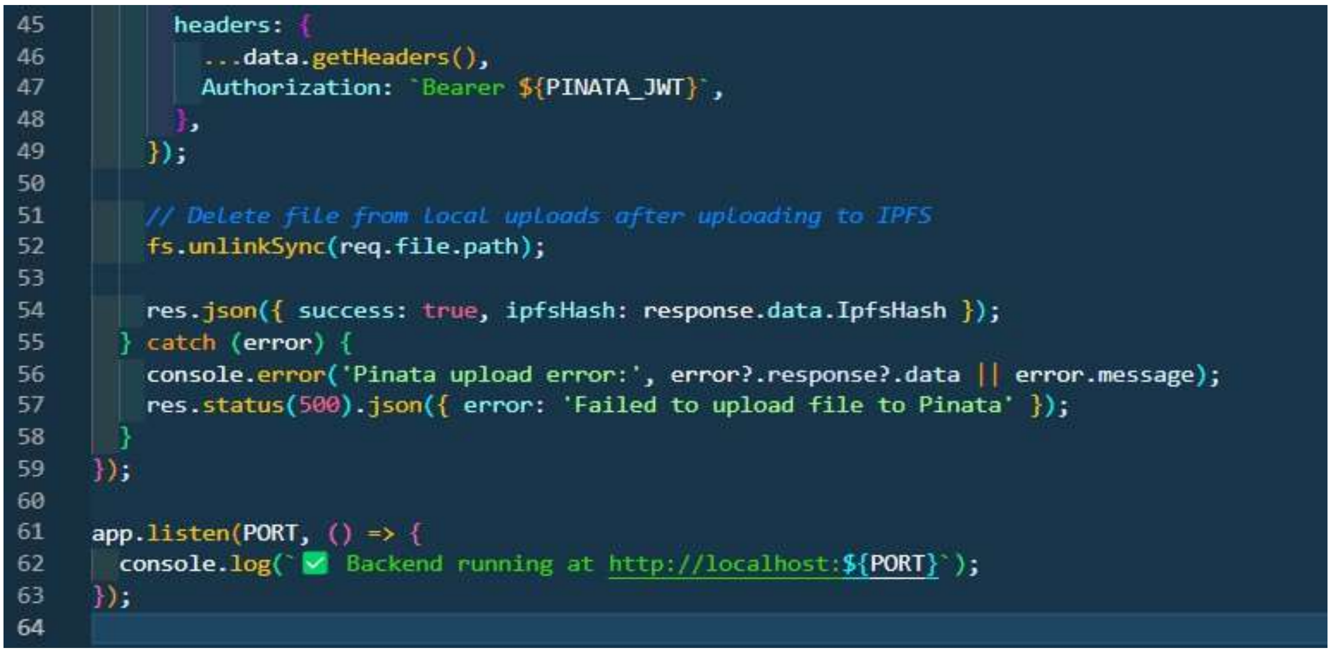
To add .json files in your folder in teminal write the command: npm init -y

Now write the code inside the server.js to upload files in pinata cloud and also create a app.jsx file for frontend



backend > server.js > ...

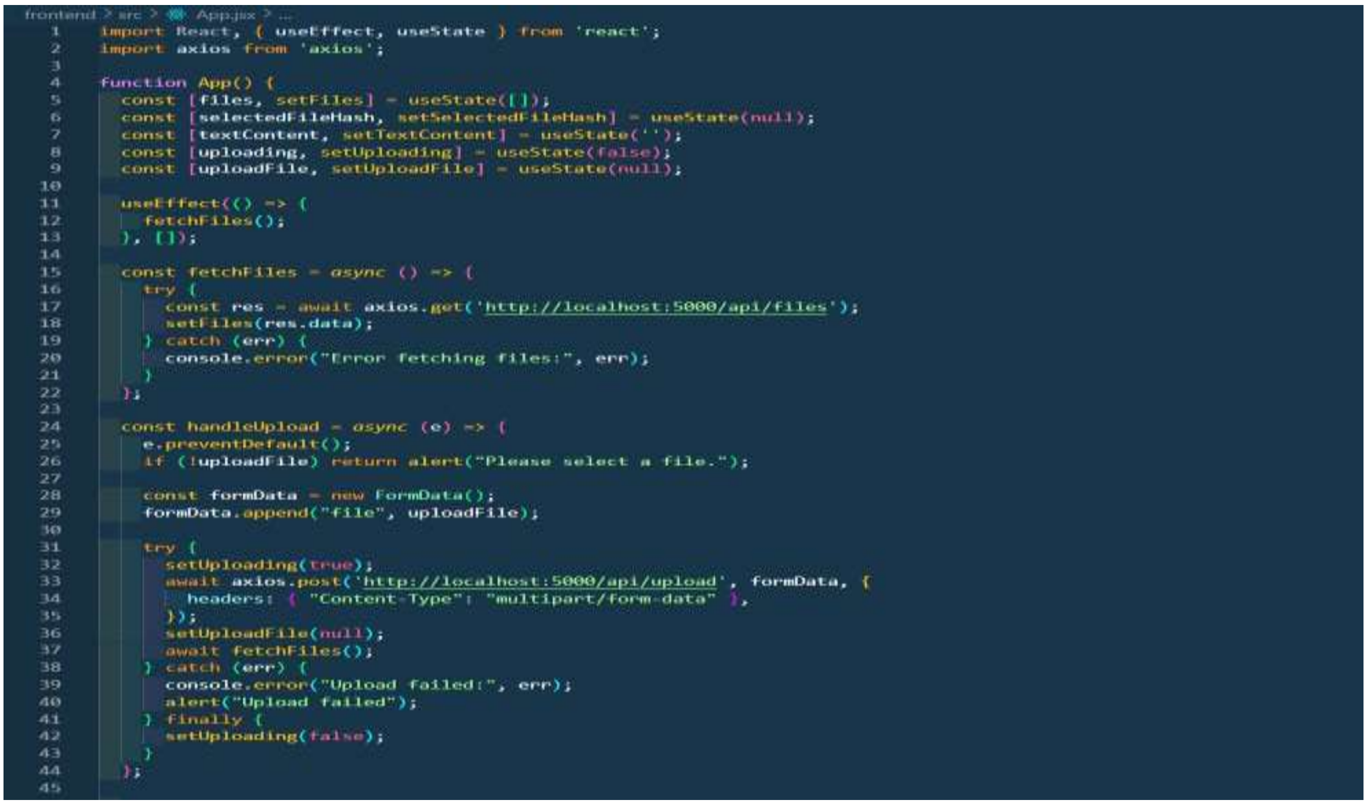
```javascript
const express = require('express');
const axios = require('axios');
const cors = require('cors');
const multer = require('multer');
const FormData = require('form-data');
const fs = require('fs');
require('dotenv').config();

const app = express();
const upload = multer({ dest: 'uploads/' });
app.use(cors());

const PORT = 5000;
const PINATA_JWT = process.env.PINATA_JWT;

// Get pinned files
app.get('/api/files', async (req, res) => {
  try {
    const response = await axios.get('https://api.pinata.cloud/data/pinList?status=pinned', {
      headers: {
        Authorization: `Bearer ${PINATA_JWT}`
      }
    });
    res.json(response.data.rows);
  } catch (error) {
    console.error('Pinata fetch error:', error?.response?.data || error.message);
    res.status(500).json({ error: 'Failed to fetch files from Pinata' });
  }
});

// Upload new file
app.post('/api/upload', upload.single('file'), async (req, res) => {
  try {
    const fileStream = fs.createReadStream(req.file.path);
    const data = new FormData();
    data.append('file', fileStream, req.file.originalname);

    // Optional metadata
    const metadata = JSON.stringify({ name: req.file.originalname });
    data.append('pinataMetadata', metadata);

    const response = await axios.post('https://api.pinata.cloud/pinning/pinFileToIPFS', data, {
      maxContentLength: Infinity,
      maxBodyLength: Infinity,
      headers: {
```



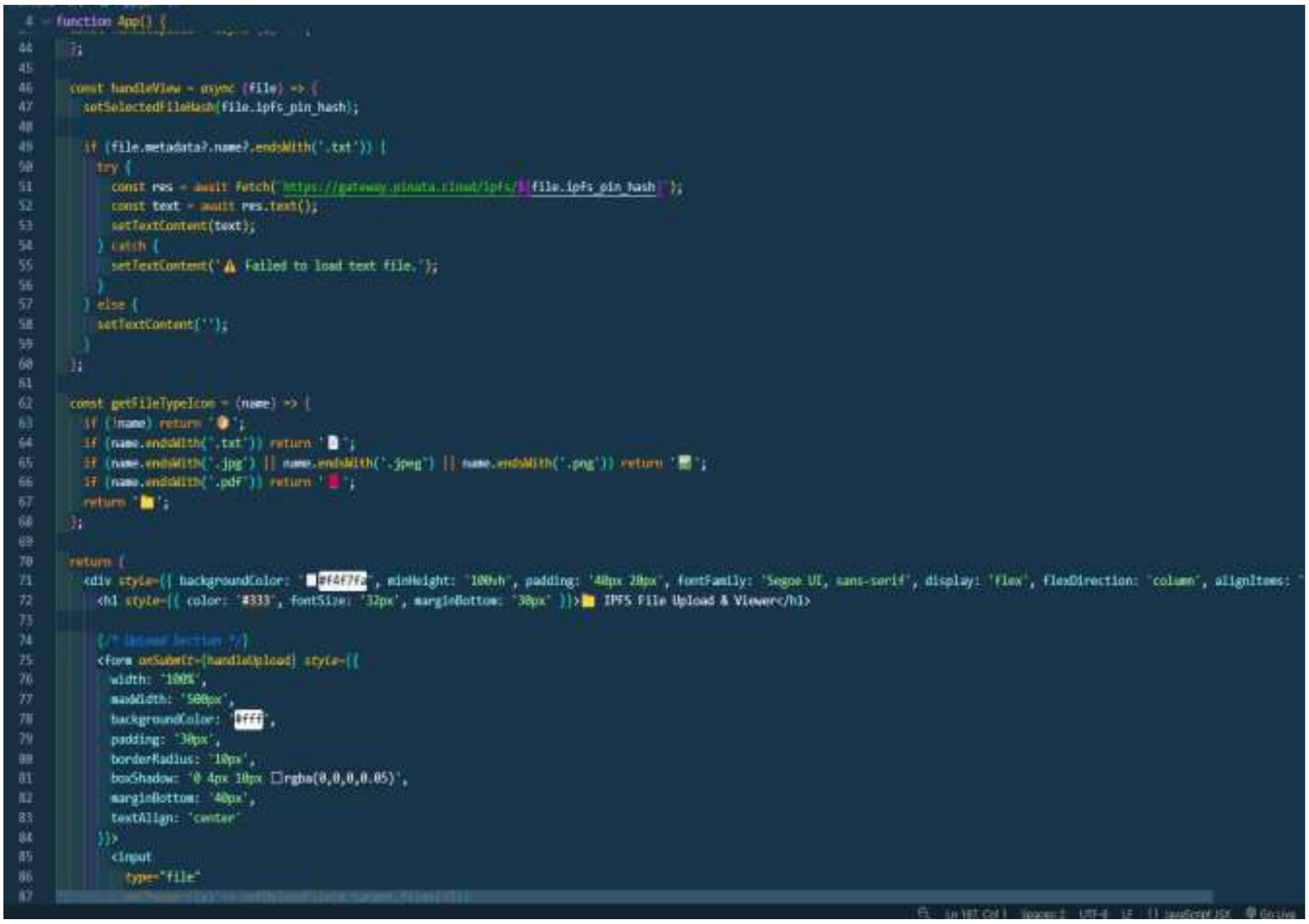
```javascript
      headers: {
        ...data.getHeaders(),
        Authorization: `Bearer ${PINATA_JWT}`,
      },
    });

    // Delete file from local uploads after uploading to IPFS
    fs.unlinkSync(req.file.path);

    res.json({ success: true, ipfsHash: response.data.IpfsHash });
  } catch (error) {
    console.error('Pinata upload error:', error?.response?.data || error.message);
    res.status(500).json({ error: 'Failed to upload file to Pinata' });
  }
});

app.listen(PORT, () => {
  console.log(`✅ Backend running at http://localhost:${PORT}`);
});
```

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)



```
frontend > src > App.jsx > ...
import React, { useEffect, useState } from 'react';
import axios from 'axios';

function App() {
  const [files, setFiles] = useState([]);
  const [selectedFileHash, setSelectedFileHash] = useState(null);
  const [textContent, setTextContent] = useState('');
  const [uploading, setUploading] = useState(false);
  const [uploadFile, setUploadFile] = useState(null);

  useEffect(() => {
    fetchFiles();
  }, []);

  const fetchFiles = async () => {
    try {
      const res = await axios.get('http://localhost:5000/api/files');
      setFiles(res.data);
    } catch (err) {
      console.error("Error fetching files:", err);
    }
  };

  const handleUpload = async (e) => {
    e.preventDefault();
    if (!uploadFile) return alert("Please select a file.");

    const formData = new FormData();
    formData.append("file", uploadFile);

    try {
      setUploading(true);
      await axios.post('http://localhost:5000/api/upload', formData, {
        headers: { "Content-Type": "multipart/form-data" },
      });
      setUploadFile(null);
      await fetchFiles();
    } catch (err) {
      console.error("Upload failed:", err);
      alert("Upload failed");
    } finally {
      setUploading(false);
    }
  };

```



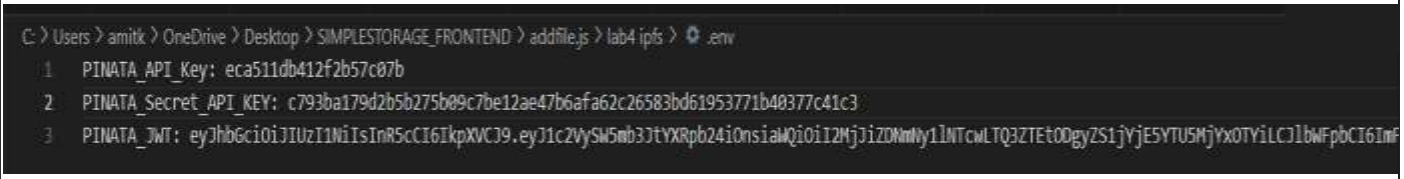
```
function App() {
  };

  const handleView = async (file) => {
    setSelectedFileHash(file.ipfs_pin_hash);

    if (file.metadata?.name?.endsWith('.txt')) {
      try {
        const res = await fetch(`https://gateway.pinata.cloud/ipfs/${file.ipfs_pin_hash}`);
        const text = await res.text();
        setTextContent(text);
      } catch {
        setTextContent('⚠ Failed to load text file.');
      }
    } else {
      setTextContent('');
    }
  };

  const getFileTypeIcon = (name) => {
    if (!name) return '❓';
    if (name.endsWith('.txt')) return '📄';
    if (name.endsWith('.jpg') || name.endsWith('.jpeg') || name.endsWith('.png')) return '🖼';
    if (name.endsWith('.pdf')) return '📕';
    return '📁';
  };

  return (
    <div style={{ backgroundColor: '#f4f7fa', minHeight: '100vh', padding: '40px 20px', fontFamily: 'Segoe UI, sans-serif', display: 'flex', flexDirection: 'column', alignItems: '
      <h1 style={{ color: '#333', fontSize: '32px', marginBottom: '30px' }}>📁 IPFS File Upload & Viewer</h1>

      {/* Upload Section */}
      <form onSubmit={handleUpload} style={{
        width: '100%',
        maxWidth: '500px',
        backgroundColor: '#fff',
        padding: '30px',
        borderRadius: '10px',
        boxShadow: '0 4px 10px rgba(0,0,0,0.05)',
        marginBottom: '40px',
        textAlign: 'center'
      }}>
        <input
          type="file"
          onChange={(e) => setUploadFile(e.target.files[0])}
```

Ln 187, Col 1   Spaces: 2   UTF-8   LF   {} JavaScript JSX   Go Live

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

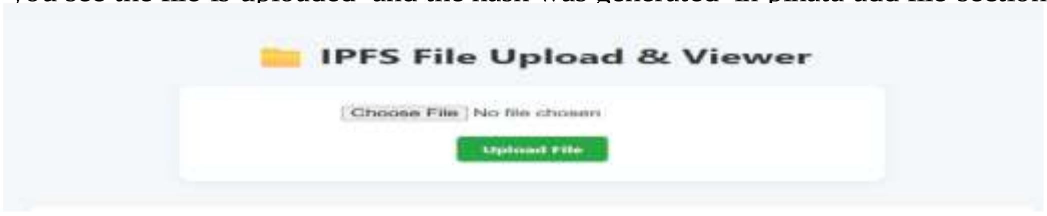
* Implementation Phase: Final Output (no error)

Now in .env file write your api key and secret api key



```
C: > Users > amitk > OneDrive > Desktop > SIMPLESTORAGE_FRONTEND > addfile.js > lab4 ipfs > .env
PINATA_API_Key: eca511db412f2b57c07b
PINATA_Secret_API_KEY: c793ba179d2b5b275b09c7be12ae47b6afa62c26583bd61953771b40377c41c3
PINATA_JWT: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySW5mb3JtYXRpb24iOnsiaWQiOiI2MjJiZDNmNy1lNTcwLTQ3ZTEtODgyZS1jYjE5YTU5MjYxOTYiLCJlbWFpbCI6ImF
```

Now you see the file is uploaded and the hash was generated in pinata add file section we can alse see with



Observation:

1. Uploading files to Pinata through the backend allows secure and automated integration without exposing API keys to the frontend.
2. Each successful upload returns a unique IPFS hash, which can be used to access the file from any IPFS gateway globally.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

******As applicable according to the experiment. Two sheets per experiment (10-20) to be used.*