# **📄 Internship Task 3: Secure File Sharing System**

## **👨‍💻 Submitted By:**

**Name:** Anurag Aryan  
 **Internship Role:** Cybersecurity Intern  
 **Task Title:** Secure File Sharing System using AES Encryption  
 **Tools Used:** Node.js, Express, Multer, Crypto Module, HTML, dotenv, Git & GitHub

## **🧠 Objective**

The objective of this task is to develop a secure file sharing portal that enables users to **upload and download files** securely. The files are encrypted using **AES-256 encryption**, ensuring confidentiality at rest and in transit.

## **💡 Skills Gained**

* Node.js Web Development
* AES Encryption and Decryption
* File Upload and Handling via Multer
* Environment Variable Management
* Secure Key Handling
* API Development and Testing

## **🛠️ Tools & Technologies**

|  |  |
| --- | --- |
| **Tool/Tech** | **Purpose** |
| Node.js | Backend server |
| Express | Web framework |
| Multer | File upload handling |
| Crypto | AES encryption/decryption |
| dotenv | Managing secret keys |
| HTML | Frontend form |
| Git & GitHub | Version control |

## **📂 Folder Structure**

pgsql

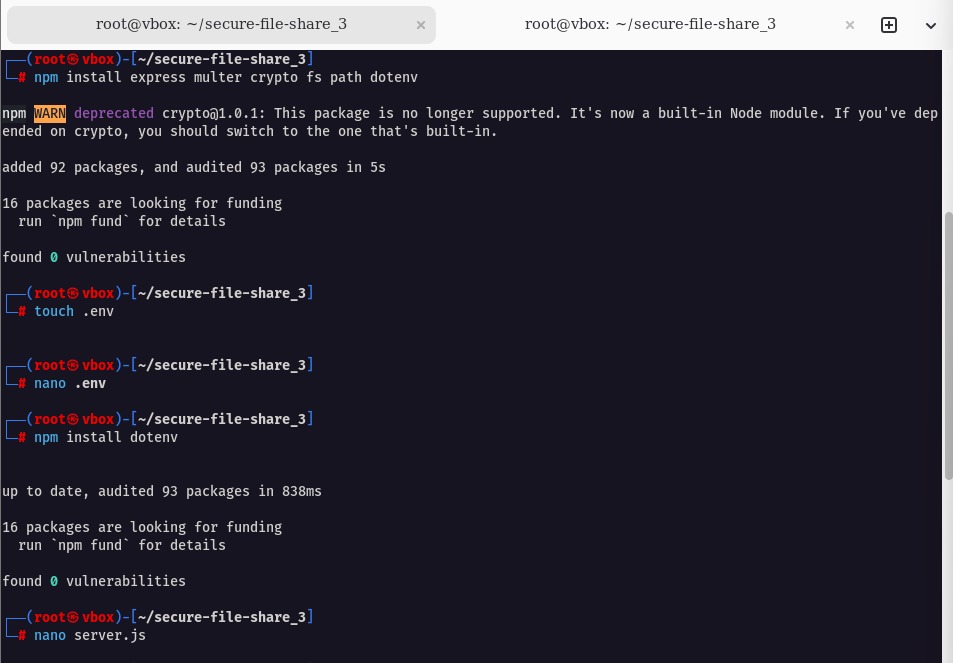
CopyEdit

secure-file-share/  
├── index.html ← Frontend for Upload/Download  
├── server.js ← Main backend logic  
├── .env ← Secret key and IV  
├── uploads/ ← Encrypted files  
├── decrypted/ ← Temporary decrypted files  
├── .gitignore ← To ignore sensitive files  
├── package.json



## **🔐 How AES Encryption Works**

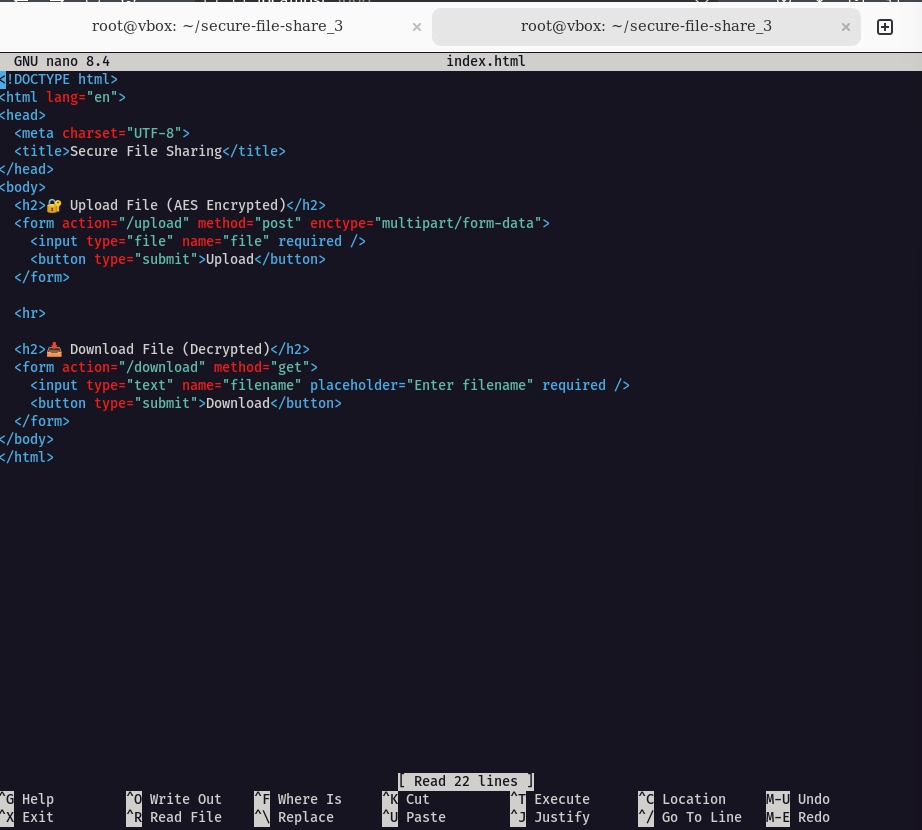
1. When a file is uploaded, it’s encrypted using **AES-256-CBC** with a key and IV from the .env file.
2. The encrypted file is saved in the uploads/ folder.
3. When a user requests a download, the file is decrypted and sent as a response.
4. The decrypted file is deleted immediately after download.

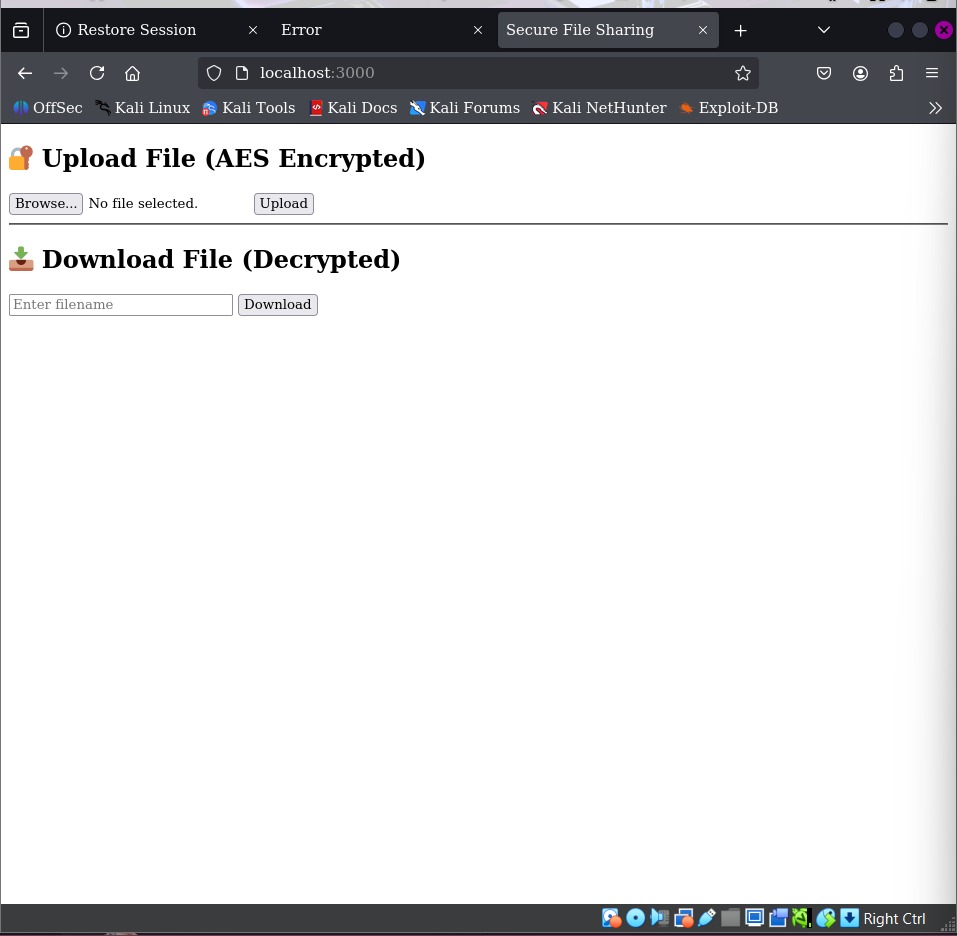


## **🌐 Frontend – HTML Upload/Download Interface**

A minimal HTML form was created to allow users to upload and download files easily without tools like Postman.

*Creating HTML file*

  
 *Upload & Download Interface*



## **🧾 API Endpoints**

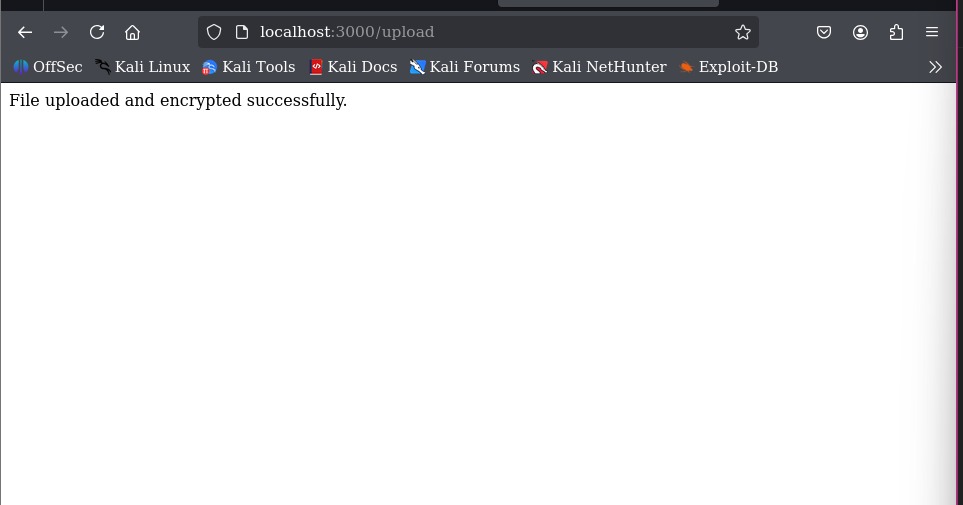
|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| POST | /upload | Upload and encrypt a file |
| GET | /download?filename=name.txt | Decrypt and download the file |

## **🧪 Testing the Application**

### **✅ Upload**

* The user selects a file using the form and submits it.
* File is encrypted and stored securely.

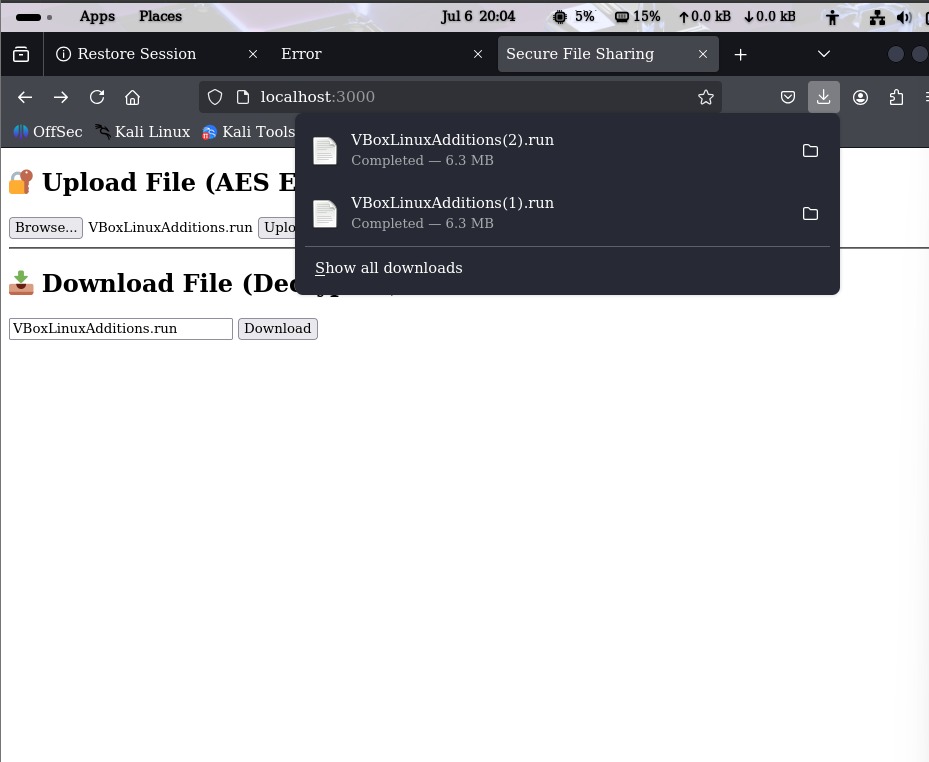
*File uploaded successfully*



### **✅ Download**

* User enters the filename.
* File is decrypted and downloaded to the system.
* Temporary decrypted version is deleted.

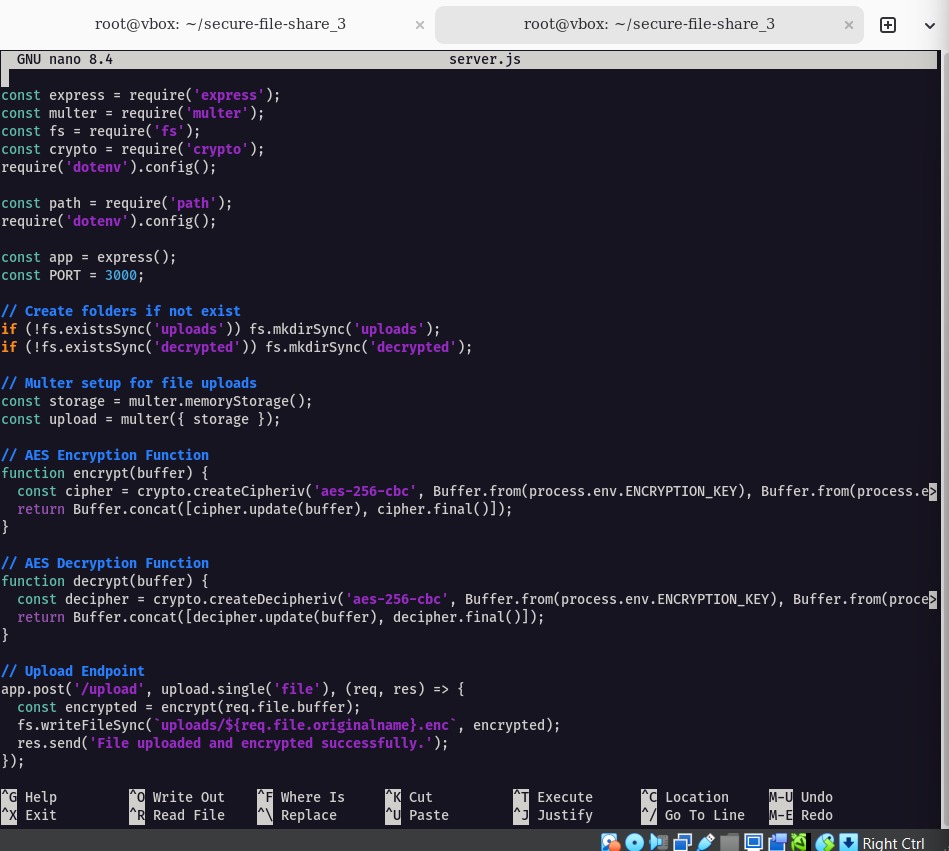
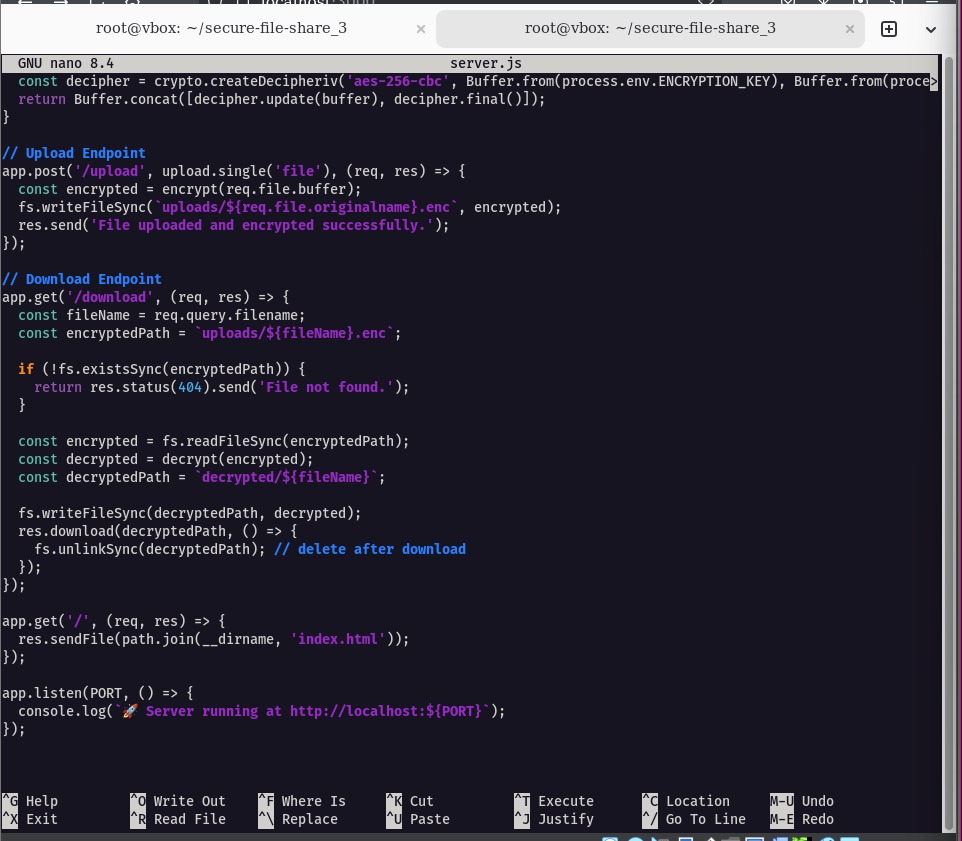
*File downloaded*

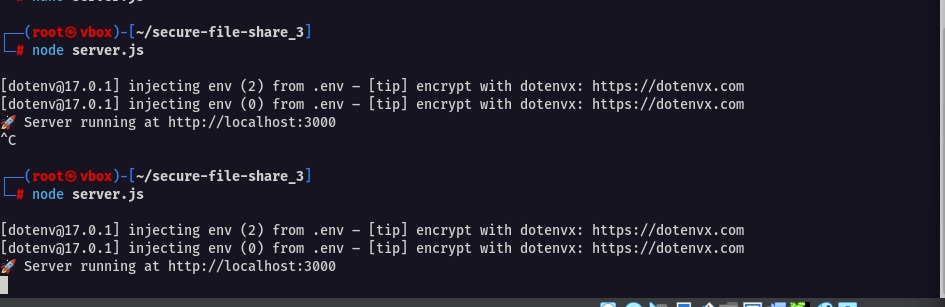


## **📜 Code Highlights**

* **File upload via Multer (memory storage)**
* **AES-256-CBC Encryption using crypto**
* **.env file securely holds ENCRYPTION\_KEY and IV**
* **File auto-deletion after download**

*Node.js file (main logic)*

  
   
 *Running server on port 3000*



## **🔐 Security Measures Implemented**

* AES-256 encryption ensures strong file confidentiality.
* Environment variables prevent key leakage.
* .gitignore used to prevent .env from being uploaded to GitHub.
* No file persists in decrypted state after use.
* Memory storage used for extra file safety.

## **📌 GitHub Repository**

🔗 GitHub repo link - https://github.com/anurag7654

## **📷 Screenshots**

|  |  |
| --- | --- |
| **Screenshot No.** | **Description** |
| 1 | Creating .env file |
| 2 | Creating project folder and init |
| 3 | Creating index.html |
| 4 | File uploaded successfully (browser) |
| 5 | File downloaded (browser) |
| 6 | Interface for file upload/download |
| 7 | Node.js backend file (code) |
| 8 | Node.js footer (app.listen) |
| 9 | Server running on port 3000 |

## **✅ Conclusion**

This task was successfully completed using secure coding practices and AES encryption standards. It allowed practical exposure to secure web development, key management, and handling sensitive file uploads and downloads in a safe manner.