Centurion University
Shaping Lives... Empowering Communities...

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

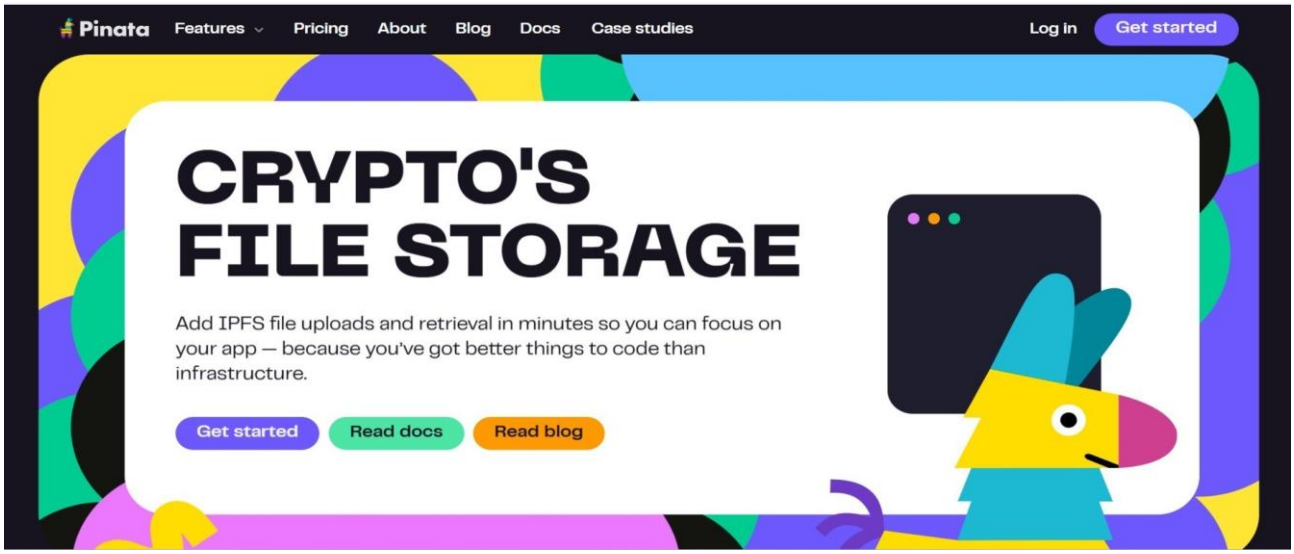
*Softwares required:

1.Brave Browser
2.Pinata
3.Vs Code

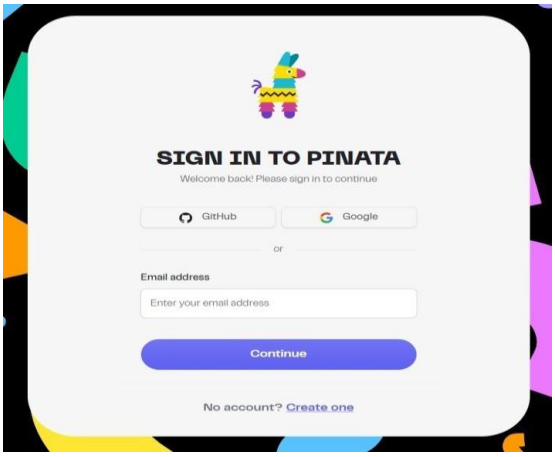
*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

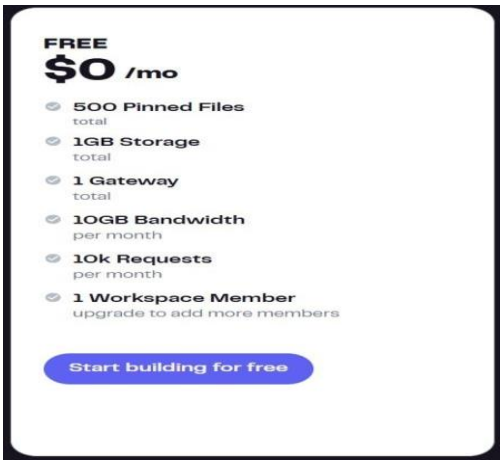
* Implementation Phase: Final Output (no error)

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).

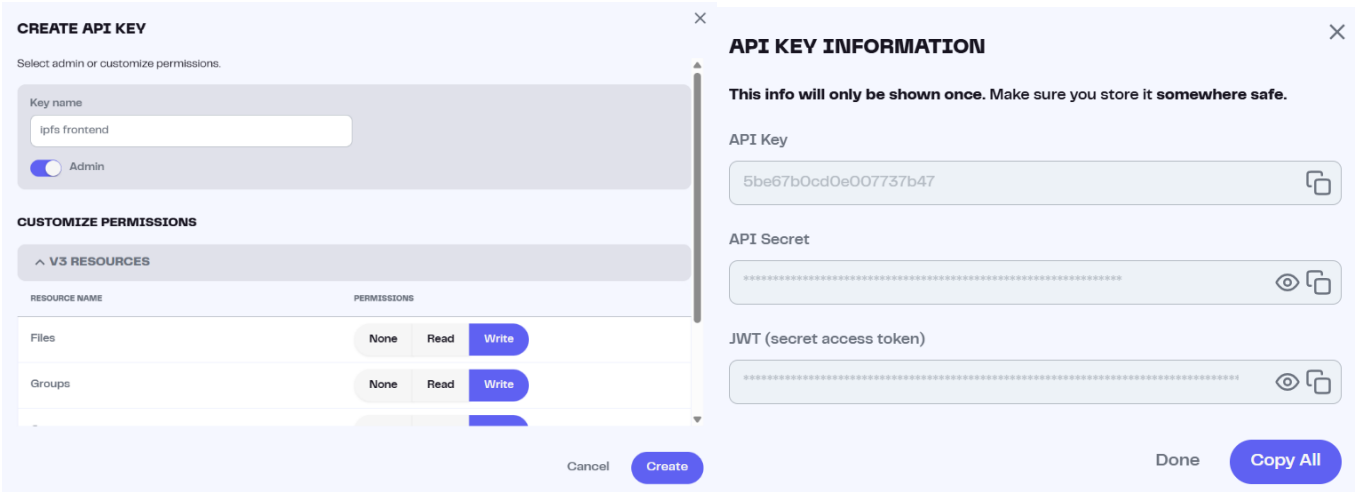


Click on "Login" and sign in using your credentials. If you're new, sign up and verify your email.





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



*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 index.js to upload files in pinata cloud and also create a app.jsx file for frontend

```
// backend/index.js
const express = require('express');
const multer = require('multer');
const cors = require('cors');
const fs = require('fs');
const FormData = require('form-data');
const axios = require('axios');
const mongoose = require("mongoose"); // MongoDB
require('dotenv').config();

const app = express();
const PORT = 5000;

// === MongoDB Connection ===
mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
mongoose.connection.once("open", () => {
  console.log("✅ Connected to MongoDB");
});

// === Middleware ===
app.use(cors());
app.use(express.json());

// === Multer (for file uploads) ===
const upload = multer({ dest: 'uploads/' });

// === Routes ===
app.get('/ping', (req, res) => {
  res.send('✅ Backend is alive!');
});

// === Auth Routes ===
const authRoutes = require("./routes/auth");
app.use("/api/auth", authRoutes);

// === Upload File to IPFS ===
app.post('/upload', upload.single('file'), async (req, res) => {
  if (!process.env.PINATA_JWT) {
    return res.status(500).json({ error: 'Missing Pinata JWT in .env' });
  }

  try {
    const fileStream = fs.createReadStream(req.file.path);
    const data = new FormData();
    data.append('file', fileStream);

    const metadata = JSON.stringify({
      name: req.body.name || req.file.originalname || 'NFT File',
    });
    data.append('pinataMetadata', metadata);

    const response = await axios.post(
      'https://api.pinata.cloud/pinning/pinFileToIPFS',
      data,
      {
        maxBodyLength: 'Infinity',
        headers: {
          'Content-Type': `multipart/form-data; boundary=${data._boundary}`,
          Authorization: `Bearer ${process.env.PINATA_JWT}`,
        },
      }
    );
    fs.unlink(req.file.path, (err) => {
      if (err) console.error('⚠️ Failed to delete temp file:', err.message);
    });
    res.status(200).json({
      ipfsHash: response.data.IpfsHash,
      ipfsURL: `https://gateway.pinata.cloud/ipfs/${response.data.IpfsHash}`,
    });
  } catch (err) {
    console.error('❌ IPFS File Upload Error:', err.message);
    res.status(500).json({ error: 'IPFS upload failed', details: err.message });
  }
});
// === Upload Metadata JSON to IPFS ===
app.post('/upload-metadata', async (req, res) => {
  if (!process.env.PINATA_JWT) {
    return res.status(500).json({ error: 'Missing Pinata JWT in .env' });
  }
  try {
    const { name, description, image } = req.body;
    if (!name || !description || !image) {
      return res.status(400).json({ error: 'Missing required fields: name, description, image' });
    }
    const metadata = { name, description, image };
    const response = await axios.post(
      'https://api.pinata.cloud/pinning/pinJSONToIPFS',
      metadata,
      {
        headers: {
          Authorization: `Bearer ${process.env.PINATA_JWT}`,
          'Content-Type': 'application/json',
        },
      }
    );
    res.status(200).json({
      ipfsHash: response.data.IpfsHash,
      metadataURL: `https://gateway.pinata.cloud/ipfs/${response.data.IpfsHash}`,
    });
  } catch (err) {
    console.error('❌ IPFS Metadata Upload Error:', err.message);
    res.status(500).json({ error: 'Metadata upload failed', details: err.message });
  }
});
// === Start Server ===
app.listen(PORT, () => {
  console.log(`🚀 Backend server 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)

```jsx
// frontend/src/App.jsx
import React, { useState } from "react";
import { BrowserRouter as Router, Routes, Route, Navigate, Link } from "react-router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.bundle.min.js";
import axios from "axios";

// ===== Navigation Bar with Wallet =====
function Navigation({ walletAddress, connectWallet }) {
  return (
    <nav className="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
      <div className="container">
        <Link className="navbar-brand" to="/home">🔗 CertifyChain</Link>
        <button className="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
          <span className="navbar-toggler-icon"></span>
        </button>
        <div className="collapse navbar-collapse" id="navbarNav">
          <ul className="navbar-nav ms-auto">
            <li classNameName="nav-item"><Link className="nav-link" to="/home">🏠 Home</Link></li>
            <li classNameName="nav-item"><Link className="nav-link" to="/nfts">🖼️ NFTs</Link></li>
            <li classNameName="nav-item"><Link className="nav-link" to="/about">ℹ️ About</Link></li>
            <li classNameName="nav-item"><Link className="btn btn-outline-light mx-2" to="/login">🔐 Login</Link></li>
            <li classNameName="nav-item"><Link className="btn btn-warning" to="/signup">🚀 Sign Up</Link></li>
            <li classNameName="nav-item">
              <button onClick={connectWallet} className="btn btn-success ms-2">
                {walletAddress ? `🟢 ${walletAddress.slice(0, 6)}...` : "🔗 Connect Wallet"}
              </button>
            </li>
          </ul>
        </div>
      </div>
    </nav>
  );
}

// ===== Pages =====
const HomePage = () => (
  <div className="container mt-5 pt-3">
    <h2>🏠 Welcome to CertifyChain</h2>
    <p>Build your blockchain-powered NFT portfolio with IPFS + on-chain minting!</p>
  </div>
);

const AboutPage = () => (
  <div className="container mt-5 pt-3">
    <h2>ℹ️ About CertifyChain</h2>
    <p>CertifyChain is a decentralized app (DApp) that allows you to upload files to IPFS and mint them as NFTs on-chain.</p>
  </div>
);

const LoginPage = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const handleLogin = async () => {
    try {
      const res = await axios.post("http://localhost:5000/api/auth/login", { email, password });
      alert("Login Successful");
      console.log(res.data);
    } catch (err) {
      alert("Login Failed: " + err.response?.data?.error);
    }
  };

  return (
    <div className="container mt-5 pt-3">
      <h2>🔐 Login</h2>
      <input type="email" placeholder="Email" className="form-control my-2" value={email} onChange={(e) => setEmail(e.target.value)} />
      <input type="password" placeholder="Password" className="form-control my-2" value={password} onChange={(e) => setPassword(e.target.value)} />
      <button className="btn btn-primary" onClick={handleLogin}>Login</button>
    </div>
  );
};

const SignupPage = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const handleSignup = async () => {
    try {
      const res = await axios.post("http://localhost:5000/api/auth/signup", { name, email, password });
      alert("Signup Successful");
      console.log(res.data);
    } catch (err) {
      alert("Signup Failed: " + err.response?.data?.error);
    }
  };

  return (
    <div className="container mt-5 pt-3">
      <h2>🚀 Sign Up</h2>
      <input type="text" placeholder="Name" className="form-control my-2" value={name} onChange={(e) => setName(e.target.value)} />
      <input type="email" placeholder="Email" className="form-control my-2" value={email} onChange={(e) => setEmail(e.target.value)} />
      <input type="password" placeholder="Password" className="form-control my-2" value={password} onChange={(e) => setPassword(e.target.value)} />
      <button className="btn btn-success" onClick={handleSignup}>Sign Up</button>
    </div>
  );
};

const ExplorePage = () => (
  <div className="container mt-5 pt-3">
    <h2>🔍 Explore NFT</h2>
    <p>Browse and view NFTs by CID.</p>
  </div>
);

// ===== Upload NFT Page =====
function MainApp() {
  const [selectedFile, setSelectedFile] = useState(null);
  const [name, setName] = useState("");
  const [description, setDescription] = useState("");
  const [metadataURL, setMetadataURL] = useState("");
  const [metadata, setMetadata] = useState(null);
  const [loading, setLoading] = useState(false);

  const handleFileChange = (e) => setSelectedFile(e.target.files[0]);

  const handleUpload = async () => {
    if (!selectedFile || !name || !description) return alert("Please fill all fields and select a file.");

    setLoading(true);
    try {
      const formData = new FormData();
      formData.append("file", selectedFile);
      formData.append("name", name);

      const fileRes = await axios.post("http://localhost:5000/upload", formData);
      const imageURL = fileRes.data.ipfsURL;

      const metadataRes = await axios.post("http://localhost:5000/upload-metadata", { name, description, image: imageURL });
      const metadataLink = metadataRes.data.metadataURL;
      setMetadataURL(metadataLink);

      const fetched = await axios.get(metadataLink);
      setMetadata(fetched.data);
    } catch (err) {
      console.error("Upload failed:", err);
      alert("Upload failed: " + err.message);
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="container mt-5 pt-3">
      <h2>🖼️ Upload NFT</h2>
      <input type="text" placeholder="NFT Name" value={name} onChange={(e) => setName(e.target.value)} className="form-control my-3" />
      <input type="text" placeholder="NFT Description" value={description} onChange={(e) => setDescription(e.target.value)} className="form-control my-3" />
      <input type="file" className="form-control" onChange={handleFileChange} />
      <button className="btn btn-primary mt-3" onClick={handleUpload} disabled={loading}>
        {loading ? "Uploading..." : "Upload NFT"}
      </button>
      {metadataURL && (
        <div className="mt-5 p-3 border rounded shadow-sm bg-light">
          <h4>✅ Metadata Uploaded</h4>
          <p><strong>Metadata URL:</strong> <a href={metadataURL} target="_blank" rel="noreferrer">{metadataURL}</a></p>
          {metadata && (
            <>
              <h5 className="mt-3">📌 Preview</h5>
              <p><strong>Name:</strong> {metadata.name}</p>
              <p><strong>Description:</strong> {metadata.description}</p>
              {metadata.image && <img src={metadata.image} alt={metadata.name} className="img-fluid rounded shadow-sm" style={{ maxWidth: 300 }} />}
            </>
          )}
        </div>
      )}
    </div>
  );
}

// ===== App Entry Point =====
export default function App() {
  const [walletAddress, setWalletAddress] = useState("");

  const connectWallet = async () => {
    if (window.ethereum) {
      const accounts = await window.ethereum.request({ method: "eth_requestAccounts" });
      const address = accounts[0];
      setWalletAddress(address);

      const res = await axios.post("http://localhost:5000/api/auth/wallet-login", { walletAddress: address });
      localStorage.setItem("token", res.data.token);
      alert("✅ Wallet connected and logged in!");
    } else {
      alert("MetaMask not detected!");
    }
  };

  return (
    <Router>
      <div style={{ paddingTop: "76px" }}>
        <Navigation walletAddress={walletAddress} connectWallet={connectWallet} />
        <Routes>
          <Route path="/" element={<Navigate to="/home" />} />
          <Route path="/home" element={<HomePage />} />
          <Route path="/about" element={<AboutPage />} />
          <Route path="/login" element={<LoginPage />} />
          <Route path="/signup" element={<SignupPage />} />
          <Route path="/nfts" element={<MainApp />} />
          <Route path="/explore/:cid" element={<ExplorePage />} />
        </Routes>
      </div>
    </Router>
  );
}
```

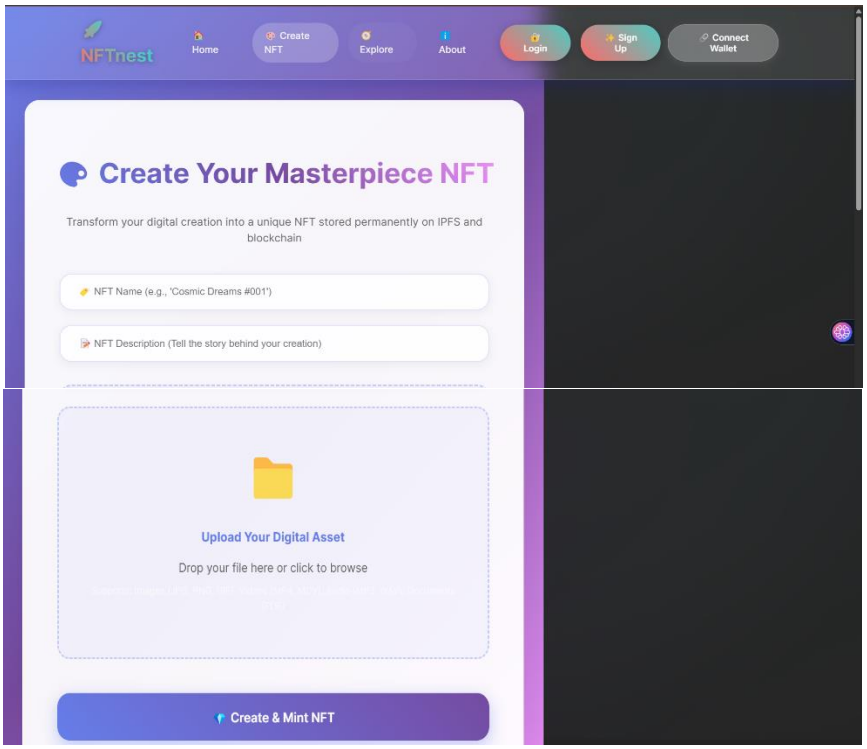

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

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

```
# === Pinata API ===
PINATA_API_KEY=447364863125b2fa779b
PINATA_API_SECRET=56ecf2e68a9f08ddf1ccc18f626313f66171d41486258c27ecbfe4fa71b9afc8
PINATA_JWT=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySW5mb3JtYXRpb24iOnsiaWQiOiJlYTVkZDJkZS03YTcwLTQxNzktYTlhOC0xN
MONGO_URI=mongodb+srv://pritam:Das%40078@nft-portfolio-dapp.hos8zww.mongodb.net/?retryWrites=true&w=majority&appName=
```

Now you see the file is uploaded and the hash was generated in pinata add file section we can alse see with view section and add files to our pinata account through frontend



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.