SECURE FILE SHARING SYSTEM REPORT

🔐 By: Lendani Canvil Kwati

# Project Objectives and Key Features

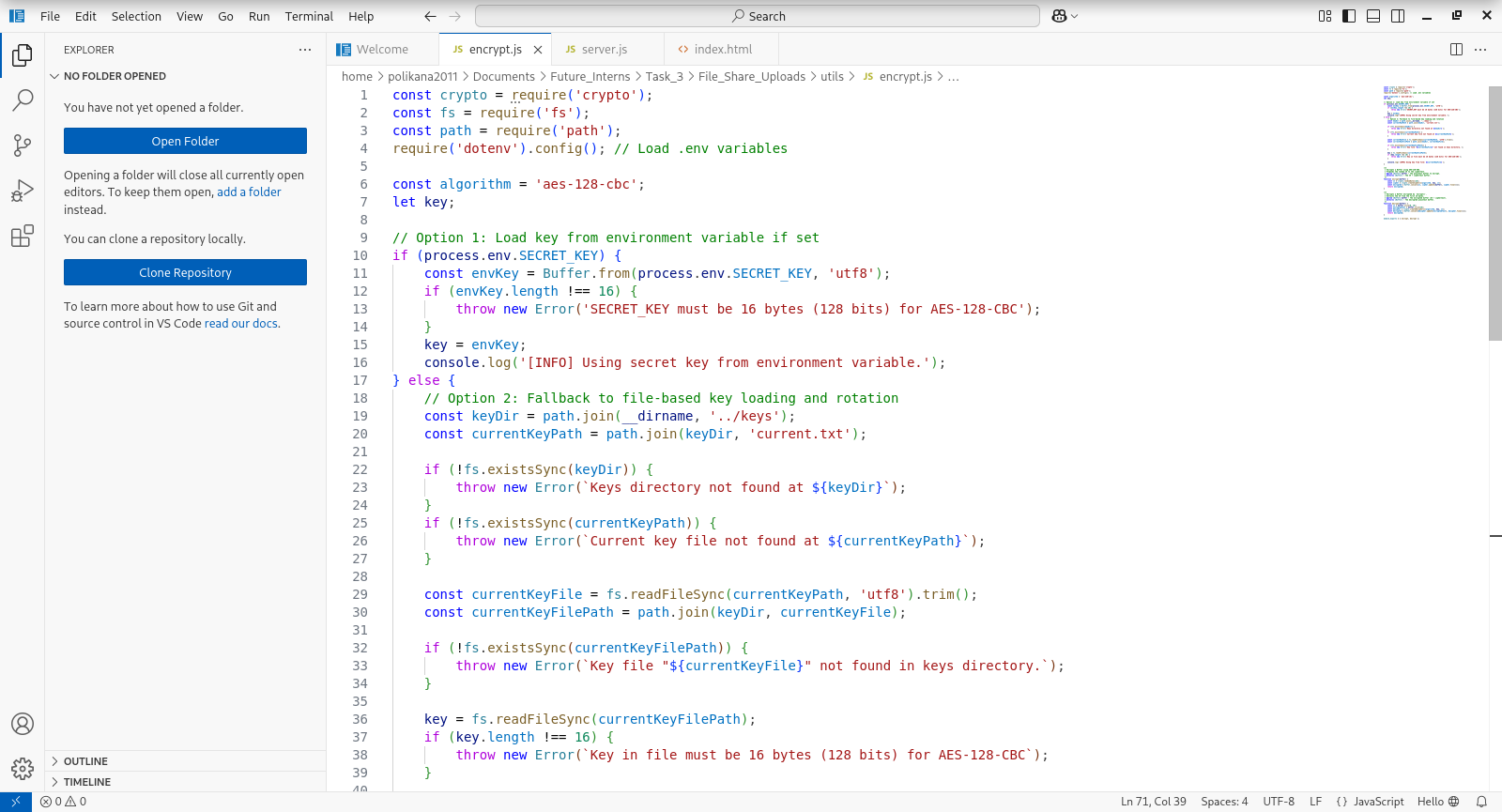
This project implements a secure file sharing system using Node.js, Express, and AES encryption. The system fulfills the following objectives:

* Secure file upload and download functionality
* AES encryption (AES-128-CBC) for files at rest
* Basic encryption key management with environment variable or file rotation
* User-friendly web interface for file handling
* Well-documented and modular codebase

# Security Overview

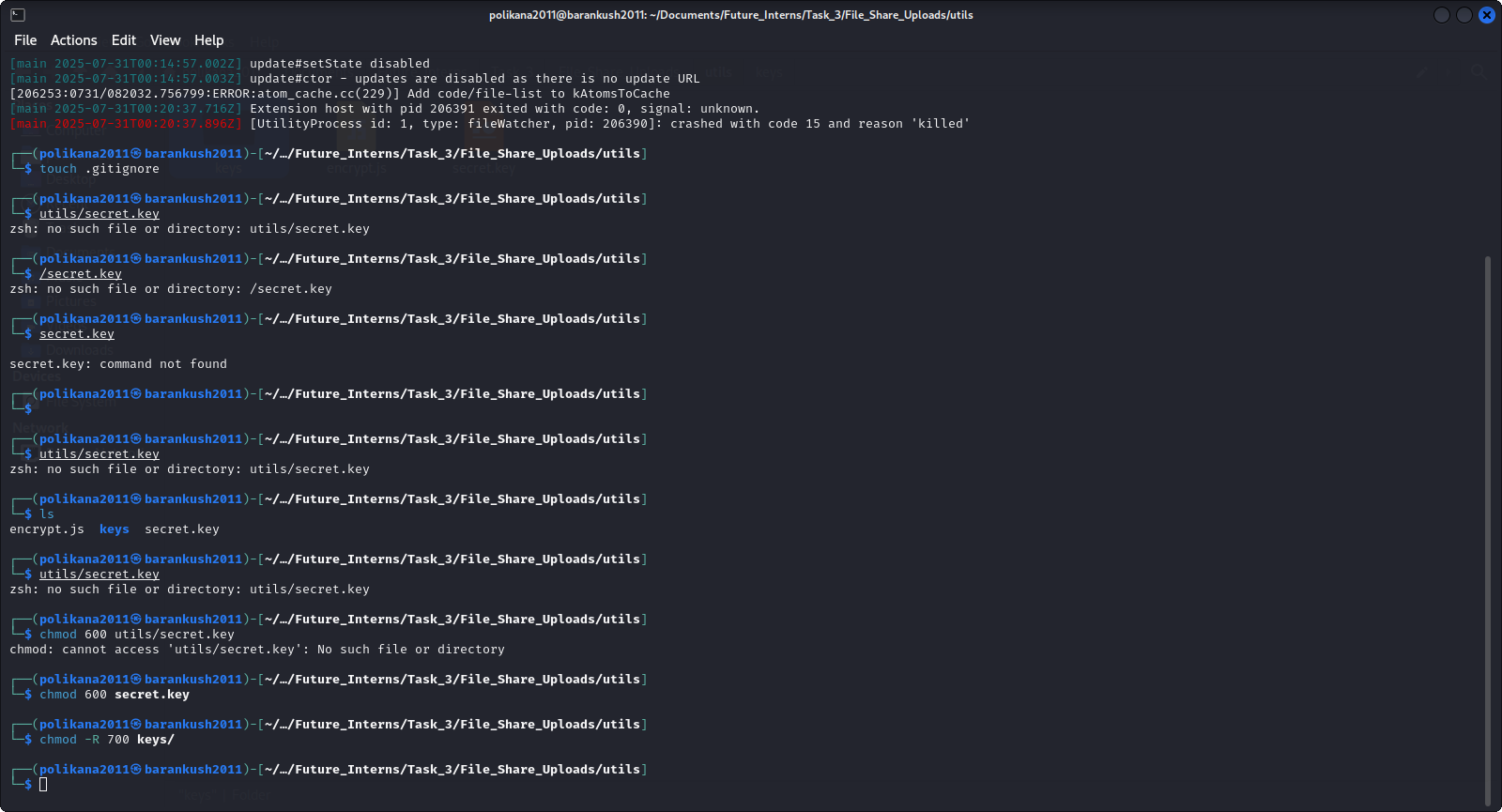
## Encryption Method: AES-128-CBC

* Algorithm: AES (Advanced Encryption Standard)
* Mode: CBC (Cipher Block Chaining)
* Key size: 128 bits (16 bytes)
* IV (Initialization Vector): Random 16-byte IV is generated per encryption and prepended to the encrypted content.



## Key Management

The system supports two options for key management:

* Environment Variable (.env) – the recommended production method.
* File-based Rotation – fallback method for development/testing. Reads current key from keys/current.txt.  
    
  (The key must always be 16 bytes to match AES-128 requirements.)  
  

## Security Considerations

* Encrypted files are stored on disk with only the IV prepended.
* Decryption occurs only when files are downloaded.
* The use of HTTPS is recommended for deployment.
* Avoid exposing the keys/ directory publicly.

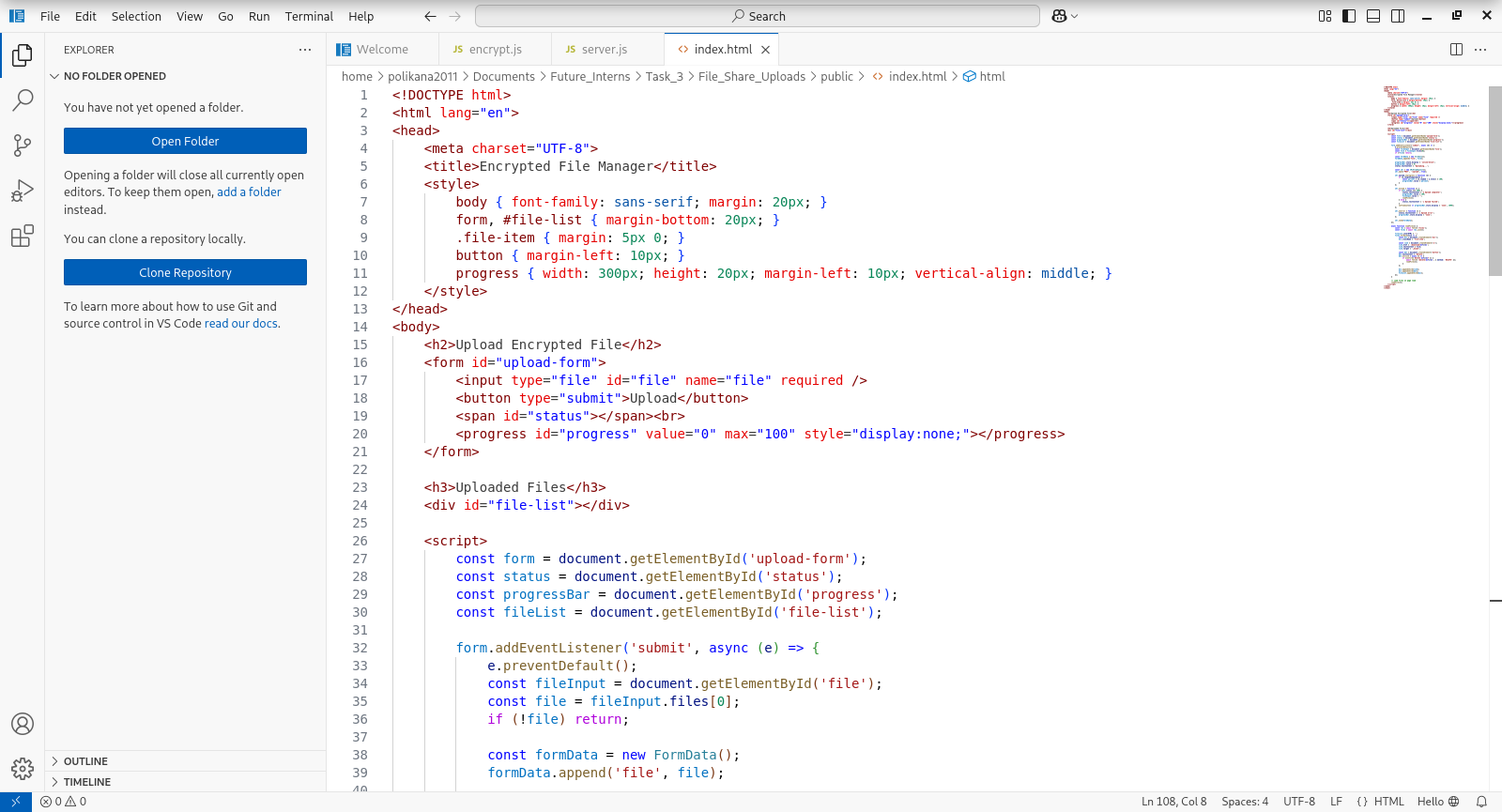
# File Structure

project-root/  
├── uploads/ Stores encrypted uploaded files  
├── public/ Contains frontend HTML/JS  
├── utils/encrypt.js Encryption and decryption logic  
├── server.js Main Express server  
└── keys/ Stores key files and key reference file

# Functional Description

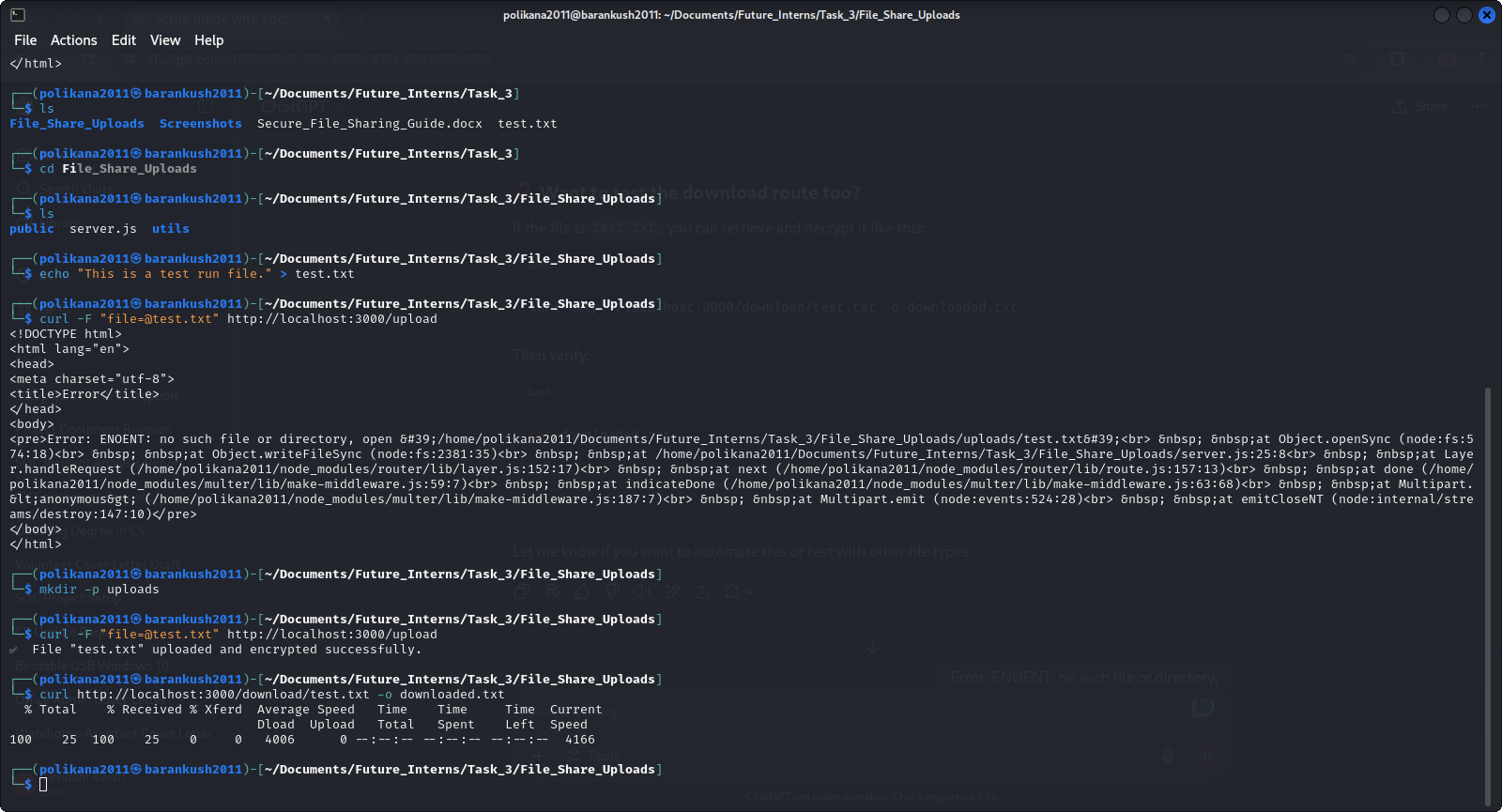
## File Upload and Encryption

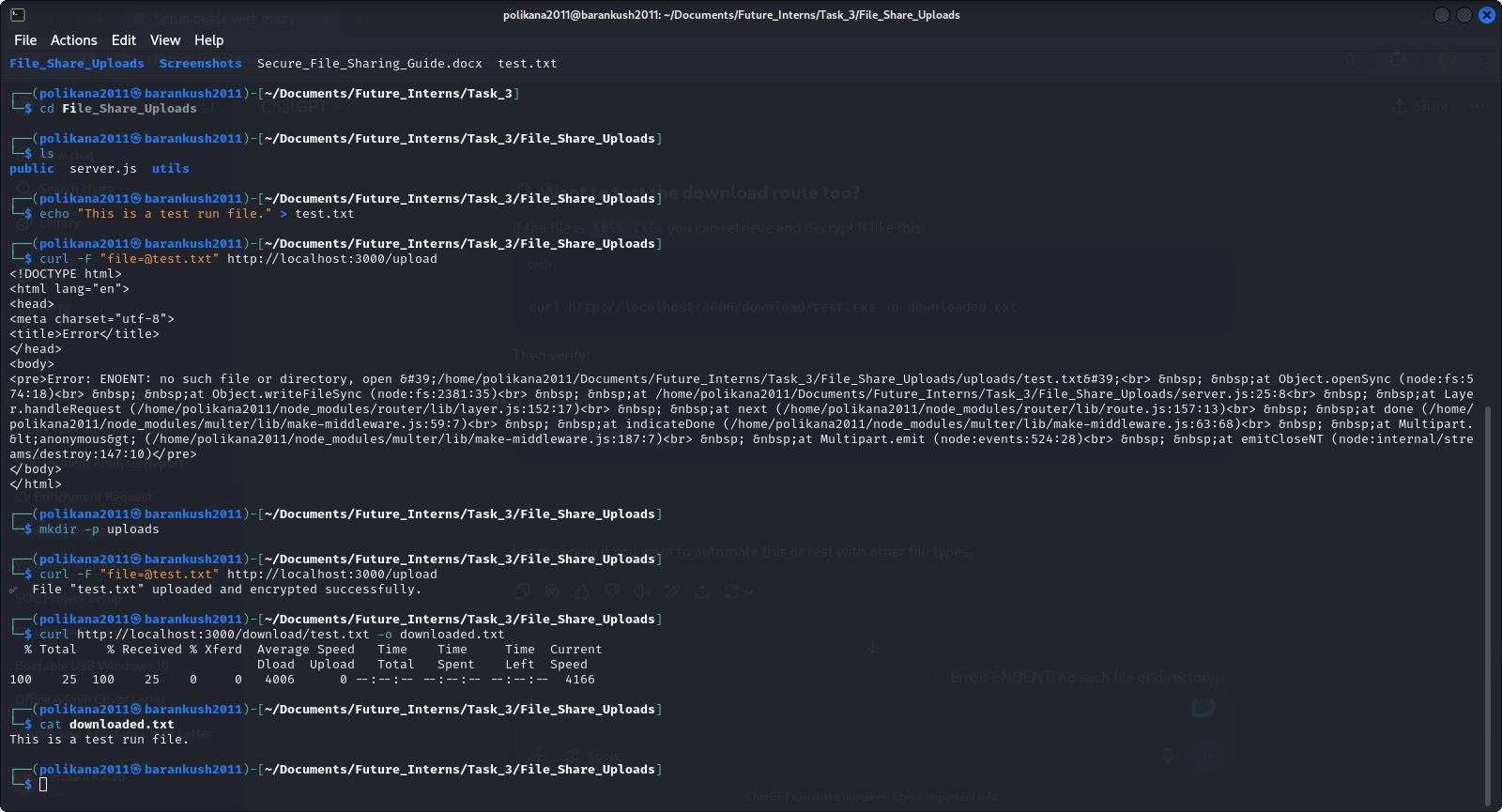
* Users upload files via the web interface.
* Files are received as memory buffers using multer.
* The buffer is encrypted using AES-128-CBC.
* The IV is prepended to the ciphertext and saved to disk in the uploads/ directory.



## File Download and Decryption

* The server reads the encrypted file from disk.
* The IV is extracted from the first 16 bytes.
* File is decrypted and served to the user as a download.



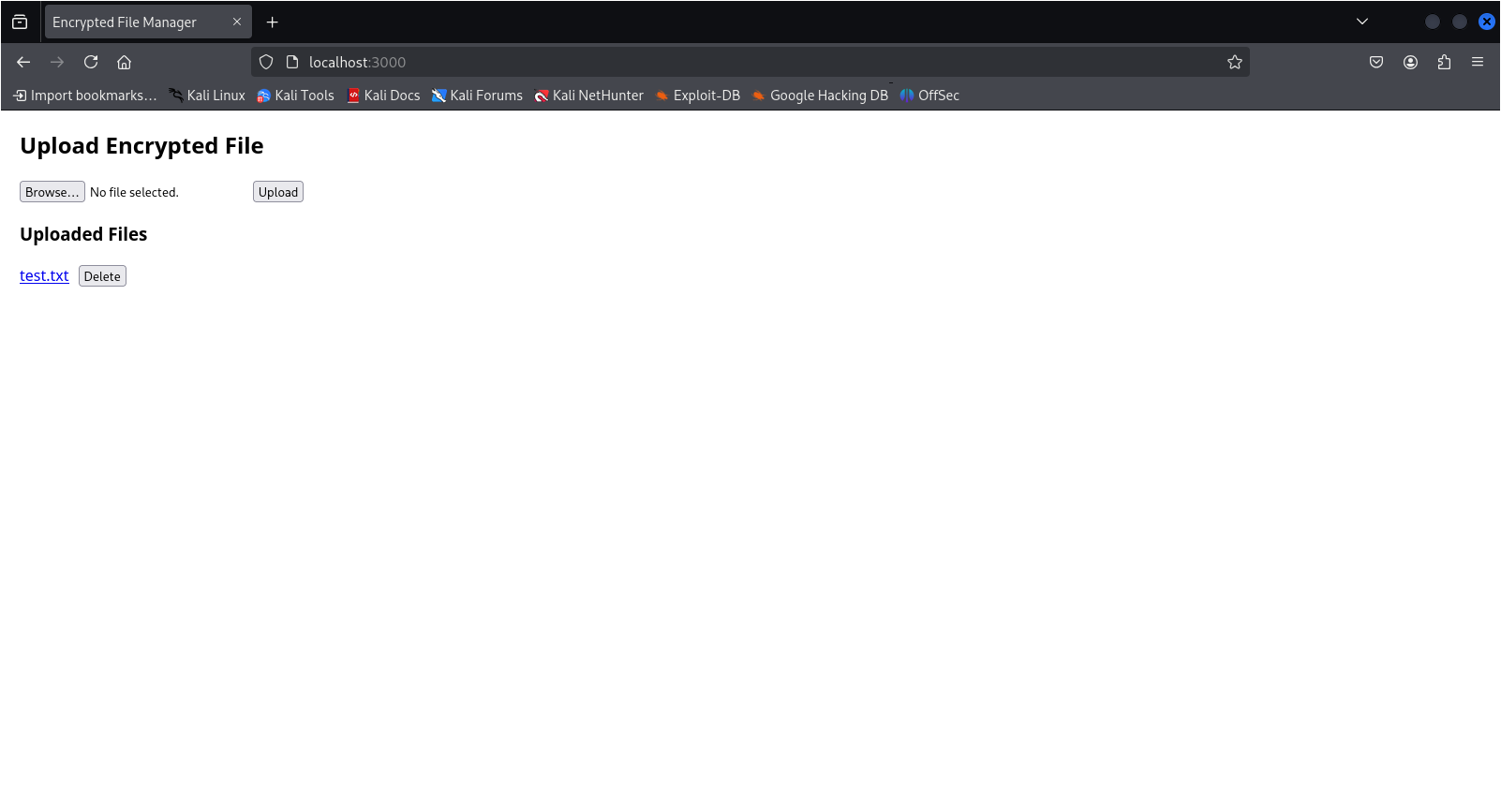


## File Listing and Deletion

* Lists all filenames in the uploads/ folder.
* Allows file deletion with confirmation.

# 📅 **User Interface**

* A simple web interface allows users to upload files and view/download/delete them.
* Upload progress is displayed using a progress bar.



# Server Routes Overview

The server defines endpoints for:

* /upload (POST): Accepts file uploads and encrypts them.
* /download/:filename (GET): Serves decrypted files for download.
* /files (GET): Lists all uploaded files.
* /delete/:filename (DELETE): Deletes a specific file.

