Digital Chain of Custody Web Application: Project Documentation

1. Project scope statement

1.1. Project goals and objectives

The goal of this project is to develop a secure web-based application that demonstrates the principles of a digital chain of custody. The system will serve as an academic exercise to track and manage evidence for school disciplinary and class-related cases.

1.2. Key objectives

- Provide secure user authentication for multiple user roles.
- Allow users to create new cases and add evidence with metadata.
- Record and track the transfer of custody for each piece of evidence.
- Implement data integrity checks using cryptographic hashing.
- Ensure evidence and case information are protected through various security measures.
- Generate printable reports detailing the complete chain of custody for a case.

1.3. Deliverables

- A working web application with a secure login.
- A case management dashboard for viewing and managing cases.
- Functionality to add, transfer, and view evidence details.
- A detailed, tamper-evident audit trail for all evidence actions.
- A web-based interface that integrates a map display to visualize evidence locations.
- Documentation covering the system architecture, features, and security measures.

1.4. Exclusions

- Integration with live school databases or systems (all data will be simulated).
- Use of production-level blockchain technology or advanced IoT hardware.
- External communication features (e.g., email notifications) are not required unless specified.
- The application will not be made publicly accessible or deployed to a live domain.

1.5. Assumptions and constraints

- Time: This is a semester-long project with a specific deadline.
- Resources: Development will be carried out by a single student or a small team.
- Evidence: All evidence is simulated. Files can be uploaded and hash values manually entered.
- Technology: The project will be built using a standard web stack (e.g., Python with Flask, a relational database like SQLite, and standard front-end technologies).

2. User roles and responsibilities

2.1. Administrator

- Responsibilities: - Manage all user accounts and assign roles. - Oversee all cases and evidence records within the system. - Monitor system security and audit logs.
- Permissions: - Full read/write access to all data. - Ability to add, edit, or delete user accounts and roles. - Access to system-level settings and logs.

2.2. Investigator

- Responsibilities: - Initiate new cases for academic or disciplinary issues. - Log new pieces of digital evidence, including descriptions and metadata. - Upload simulated evidence files and record their initial hash values. - Document the date, time, and location of evidence collection. - Request transfer of custody for evidence.
- Permissions: - Read/write access to cases they create. - Read-only access to the full history of evidence they have handled. - Cannot modify or delete evidence records after initial submission.

2.3. Forensic analyst

- Responsibilities: - Accept and process requests for evidence analysis. - Record and document their analysis steps and findings. - Update the evidence record with new hash values or notes after handling. - Transfer custody of evidence back to storage.
- Permissions: - Read/write access to evidence under their active custody. - Read-only access to all case details.

2.4. Evidence custodian

- Responsibilities: - Accept custody of evidence after collection. - Log the physical or digital evidence into storage. - Release evidence for analysis and accept its return.
- Permissions: - Read/write access to all evidence records for management purposes. - Ability to log and approve custody transfers.

2.5. Auditor

- Responsibilities: - Review case files and evidence records. - View the full, immutable audit trail for any piece of evidence.
- Permissions: - Read-only access to all data. - No ability to modify any records.

3. Security measures

Multi-factor authentication (MFA): Implement MFA to secure user login, preventing unauthorized access even if a password is compromised.
Password hashing: Store all user passwords using a strong, one-way hashing algorithm like bcrypt to prevent them from being read by unauthorized parties.
Role-based access control (RBAC): Ensure that access to specific data and functions is strictly limited to authorized user roles.
Input validation: Sanitize all user-supplied input to prevent web vulnerabilities like cross-site scripting (XSS) and SQL injection.
Cryptographic hashing: Use a hashing algorithm (e.g., SHA-256) to create and store a unique, unchangeable 'fingerprint' of each piece of digital evidence. Any change to the evidence file will produce a different hash, indicating tampering.

4. Simulated evidence examples

Academic: A student's assignment document, a plagiarism report, or an email from a teacher or student.
Disciplinary: A screenshot of a social media conversation, a video file from a school surveillance camera, or a signed incident report.
Other: Simulated system logs from a server, GPS coordinates from a mobile device (captured via the browser's Geolocation API), or a witness statement transcribed as a text file.