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
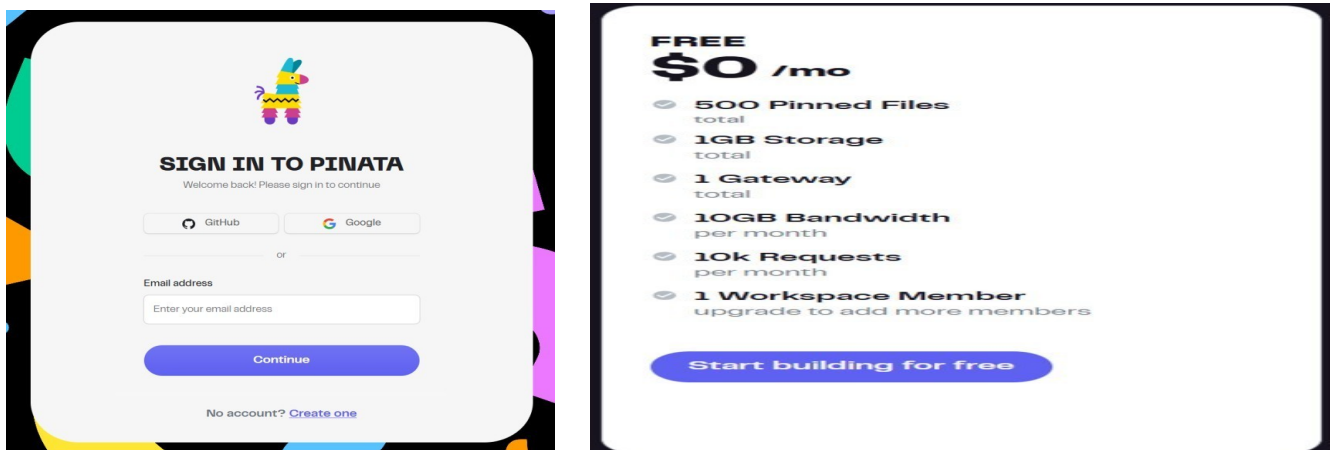
*Softwares required:

1. Brave Browser
2. Pinata
3. Vs Code

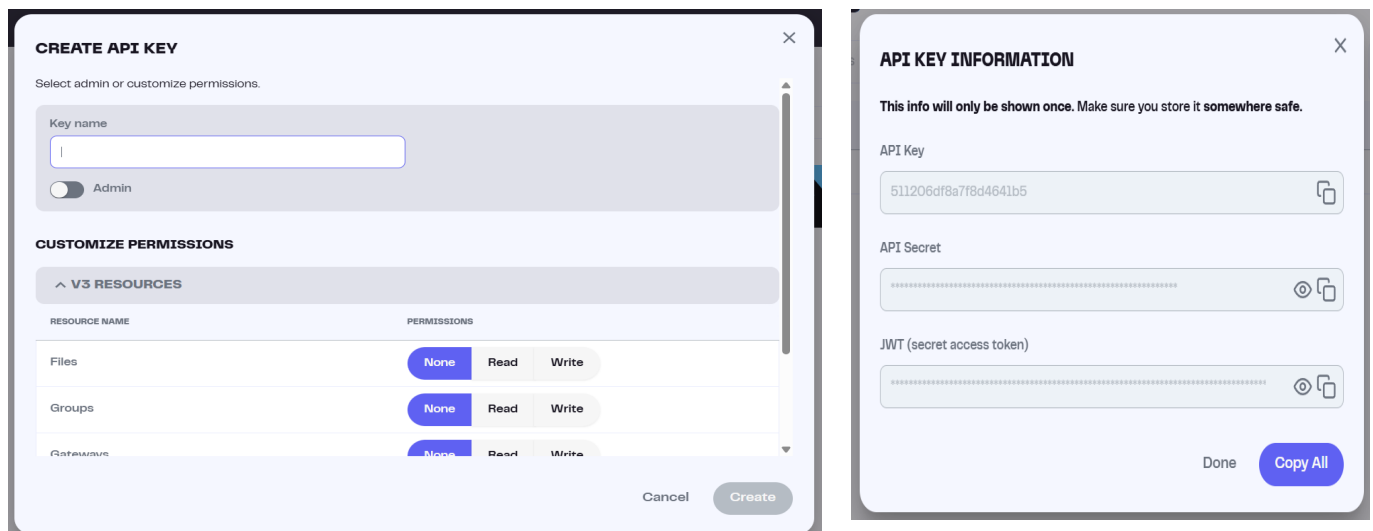## * **Implementation Phase: Final Output (no error)**

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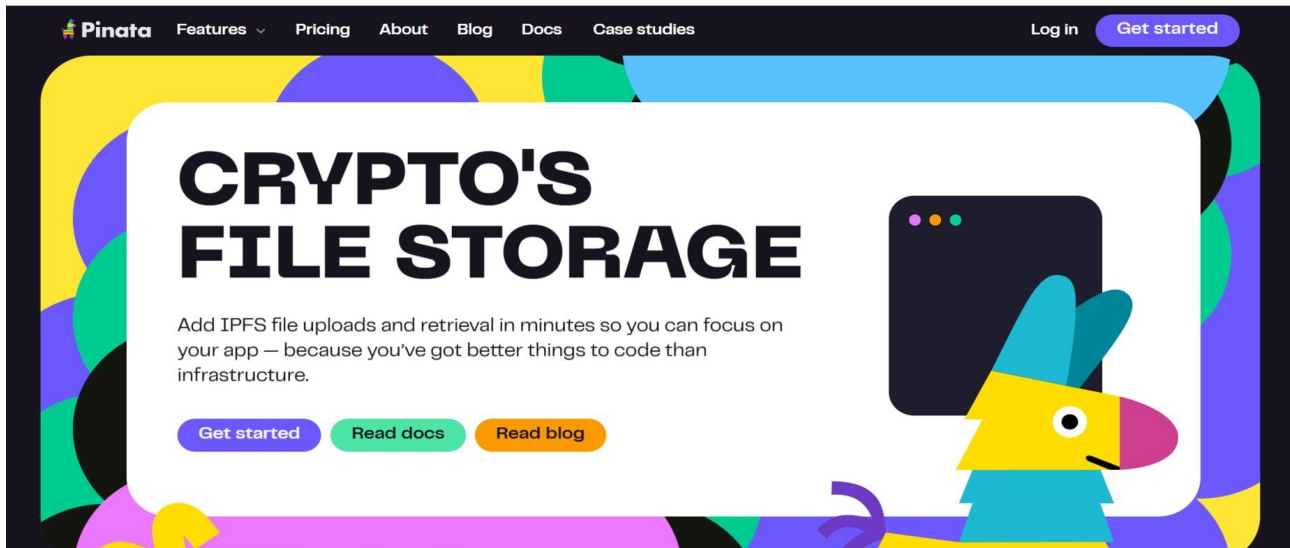
Click on "Login" and sign in using your credentials. If you're new, sign up and verify your email.

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

* 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

```js
backend > Js server.js > ...
1   const express = require('express');
2   const axios = require('axios');
3   const cors = require('cors');
4   const multer = require('multer');
5   const FormData = require('form-data');
6   const fs = require('fs');
7   require('dotenv').config();
8
9   const app = express();
10  const upload = multer({ dest: 'uploads/' });
11  app.use(cors());
12
13  const PORT = 5000;
14  const PINATA_JWT = process.env.PINATA_JWT;
15
16  // Get pinned files
17  app.get('/api/files', async (req, res) => {
18      try {
19        const response = await axios.get('https://api.pinata.cloud/data/pinList?status=pinned', {
20          headers: {
21            Authorization: `Bearer ${PINATA_JWT}`
22          }
23        });
24        res.json(response.data.rows);
25      } catch (error) {
26        console.error('Pinata fetch error:', error?.response?.data || error.message);
27        res.status(500).json({ error: 'Failed to fetch files from Pinata' });
28      }
29  });
30
31  // Upload new file
32  app.post('/api/upload', upload.single('file'), async (req, res) => {
33      try {
34        const fileStream = fs.createReadStream(req.file.path);
35        const data = new FormData();
36        data.append('file', fileStream, req.file.originalname);
37
38        // Optional metadata
39        const metadata = JSON.stringify({ name: req.file.originalname });
40        data.append('pinataMetadata', metadata);
41
42        const response = await axios.post('https://api.pinata.cloud/pinning/pinFileToIPFS', data, {
43          maxContentLength: Infinity,
44          maxBodyLength: Infinity,
45          headers: {
```

```js
45          headers: {
46            ...data.getHeaders(),
47            Authorization: `Bearer ${PINATA_JWT}`,
48          },
49        });
50
51        // Delete file from local uploads after uploading to IPFS
52        fs.unlinkSync(req.file.path);
53
54        res.json({ success: true, ipfsHash: response.data.IpfsHash });
55      } catch (error) {
56        console.error('Pinata upload error:', error?.response?.data || error.message);
57        res.status(500).json({ error: 'Failed to upload file to Pinata' });
58      }
59  });
60
61  app.listen(PORT, () => {
62    console.log(`✅ Backend running at http://localhost:${PORT}`);
63  });
64
```

* Implementation Phase: Final Output (no error)

```jsx
frontend > src > ⚙ App.jsx > ...
 1  import React, { useEffect, useState } from 'react';
 2  import axios from 'axios';
 3
 4  function App() {
 5    const [files, setFiles] = useState([]);
 6    const [selectedFileHash, setSelectedFileHash] = useState(null);
 7    const [textContent, setTextContent] = useState('');
 8    const [uploading, setUploading] = useState(false);
 9    const [uploadFile, setUploadFile] = useState(null);
10
11    useEffect(() => {
12      fetchFiles();
13    }, []);
14
15    const fetchFiles = async () => {
16      try {
17        const res = await axios.get('http://localhost:5000/api/files');
18        setFiles(res.data);
19      } catch (err) {
20        console.error("Error fetching files:", err);
21      }
22    };
23
24    const handleUpload = async (e) => {
25      e.preventDefault();
26      if (!uploadFile) return alert("Please select a file.");
27
28      const formData = new FormData();
29      formData.append("file", uploadFile);
30
31      try {
32        setUploading(true);
33        await axios.post('http://localhost:5000/api/upload', formData, {
34          headers: { "Content-Type": "multipart/form-data" },
35        });
36        setUploadFile(null);
37        await fetchFiles();
38      } catch (err) {
39        console.error("Upload failed:", err);
40        alert("Upload failed");
41      } finally {
42        setUploading(false);
43      }
44    };
45
```

```jsx
 4 ∨ function App() {
44    };
45
46    const handleView = async (file) => {
47      setSelectedFileHash(file.ipfs_pin_hash);
48
49      if (file.metadata?.name?.endsWith('.txt')) {
50        try {
51          const res = await fetch(`https://gateway.pinata.cloud/ipfs/${file.ipfs_pin_hash}`);
52          const text = await res.text();
53          setTextContent(text);
54        } catch {
55          setTextContent('⚠ Failed to load text file.');
56        }
57      } else {
58        setTextContent('');
59      }
60    };
61
62    const getFileTypeIcon = (name) => {
63      if (!name) return '📦';
64      if (name.endsWith('.txt')) return '📄';
65      if (name.endsWith('.jpg') || name.endsWith('.jpeg') || name.endsWith('.png')) return '🖼';
66      if (name.endsWith('.pdf')) return '📕';
67      return '📁';
68    };
69
70    return (
71      <div style={{ backgroundColor: '#f4f7fa', minHeight: '100vh', padding: '40px 20px', fontFamily: 'Segoe UI, sans-serif', display: 'flex', flexDirection: 'column', alignItems: '
72        <h1 style={{ color: '#333', fontSize: '32px', marginBottom: '30px' }}>📤 IPFS File Upload & Viewer</h1>
73
74        {/* Upload Section */}
75        <form onSubmit={handleUpload} style={{{
76          width: '100%',
77          maxWidth: '500px',
78          backgroundColor: '#fff',
79          padding: '30px',
80          borderRadius: '10px',
81          boxShadow: '0 4px 10px rgba(0,0,0,0.05)',
82          marginBottom: '40px',
83          textAlign: 'center'
84        }}>
85          <input
86            type="file"
87            onChange={(e) => setUploadFile(e.target.files[0])}
```
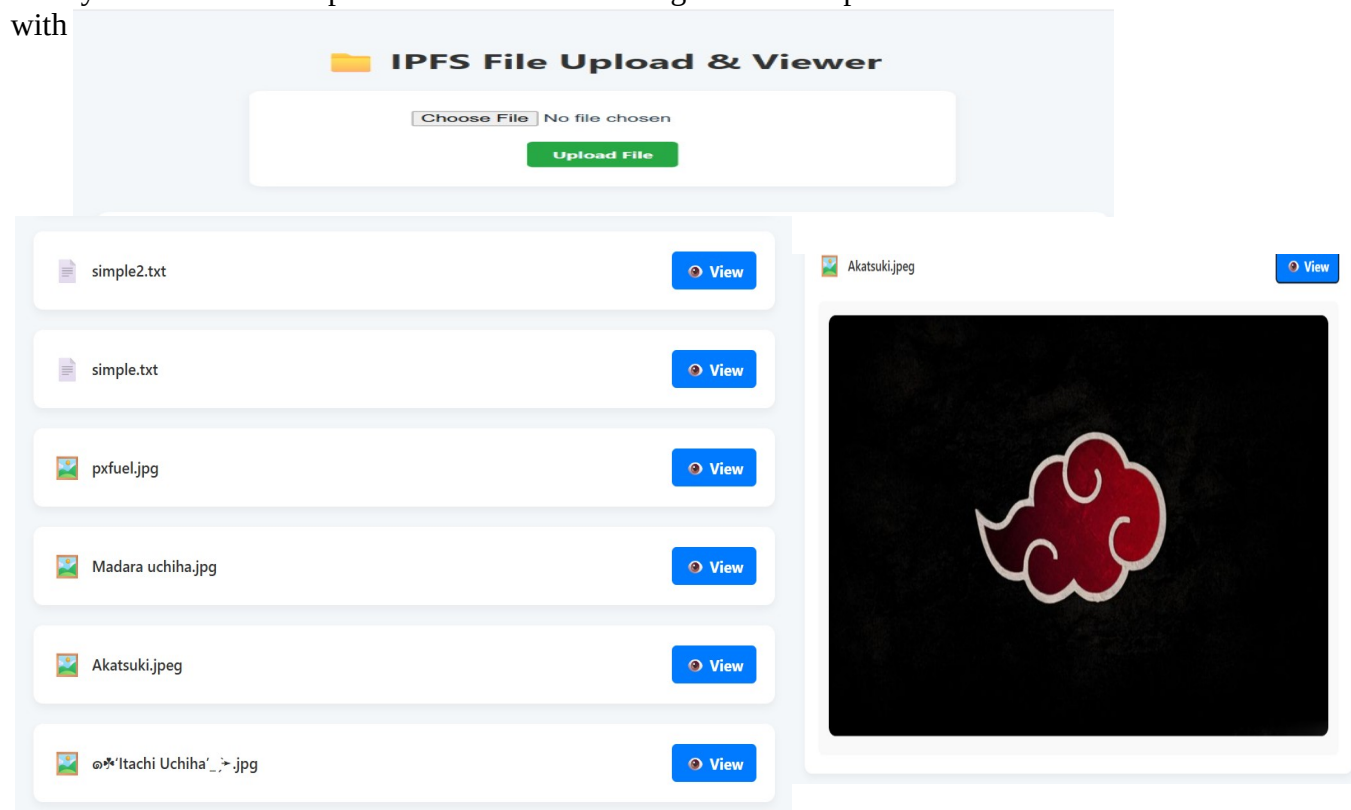
Now in .env file write your api key and secret api key

```
project > ⫶⫶ .env
  1   PINATA_API_KEY=86ef7b1dfd1d1577f3e9
  2   PINATA_SECRET_API_KEY=329b2b0b989573bfca3faec6da56832a22875be52a68fc19f3cf89e11c142c27
```

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 :

Signature of the Faculty:

Regn. No. :

Page No.............

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