**Documentation: Audit Log Implementation (Including Error Handling)**

**🎯 Purpose**

This document outlines the requirement to implement a centralized **Audit Log** system for the application. The audit log must record **critical user activities** (e.g., logins, logouts, access attempts) and **system errors** (for security, compliance, and troubleshooting purposes).

**📌 Requirements Overview**

**✅ 1. Audit Log Functionality**

Implement a logging mechanism to capture and store key application events, including:

**🔐 User Activity Events:**

* Successful logins
* Failed login attempts
* Logouts
* Session expirations
* Account lockouts

**⚠️ Error Events:**

* Unhandled exceptions
* Critical failures (e.g., database issues, API errors)
* Application crashes

**🧩 Technical Implementation**

**🗂️ Audit Log Structure**

Each audit log entry must include:

| **Field** | **Description** |
| --- | --- |
| timestamp | UTC time of the event |
| event\_type | login\_success, login\_failure, error, etc. |
| user\_id or email | User identifier if applicable |
| ip\_address | IP address of the client (if available) |
| device\_info | Browser or platform used |
| action\_description | Description of the action or error |
| stack\_trace | (Optional) Full stack trace for error events |

**💻 Sample Log Entry (JSON)**

json

CopyEdit

{

"timestamp": "2025-06-11T15:45:30Z",

"event\_type": "login\_success",

"user\_id": "user@example.com",

"ip\_address": "192.168.0.101",

"device\_info": "Chrome on Windows 10",

"action\_description": "User successfully logged in"

}

json

CopyEdit

{

"timestamp": "2025-06-11T15:46:02Z",

"event\_type": "error",

"user\_id": "user@example.com",

"action\_description": "Unhandled exception during profile update",

"exception\_type": "DatabaseConnectionError",

"stack\_trace": "Traceback (most recent call last)..."

}

**🔧 Implementation Notes**

**✅ 2. Logging Location**

Audit logs should be stored in one of the following:

* A dedicated log file (e.g., audit\_log.json or audit\_log.txt)
* A database table (e.g., audit\_logs)
* An external logging system (e.g., Loggly, Sentry, ELK Stack) — optional

**✅ 3. Error Handling Integration**

**Global Try-Catch Pattern:**

Wrap key areas of the application with error handling logic that logs to the audit log.

python

CopyEdit

import traceback

from datetime import datetime

def log\_audit\_event(event\_type, user=None, description="", error=None):

entry = {

"timestamp": datetime.utcnow().isoformat(),

"event\_type": event\_type,

"user\_id": user or "anonymous",

"action\_description": description

}

if error:

entry["exception\_type"] = type(error).\_\_name\_\_

entry["stack\_trace"] = traceback.format\_exc()

with open("audit\_log.json", "a") as log\_file:

log\_file.write(f"{entry}\n")

**✅ Action Items for Replit Team**

Please implement a centralized **audit log system** that captures:

* User login/logout and authentication-related events
* System and application-level errors using structured error handling
* Logs should be stored in a file (audit\_log.json) or a persistent database table for future review and analysis