STEALTH GUARD
Private Limited

MACHINE TEST

Question: Secure File Upload System Development

You have been assigned the task of building a secure file upload system in PHP. Users should have the ability to upload files, but your primary concern is implementing robust security measures to safeguard the system against malicious uploads and potential attacks. You are required to create three tables: `user`, `uploads`, and `logs`. Below are the specific requirements:

User Table:

- Store user information, including encrypted passwords.
- Implement user authentication, ensuring that only authenticated users can upload files.

Uploads Table:

- Store information about uploaded files, including the filename and associated user.

Logs Table:

- Keep a record of all file uploads, including the uploader's IP address, file name, and upload timestamp.
- Log any suspicious or rejected uploads for security monitoring.

Your PHP-based secure file upload system should include the following security features and functionality:

1. File Type Validation:

- Develop a function that verifies the file type of uploaded files. Accept only files with extensions `.jpg`, `.png`, `.pdf`, and `.docx`. Reject uploads with any other file extensions.

2. File Size Limit:

- Set a maximum file size limit for uploads (e.g., 5MB). Reject files that exceed this limit.

3. File Name Sanitization:

- Implement a mechanism to sanitize uploaded file names, removing potentially harmful characters and symbols.

- Ensure that each uploaded file has a unique name to prevent overwriting existing files.

4. File Upload Directory:

- Specify a secure directory for file uploads that is located outside the web server's root directory.
- Ensure that uploaded files cannot be executed as scripts or accessed directly through the web server.

5. Logging:

- Implement logging for all file uploads, recording the uploader's IP address, file name, and upload timestamp.
- Log any suspicious or rejected uploads for security auditing purposes.

6. Security Headers:

- Configure the server to send appropriate security headers in the HTTP response to mitigate potential cross-site scripting (XSS) or content sniffing attacks.

7. User Authentication:

- If not already implemented, create a user authentication system to restrict file uploads to authorized users only.

8. Testing:

- Provide a comprehensive test script that demonstrates the functionality of your secure file upload system.
- Include scenarios for both successful file uploads and rejected upload attempts.

Submission Requirements:

Candidates are expected to develop the secure file upload system using Core PHP. Additionally, they must:

- Create a public GitHub repository to host the source code.
- Share the repository link for evaluation.
- Provide clear documentation on how to use and test the secure file upload system.

This question assesses the candidate's ability to develop a secure PHP application, implement security best practices, and document their work effectively.