📌 Step 1: Project Setup

✅ Initialize a Node.js Project

1. **Open a terminal and navigate to your desired project folder:**
2.
3. **mkdir file-manager-app**
4. **cd file-manager-app**
5. **Initialize a new Node.js project:**
6.
7.
8. **npm init -y**
9. **This will create apackage.json file.**

📌 Step 2: Install Required Dependencies

✅ Install Express & CORS

Since we are building an API, we need:

- **Express (for creating routes)**
- **CORS (to allow frontend to communicate with the backend)**

Run:

npm install express cors

📌 Step 3: Create Project Structure

Inside the file-manager-app folder, create the following structure:

php

file-manager-app/

│ —— public/ # Frontend files

│ ├—— index.html # User Interface

│ ├—— script.js # Handles frontend interactions

```
├── routes/           # Backend routes
│   ├── fileRoutes.js    # API routes for file operations
├── storage/          # Directory for storing files
├── app.js            # Main server file
├── package.json      # Project dependencies
```

📌 Step 4: Create app.js (Main Server)

This is the entry point of the application.

📝 **Create app.js and add this code:**

```
const express = require('express');

const express = require('express');

const cors = require('cors');

const fileRoutes = require('./routes/fileRoutes');


const app = express();

app.use(cors());

app.use(express.json());

app.use(express.static('public')); // Serve frontend

app.use('/api', fileRoutes);


app.listen(3000, () => console.log('Server running at http://localhost:3000'));
```

📌 What This Code Does

1. **Imports required modules (express, cors, fileRoutes).**
2. **Enables CORS so the frontend can communicate with the backend.**
3. **Parses incoming JSON data using express.json().**

4. **Serves the frontend (public/ folder) using express.static().**
5. **Uses the fileRoutes.js file for handling file operations.**
6. **Starts the server on port 3000.**

📌 Step 5: Create routes/fileRoutes.js (File Operations API)

This file contains all the backend logic to read, write, delete, and manage files.

📝 Create routes/fileRoutes.js and add this code:

```
const express = require('express');

const fs = require('fs').promises;

const path = require('path');


const router = express.Router();


// Helper function to get the absolute file path (prevents security issues)

const getFilePath = (fileName) => path.join(__dirname, '..', 'storage',
path.basename(fileName));


// 📌 Read File

router.get('/read', async (req, res) => {

 try {

  const data = await fs.readFile(getFilePath(req.query.fileName), 'utf8');

  res.json({ content: data });

 } catch (err) {

  res.status(404).json({ error: 'File not found' });

 }
```

```
});

// 📌 Write File
router.post('/write', async (req, res) => {
  try {
    await fs.writeFile(getFilePath(req.body.fileName), req.body.content, 'utf8');
    res.json({ message: 'File written successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// 📌 Append to File
router.post('/append', async (req, res) => {
  try {
    await fs.appendFile(getFilePath(req.body.fileName), req.body.content, 'utf8');
    res.json({ message: 'Content appended successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// 📌 Rename File
router.put('/rename', async (req, res) => {
  const { oldName, newName } = req.body;
```

```
  if (!oldName || !newName) return res.status(400).json({ error: 'Both file names are required' });

  try {
    await fs.rename(getFilePath(oldName), getFilePath(newName));
    res.json({ message: 'File renamed successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// 📌 Delete File
router.delete('/delete', async (req, res) => {
  try {
    await fs.unlink(getFilePath(req.query.fileName));
    res.json({ message: 'File deleted successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// 📌 Create Directory
router.post('/create-dir', async (req, res) => {
  try {
    await fs.mkdir(getFilePath(req.body.dirName), { recursive: true });
    res.json({ message: 'Directory created successfully' });
```

```
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// 📌 Delete Directory
router.delete('/delete-dir', async (req, res) => {
  try {
    await fs.rm(getFilePath(req.query.dirName), { recursive: true, force: true });
    res.json({ message: 'Directory deleted successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

📌 What This Code Does

✅ Provides API endpoints to:

- Read, write, append, rename, and delete files
- Create and delete directories

✅ Uses fs.promises for async file operations.

✅ Prevents security vulnerabilities (e.g., prevents users from accessing system files).

📌 Step 6: Create public/index.html (Frontend UI)

📝 Create public/index.html and add this code:

```
html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>File Manager</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    input, textarea { display: block; margin-bottom: 10px; width: 300px; }
    button { margin-bottom: 20px; }
    pre { background: #f4f4f4; padding: 10px; }
  </style>
</head>
<body>
  <h1>File Manager</h1>

  <section>
    <h2>Read File</h2>
    <input type="text" id="readFileName" placeholder="Enter file name">
    <button onclick="readFile()">Read File</button>
    <pre id="readContent"></pre>
  </section>

  <section>
    <h2>Write File</h2>
    <input type="text" id="writeFileName" placeholder="Enter file name">
    <textarea id="writeContent" placeholder="Enter content"></textarea>
```

```
  <button onclick="writeFile()">Write File</button>
</section>

<section>
  <h2>Append to File</h2>
  <input type="text" id="appendFileName" placeholder="Enter file name">
  <textarea id="appendContent" placeholder="Enter content"></textarea>
  <button onclick="appendFile()">Append to File</button>
</section>

<section>
  <h2>Delete File</h2>
  <input type="text" id="deleteFileName" placeholder="Enter file name">
  <button onclick="deleteFile()">Delete File</button>
</section>

<section>
  <h2>Rename File</h2>
  <input type="text" id="oldFileName" placeholder="Enter current file name">
  <input type="text" id="newFileName" placeholder="Enter new file name">
  <button onclick="renameFile()">Rename File</button>
</section>

<section>
  <h2>Create Directory</h2>
  <input type="text" id="createDirName" placeholder="Enter directory name">
  <button onclick="createDirectory()">Create Directory</button>
```

```
  </section>

  <section>
    <h2>Delete Directory</h2>
    <input type="text" id="deleteDirName" placeholder="Enter directory name">
    <button onclick="deleteDirectory()">Delete Directory</button>
  </section>

  <script src="script.js"></script>
</body>
</html>
```

📌 Step 7: Create public/script.js (Frontend Logic)

📝 Create public/script.js and add this code:

```
const apiUrl = 'http://localhost:3000/api';

async function readFile() {
  const fileName = document.getElementById('fileName').value;
  const response = await fetch(`${apiUrl}/read?fileName=${fileName}`);
  const data = await response.json();
  document.getElementById('output').textContent = data.content || data.error;
}
```

📌 Step 8: Run the Project

1. **Start the server:**
2.
3.
4. **node app.js**
5. **Open http://localhost:3000 in the browser.**
6. **Enter a file name and click Read File to test.**

🎯 Summary

✅ What Students Learn

- **How to set up an Express project with routes.**
- **How to interact with the filesystem in Node.js.**
- **How to build a frontend that communicates with a backend.**