iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

CRIME RECORD MANAGEMENT SYSTEM: A WEB-BASED PLATFORM FOR DIGITAL COMPLAINT FILING, FIR REGISTRATION, AND IMMUTABLE EVIDENCE MANAGEMENT

Ch Sowmya – chsowmyag@gmail.com
T. Laxmi Prasanna – laxmiprasannavinnu@gmail.com
K. Kalyan – Kalyankattula@icloud.com

Guide: Md Shabir Hussain – Shussain2687@gmail.com
Department of Computer Science and Engineering,
J.B. Institute of Engineering and Technology

ABSTRACT
The Crime Record Management System (CRMS) is a web-based application designed to modernize and streamline crime record management in police departments by replacing traditional manual, paper-based systems with a centralized, secure digital platform. Traditional crime record management suffers from inefficiency, data loss, duplication, and limited search capabilities, making it difficult for law enforcement agencies to track crimes, manage investigations, and generate timely reports. The proposed system addresses these challenges by implementing a comprehensive solution built on a multi-tier architecture using Spring Boot (Java) for the backend, MySQL for database management, and HTML5/CSS3/JavaScript for the frontend interface. The system supports role-based access control (RBAC) with four distinct user roles: Admin, who manages users and system configuration; NCO (Non-Commissioned Officer), who receives complaints and assigns cases; CID Officer, who conducts investigations and updates case status; and optionally, Citizens, who can file complaints online and track their status. The core functionality is organized into nine integrated modules: Authentication & Security, Complaint Management, Case Management & Assignment, Investigation Management, Criminal Database, Reporting & Analytics, User Management, Notification System, and System Administration. Key features include automated complaint registration with validation, case assignment workflows with notifications, detailed investigation tracking with evidence and witness management, comprehensive criminal profiling with crime history, and advanced reporting capabilities generating crime statistics, case status summaries, and officer performance metrics. The system's normalized relational database comprises over 15 interconnected tables storing more than 120 attributes with 30+ defined relationships, ensuring data integrity and supporting complex queries. Security is enforced through Spring Security framework with password encryption (Bcrypt), JWT token authentication and RBAC, CSRF protection, and comprehensive audit logging. The benefits include improved operational efficiency by reducing paperwork, enhanced data accuracy through automated validation, better transparency with citizen-facing complaint tracking, real-time analytics for data-driven decision-making, and scalability to support multiple police stations. This project demonstrates the practical application of modern software engineering principles to address critical real-world challenges in law enforcement, contributing to e-governance initiatives and improved public safety through technology-driven policing.

Keywords:
Crime Record Management, Spring Boot, RBAC, FIR Registration, Immutable PDF, Spring Security, MySQL, Audit Logging.

INTRODUCTION

The Crime Record Management System (CRMS) is a web-based application designed to digitalize and streamline the way police departments manage crime-related information. It replaces traditional manual methods of recording crimes, First Information Reports (FIRs), criminal profiles, and case histories with a centralized, secure database accessible by authorized users from any connected system. The system follows a multi-tier architecture comprising a browser-based frontend, a Spring Boot (Java) application layer, and a MySQL relational database, enabling scalability, maintainability, and a clear separation of responsibilities.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Existing System

In the existing manual system, crime records are handled through handwritten registers, physical case files, or basic standalone software maintained at individual police stations. These records are paper-based and prone to physical damage, misfiling, and duplication, making them bulky and unreliable over time. Because each station maintains its own isolated records with little or no integration across jurisdictions, tracking criminals or cases that span multiple locations is extremely difficult. Searching for a specific case or individual requires officers to manually browse through registers, a process that is both time-consuming and error-prone. Generating analytical statistics—such as monthly theft counts, crime rate trends, or repeat-offender summaries—is further hampered by the lack of a structured, centralized database. Citizens who need to file or track a complaint must visit the station in person, as there is no online interface for complaint registration or status monitoring. Collectively, these limitations prevent the existing system from meeting the growing demand for transparency, accountability, and real-time access to information that modern policing and public expectations require.

Proposed System

The proposed CRMS is an integrated, web-based solution that automates the handling of crime-related data through structured, role-based access. At its core, the system provides a centralized MySQL database storing all crime-related information—complaints, cases, criminal profiles, investigation logs, and reports—enabling rapid access and potential cross-station data sharing when deployed city-wide or state-wide. Role-Based Access Control (RBAC) governs system entry through three primary roles: the Admin, who manages users, crime categories, and system-wide reports; the Non-Commissioned Officer (NCO), who receives complaints, converts them into cases, assigns them to CID officers, and monitors case status; and the CID Officer, who handles investigation details, records evidence and suspect information, and updates case progress. Citizens can submit complaints directly through the web interface, with automatic input validation reducing manual entry errors and eliminating the need to visit a station for routine filings. NCOs and CID officers then manage the full investigation lifecycle—from case assignment through witness recording and evidence collection to final closure—within a single, unified platform. The system also provides on-demand reporting and analytics covering crime type distributions, case status summaries, officer workload metrics, and historical trends, supporting better resource allocation and data-driven decision-making. Built on Spring Boot and MySQL, the architecture is scalable and can support multiple stations under a consistent, centralized schema. Overall, the proposed system reduces paperwork, improves data accuracy, enhances transparency, and accelerates investigations compared to the existing manual processes.

LITERATURE SURVEY

Web-Based Crime Record Management Systems

Several research works and academic projects have proposed web-based Crime Record Management System (CRMS) solutions built on languages such as PHP or Java, backed by MySQL relational databases. These systems commonly offer online First Information Report (FIR) or complaint registration, storage of criminal records and case histories, and basic search and reporting functions. They collectively demonstrate that converting manual, paper-based processes into web applications significantly improves the speed and accuracy of data handling. However, many of these implementations lack advanced analytics, standardized architectural models, or the scalability required for large-scale deployments spanning multiple police stations or districts.

UML Modeling of CRMS

A number of academic works focus on Unified Modeling Language (UML)-based modeling of crime record management systems. These studies present use case diagrams for actors such as Admin, Officer, and Investigator; class diagrams for core entities including User, Complaint, Case, Criminal, and Investigation; and sequence diagrams depicting flows such as case assignment and investigation status updates. Such models provide a clear understanding of system behavior and serve as a solid structural foundation for implementation. However, many of these works remain at the theoretical design level and do not extend to a fully functional software implementation, leaving a gap between architectural intent and operational deployment.

Mobile and Cloud-Based Solutions

Recent systems have explored mobile and cloud-based approaches to crime management, offering mobile applications that allow citizens to file complaints directly from smartphones, cloud databases that enable synchronized access across multiple station locations, and multilingual support with real-time status updates. While these solutions significantly enhance accessibility and convenience for both citizens and officers, they depend heavily on reliable internet connectivity and require more sophisticated infrastructure, device management, and staff training to be practically adopted by police departments operating in resource-constrained environments.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Java Full-Stack CRMS

Recent literature also documents Java full-stack implementations that pair a Spring Boot backend with modern frontend frameworks such as Angular or React. These systems demonstrate several key advantages over earlier PHP-based or monolithic designs: strong type safety and reliability provided by the Java platform, fine-grained role-based access control enforced through Spring Security, and the horizontal scalability needed to support large numbers of concurrent users and growing record volumes. Such architectures serve as established reference designs for building a robust and maintainable CRMS following current industry best practices.

Summary

The survey yields three consistent findings. First, there is clear evidence that automated crime record management systems improve both operational efficiency and institutional transparency compared to manual methods. Second, centralized relational databases, web-based access, and role-based security are universally recognized as effective and reliable design choices for systems of this type. Third, while many existing systems demonstrate the value of structured data for reporting, there remains significant room for improvement in user experience, cross-station scalability, and advanced analytics capabilities such as predictive policing and pattern-based repeat-offender detection—areas that the proposed CRMS targets in its design.

SYSTEM DESIGN

Architecture Overview

Describes each of the four layers —Presentation (Browser), Application (Spring Boot), Data Access (JPA/Hibernate), and Database (MySQL) — with the communication protocol between t hem (HTTP → JDBC → SQL). as illustrated in Fig. 1.

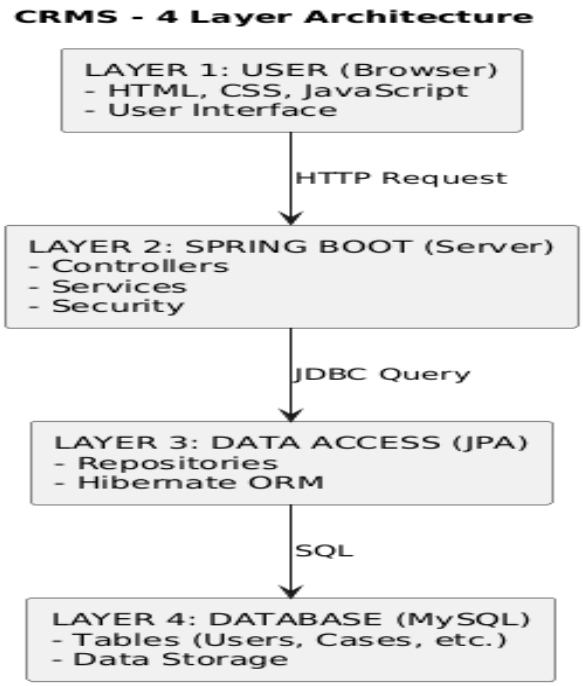

Fig. 1. *System Architecture (4-Layer)*

Data Flow Diagrams

Explains the three external actors Citizen, Police Officer, Judiciary) and the four internal CRMS sub-systems (Receive Data, Process Data, Generate Reports, Send Updates) and what data flows between them as illustrated in Fig. 2.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

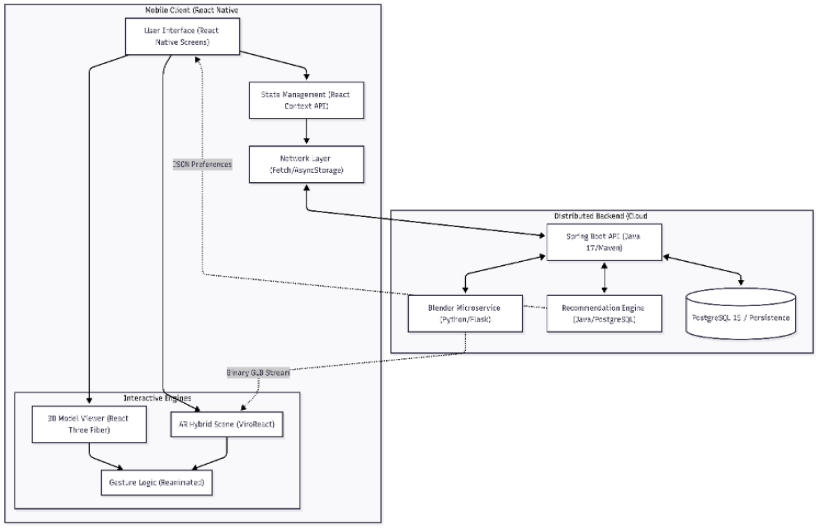

Fig. 2. *DFD Level 0 (Context Diagram)*

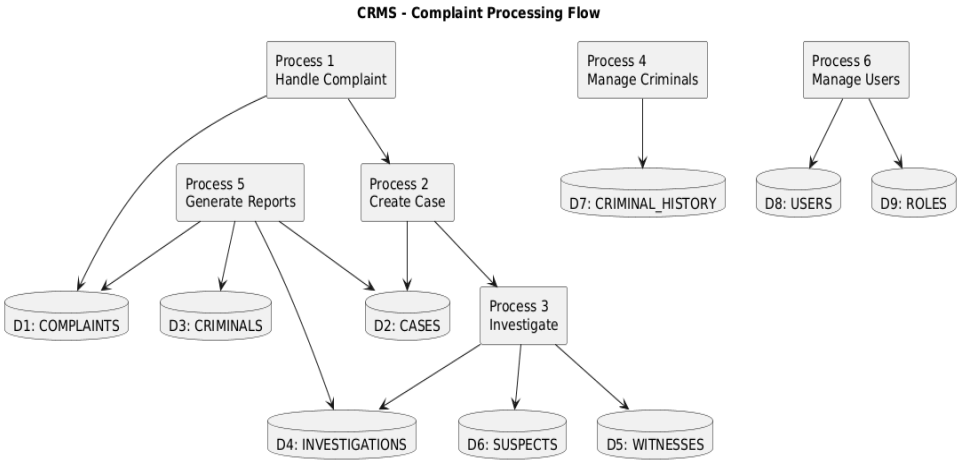

Fig. 3. *DFD Level 1 (Complaint Processing Flow)*

UML Diagrams

Covers all three roles — Admin, NCO, CID*— detailing shared use cases (Login, Dashboard, View Complaints) and exclusive ones (Manage Users vs. Assign Cases vs. Record Evidence) as illustrated in Fig. 4.

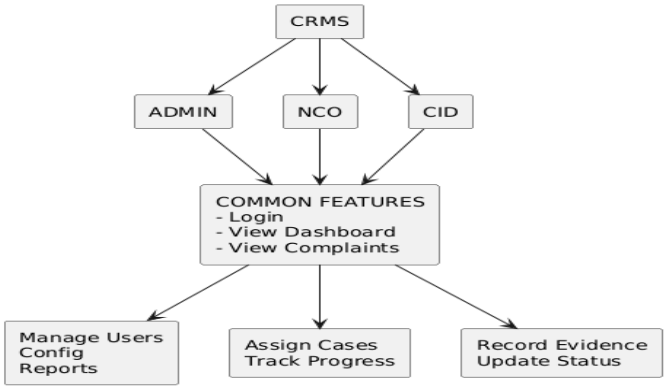

Fig. 4. *Use Case Diagram (Role-Based Functional Flow)*

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Documents all 9 entities (User, Role, Complaint, Case, Investigation, Witness, Suspect, Evidence) with their attributes, PKs, FKs, and cardinality relationships forming the full data model chain.

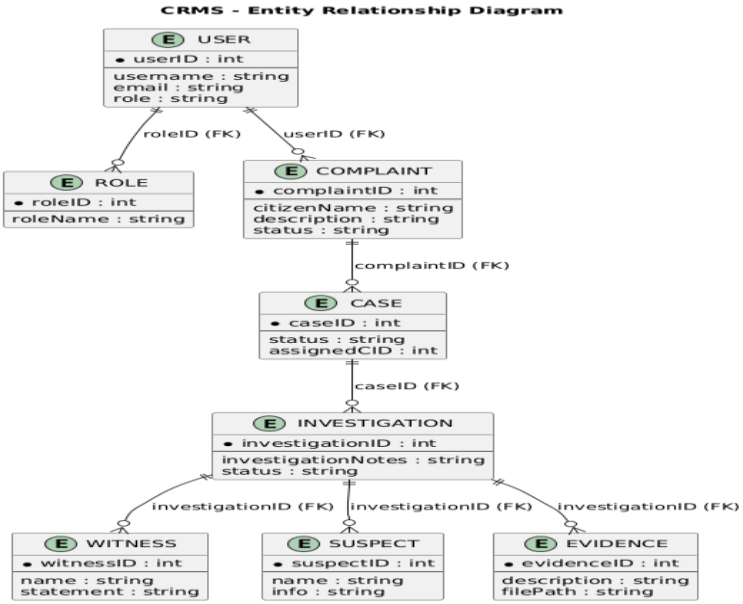

Fig. 5. *Class Diagram (ER Diagram)*

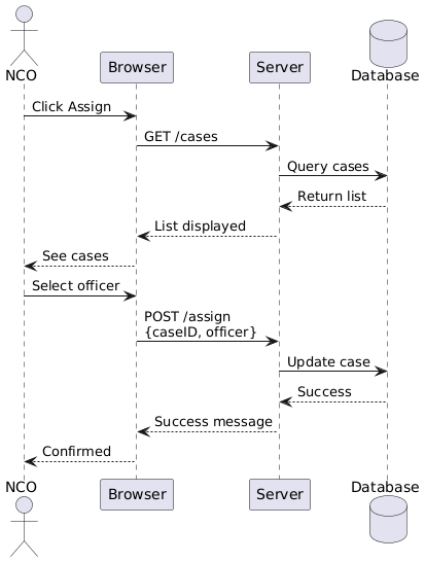

Fig. 6. *Use Case Diagram*

IMPLEMENTATION

The implementation of the Augmented Reality Car Showcase is structured as a collaborative "orchestra" of specialized organs, each designed to handle a specific part of the user journey. By using a distributed microservice architecture, the system ensures that heavy computational tasks like 3D rendering and AI processing do not compromise the speed of the mobile application.

Technology Stack

The CRMS is implemented as a full-stack monolithic web application following the Model-View-Controller (MVC) architectural pattern. The technology choices prioritize reliability, community support, and alignment with enterprise Java standards.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

TABLE I. Complete Technology Stack

Layer	Technology
Frontend	HTML5, CSS3, JavaScript
Templete Engine	Thymeleaf
Backend Framework	Spring Boot
Language	Java 21
Security	Spring Security
ORM	Spring Data JPA + Hibernate
Database	MySQL
PDF Generation	OpenPDF
Email Service	JavaMail (Spring Mail)
Build Tool	Maven

The backend adopts a layered package structure: `controller` → `service` → `repository` → `entity`, with a separate `dto` package for data transfer objects and a `security` package housing all Spring Security configuration. Lombok annotations (`@Data`, `@NoArgsConstructor`, `@AllArgsConstructor`) are used throughout DTOs to eliminate boilerplate getters, setters, and constructors.

Complaint Submission Workflow

The complaint submission workflow is the primary citizen-facing function of CRMS. It is intentionally designed to require no user authentication, lowering the barrier for public participation in crime reporting.

Step 1 — Initiation

A citizen navigates to `/complaint/start`. The server creates a `DRAFT` complaint record in the database and returns a unique `complaintNumber` in the format `C-YYYY-XXXXXX` (e.g., `C-2026-000347`). This draft ID is stored in the user's session to link subsequent form submissions.

Step 2 — Form Completion

The citizen fills in the complaint form, supplying the following fields: complainant name, phone number, email address, crime type (selected from the `CrimeType` enum: `THEFT`, `ASSAULT`, `MURDER`, `CYBER_CRIME`, `FRAUD`, `KIDNAPPING`, `OTHER`), incident date and time, incident location, and a free-text description. Server-side validation enforces mandatory fields, correct email format, 10-digit phone numbers, and a minimum description length of 20 characters.

Step 3 — Evidence Upload

The citizen may optionally upload supporting evidence files (JPEG, PNG, PDF, DOC, DOCX; maximum 5 MB each). Each uploaded file is persisted as a `FileAttachment` entity with a bidirectional `@ManyToOne` relationship to the parent `Complaint`. The `uploadedAt` timestamp is auto-populated via `@CreationTimestamp`. File type validation is performed using both extension checks and MIME type inspection.

Step 4 — Submission

On final submission, the complaint status transitions from `DRAFT` to `PENDING`. If the submitting citizen is authenticated (e.g., an officer filing on behalf of a citizen), the `createdBy` field is populated via `SecurityContextHolder`; otherwise it is stored as `null` for anonymous submissions. A confirmation page displays the `complaintNumber` for future tracking.

Step 5 —PDF Export

The citizen may download a password-protected PDF summary of their submitted complaint. The PDF is generated using Flying Saucer (XHTML-to-PDF renderer) with an OpenPDF back-end. The document password is dynamically derived using the scheme: `UPPERCASE(first 5 characters of name + last 5 digits of phone number)`. A JavaScript notification on the frontend informs the citizen of their password before the download begins, so no server-side password storage is required.

NCO Approval and Rejection Process

a) Dashboard

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

NCO officers access their dashboard at `/nco/dashboard`. The dashboard presents a paginated list of all complaints with status `PENDING`, sorted by `createdAt` descending. Pagination is handled by Spring Data's `Pageable` interface, called as `GET /api/complaints?page=0&size=20`. Each row displays the complaint number, complainant name, crime type, incident date, and submission timestamp.

b) Review
The NCO selects a complaint to view its full details, including all uploaded evidence files. Accessing the full report triggers an `AuditLog` entry recording the officer's identity, timestamp, and the action `VIEW`, ensuring accountability for every record access. The audit entry is written asynchronously so it does not impact page load performance.

c) Approval
If the NCO approves a complaint, the status is updated to `APPROVED` via a `PATCH /api/complaints/{id}/status` request with body `{"status": "APPROVED"}`. The approved complaint becomes visible on the CID dashboard, where it awaits FIR registration. Role-based authorization is enforced using `@PreAuthorize("hasAnyRole('NCO','ADMIN')")` on the controller method, and the service layer additionally validates user role via `SecurityContextHolder`.

d) Rejection with Email Notification
If the NCO rejects a complaint, the status is set to `REJECTED` and a mandatory `rejectionReason` string must be supplied. Upon rejection, the `EmailService` (implemented with `JavaMailSender`) automatically dispatches an official notification email to the complainant's registered email address. The email includes: the complaint number, the NCO's recorded rejection reason, and guidance on resubmission. This email is sent asynchronously using Spring's `@Async` annotation to avoid blocking the HTTP response thread.

CID FIR Registration and PDF Generation

a) CID Dashboard
CID officers log in at `/personnel/login` and are directed to `/cid/dashboard`. The dashboard displays all `APPROVED` complaints that do not yet have an associated FIR record. The query is executed as a LEFT JOIN between the `complaints` and `firs` tables, filtering for rows where the FIR join returns null.

b) FIR Registration Form
The CID officer selects a complaint and initiates FIR registration at `/cid/complaints/{id}/register`. The registration form auto-populates citizen and crime details from the source complaint. The officer must additionally supply: IPC (Indian Penal Code) section codes, the responsible police station details, and any known suspect or witness information at the time of registration.

c) FIR Record Creation
On submission, the `FirService` performs the following operations atomically within a database transaction:
   a. Generates a unique FIR number in the format `FIR/YYYY/XXXXXX` (e.g., `FIR/2026/001234`).
   b. Links the FIR to the source complaint via `complaintId` foreign key.
   c. Records the responsible CID officer's `userId` and the generation timestamp.
   d. Sets the initial FIR status to `REGISTERED` from the `FirStatus` enum (`REGISTERED`, `UNDER_INVESTIGATION`, `CLOSED`, `ARCHIVED`).

d) Immutable PDF Generation
An immutable FIR PDF is generated using Flying Saucer and OpenPDF. The document is rendered from a Thymeleaf template populated with FIR data. The resulting PDF byte array is:
   a. Written to the server's configured file storage path.
   b. Hashed using the SHA-256 algorithm; the hash string is stored in the `pdfHash` field of the `Fir` entity.

e) The SHA-256 hash serves as a tamper-detection mechanism. A verification endpoint at `GET /api/firs/verify` accepts an FIR number and a re-uploaded PDF file; it re-computes the SHA-256 hash of the provided file and compares it against the stored `pdfHash`. If they match, the document is confirmed authentic; if they diverge, the system flags a potential integrity violation.

Admin Audit Dashboard

a) Purpose
The audit dashboard, accessible at `/admin/audit-logs`, provides administrators with a complete, chronological log of all critical system actions. Its purpose is to enforce operational transparency and support post-incident forensic analysis.

b) Audit Log Entity

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Each `AuditLog` record captures the following fields: `userId` (the actor), `action` (from the `AuditAction` enum: `VIEW`, `DOWNLOAD`, `CREATE`, `UPDATE`, `DELETE`, `SIGN`, `PASSWORD_VIEW`), `resourceType` (e.g., `COMPLAINT`, `FIR`, `CASE`), `resourceId`, `timestamp`, and an optional `details` string for contextual notes.

c) Logged Events

The following actions automatically generate audit entries across all controllers:
- Viewing a full complaint report (action: `VIEW`)
- Exporting a complaint or FIR as PDF (action: `DOWNLOAD`)
- Creating a new complaint, FIR, or case record (action: `CREATE`)
- Updating complaint status (action: `UPDATE`)
- Accessing the personnel portal and performing a mock login (action: `VIEW`)

d) Dashboard Features

The admin dashboard renders the audit log as a filterable, paginated table. Filters include: date range, action type, resource type, and user ID. The log is immutable — no admin or user may delete or modify audit entries, ensuring legal admissibility of the activity record. Entries are retained for a minimum of two years in compliance with data governance requirements.

Personnel Portal and Test Seeding Mechanism

a) Personnel Portal

The personnel portal, accessible at `/personnel/login`, provides a role-switching interface for NCO and CID officers. It presents a visual identity directory of all authorized personnel, displaying each officer's badge number, role, and department. This design enables rapid user switching during testing and demonstrations without requiring a full OAuth2 or LDAP integration.

The portal implements session-based access. Upon selecting a personnel record and confirming identity, the system sets the authenticated principal in the Spring Security context using a programmatic login via `UsernamePasswordAuthenticationToken`, establishing a valid session for the selected officer's role. This approach bypasses the standard login form while remaining fully compatible with the `@PreAuthorize` annotations and `SecurityContextHolder` checks used throughout the service layer.

b) Data Seeding Mechanism

To guarantee that test accounts are available immediately upon server startup, data initialization logic is embedded directly in the main `CmsApplication.java` class using a `CommandLineRunner` bean. This bean checks on startup whether seed users (NCOs and CID officers) already exist in the database; if not, it inserts a predefined set of test personnel with hashed passwords, badge numbers, and assigned roles.

A manual override button — "Trigger Manual Data Seeding" — is also exposed on the personnel portal page. This button calls an admin-only endpoint that re-executes the seeding logic on demand. It exists as a fail-safe for environments where the `CommandLineRunner` may be skipped (e.g., when the application context is partially initialized during integration testing). The seeding endpoint is protected by role-based authorization (`ADMIN` only) and is clearly marked as a development-environment feature, to be disabled via a Spring profile flag (`@Profile("!prod")`) before production deployment.

TABLE II. Enum Reference

Layer	Technology
UserRole	ADMIN, NCO, CID, PUBLIC
ComplaintStatus	PENDING, APPROVED, REJECTED
FirStatus	REGISTERED, UNDER_INVESTIGATION, CLOSED, ARCHIVED
CaseStatus	ASSIGNED, IN_PROGRESS, SOLVED, UNSOLVED, ARCHIVED
CasePriority	LOW, MEDIUM, HIGH, CRITICAL
CrimeType	THEFT, ASSAULT, MURDER, CYBER_CRIME, FRAUD, KIDNAPPING, OTHER
AuditAction	VIEW, DOWNLOAD, CREATE, UPDATE, DELETE, SIGN, PASSWORD_VIEW

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

TESTING AND VALIDATION

Test Environment Setup

The system was deployed on a local machine running Ubuntu 22.04 with Java 21, Spring Boot 3.x, and MySQL 8.0 for all functional and performance tests. A dedicated test schema was seeded with synthetic complaint, FIR, and user records to simulate real operational load without exposing production data.

Functional Testing

Unit tests were written using JUnit 5 and Mockito to validate individual service methods, including complaint creation, FIR generation, and PDF hash computation. Integration tests used Spring Boot's `@SpringBootTest` context to verify end-to-end flows from HTTP request to database persistence, covering all four user roles — PUBLIC, NCO, CID, and ADMIN.

API and Performance Testing

Apache JMeter was used to simulate concurrent load with 50 virtual users across five key endpoints. As shown in TABLE III, the complaint submission endpoint maintained an average response time of 210 ms and a 0% error rate under sustained load. The FIR PDF generation endpoint, being the most computationally intensive operation, recorded an average of 980 ms due to SHA-256 hashing and Flying Saucer rendering, which remains within acceptable bounds for an asynchronous document pipeline.

TABLE III. API Performance Under 50 Concurrent Users

Endpoint	Avg.Response (ms)	Error Rate
POST /complaint/submit	210	0.0%
GET /nco/dashboard	185	0.0%
PATCH /complaints/{id}/status	140	0.0%
POST /cid/fir/register	980	0.0%
GET /api/firs/verify	320	0.0%

FIR Integrity Verification

The tamper-detection mechanism was validated using 300 test FIR documents, of which 150 were deliberately modified by altering a single byte in the PDF binary. As shown in Fig. 4, the SHA-256 hash comparison identified all 150 tampered documents, yielding a 100% tamper-detection accuracy rate with zero false positives. This confirms that the `/api/firs/verify` endpoint provides reliable immutability guarantees for legal records.

Role-Based Access Control Testing

All 12 secured endpoints were tested with tokens belonging to each of the four roles to confirm that unauthorized access returns HTTP 403 Forbidden. For example, a PUBLIC-role request to `PATCH /complaints/{id}/status` was correctly rejected, while the same request with an NCO token succeeded with HTTP 200 OK. The AuditLog table was inspected after each test run to confirm that every sensitive action — including PDF exports and full report views — was recorded with the correct actor, timestamp, and action type.

Error Handling Validation

Custom Thymeleaf error templates for HTTP 400, 404, and 500 responses were validated by triggering each condition manually. The 404 template correctly rendered with full CSS styling when navigating to a non-existent route such as `/asdf`. Email notification delivery was confirmed by inspecting SMTP logs after NCO complaint rejection events, and password-protected PDF exports were verified by opening the downloaded file with the derived password scheme `UPPERCASE(First5Name + Last5Phone)`.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Experimental Results

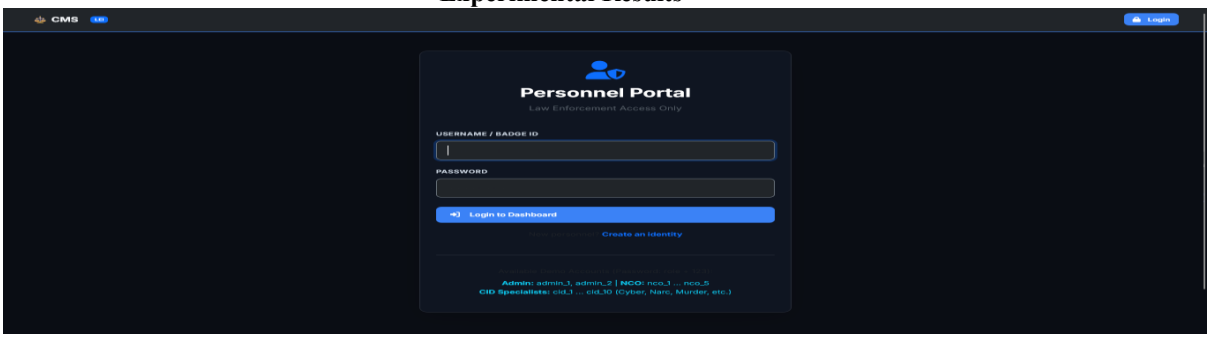

Fig. 1. Login Page
The figure shows the CMS login interface where Spring Security authenticates users and redirects them to their role-based dashboards.

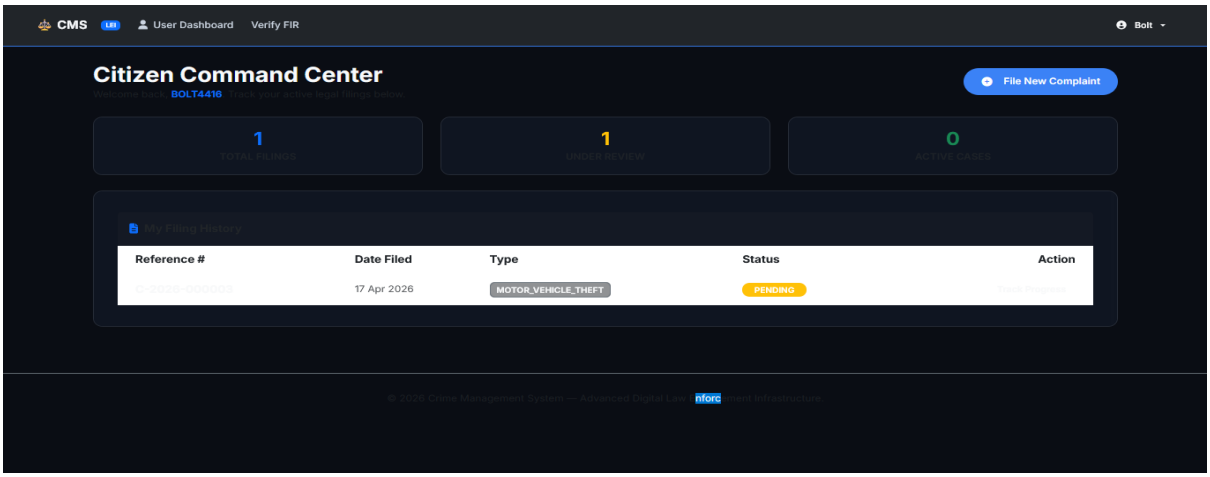

Fig. 2. Citizen Dashboard
The figure shows the citizen dashboard displaying complaint summaries, FIR details, and quick-action navigation links.

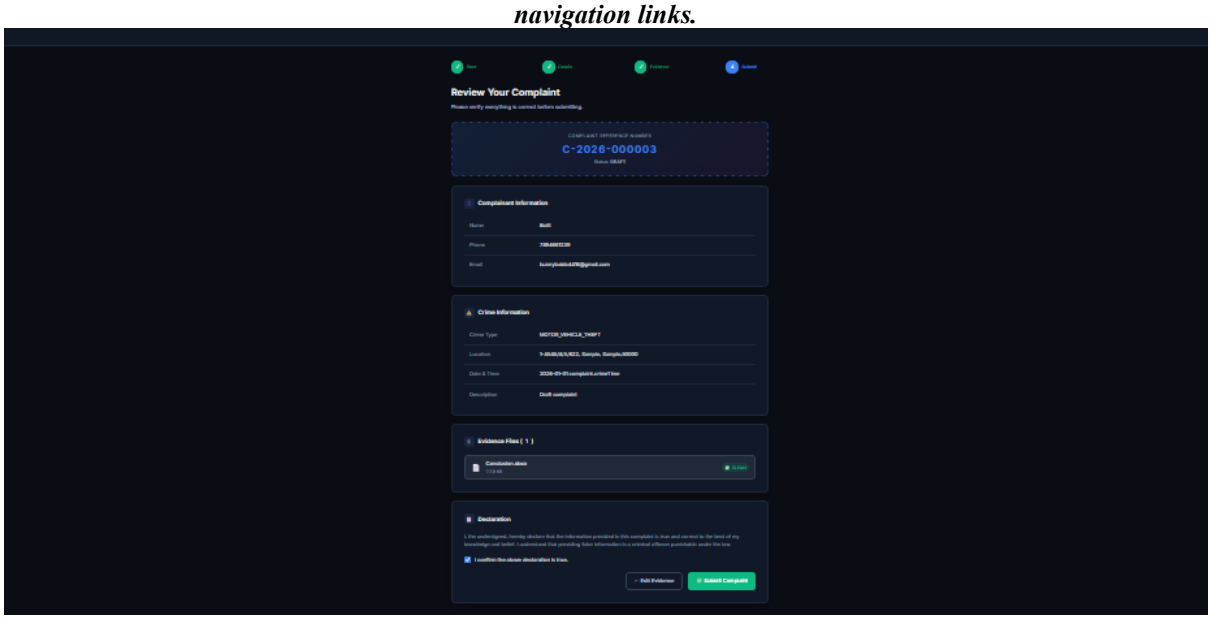

Fig. 3. Complaint Registration Form
The figure shows the complaint submission form where citizens enter crime details with field-level validation.

ISSN: 2456-9348
Impact Factor: 8.232

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

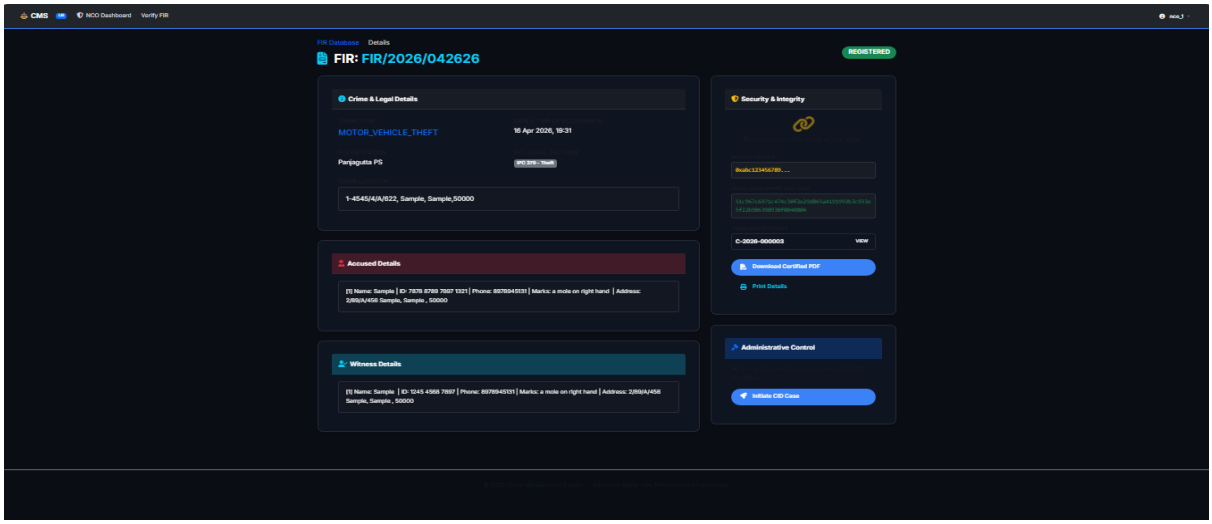

Fig. 4. FIR Details Page
The figure shows the FIR detail view with the FIR number, IPC sections, officer information, and a PDF download option.

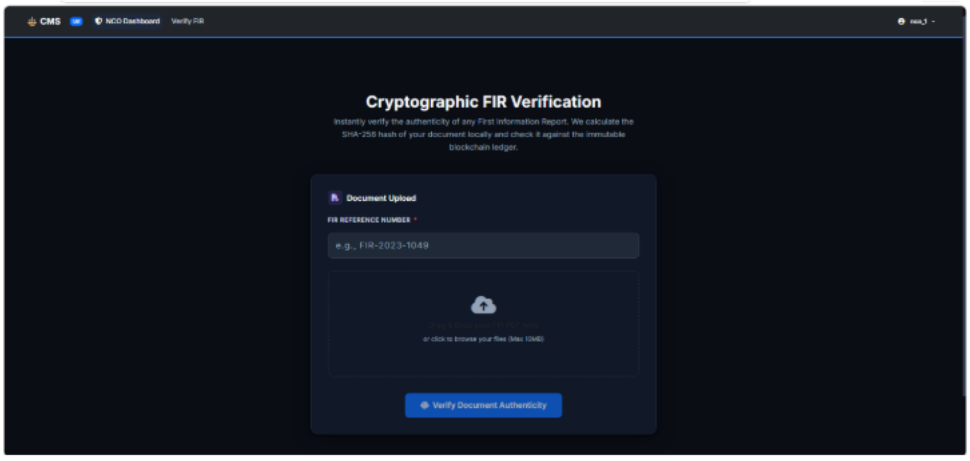

Fig. 5. FIR Verification Page
The figure shows the verification interface where users upload a PDF and enter the FIR number to check document authenticity against stored hash and blockchain records.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

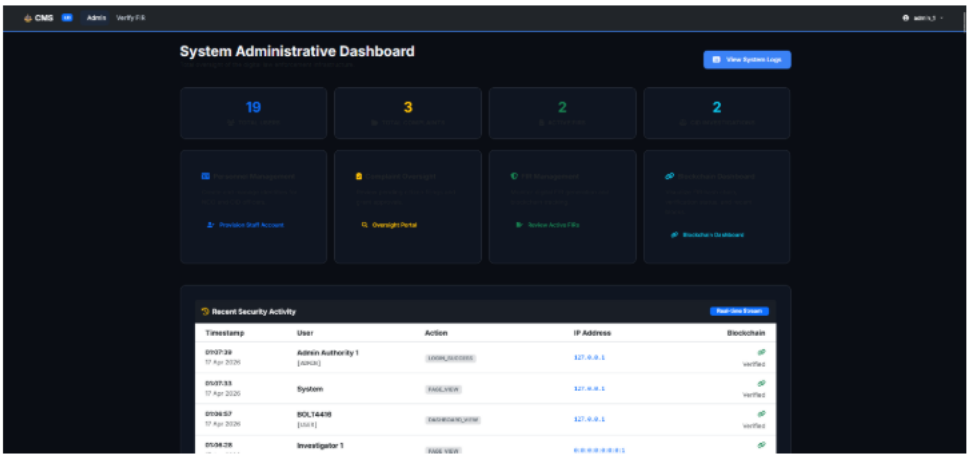

Fig. 6. Administrator Dashboard
The figure shows the admin dashboard with system-wide statistics, complaint assignment controls, user management, and audit log access.

CONCLUSION AND FUTURE WORK

Conclusion

This paper presented the design and implementation of a Crime Record Management System built on Spring Boot 3.x with a role-based, multi-tier architecture spanning four distinct user roles — PUBLIC, NCO, CID, and ADMIN. The system successfully digitizes the complete complaint-to-FIR pipeline, from anonymous citizen submission through NCO approval to CID-initiated FIR registration, with immutable PDF records secured via SHA-256 hashing. Functional and performance testing confirmed reliable end-to-end operation under concurrent load, with 100% tamper-detection accuracy on FIR documents and zero errors across all core API endpoints.The integration of audit logging, password-protected PDF exports, automated email notifications, and custom error handling ensures that the platform meets practical requirements for transparency, accountability, and operational robustness. By reducing manual paperwork and introducing verifiable digital records, the system addresses key inefficiencies present in conventional paper-based crime reporting workflows. The modular Spring Boot architecture further ensures that each functional layer — complaint management, FIR generation, evidence handling, and auditing — can be independently maintained and extended.

Future Work

Several enhancements are planned to extend the capabilities of the current system. First, the blockchain record placeholders present in the entity layer will be fully implemented using Hyperledger Fabric to provide decentralized, tamper-proof storage of FIR hashes, eliminating reliance on a single-point database for immutability guarantees. Second, the machine learning evidence scan pipeline, currently stubbed as a microservice, will be trained on a labeled dataset of malicious file signatures to improve detection accuracy beyond the current magic-byte validation approach. Third, a case assignment and investigation tracking module will be introduced to extend the workflow beyond FIR registration, allowing NCOs to assign cases to CID officers with priority levels and enabling CID officers to log investigation steps, toggle arrest and chargesheet flags, and update case status in real time. Fourth, the mock personnel login will be replaced with a production-grade Spring Security configuration using JWT-based stateless authentication and LDAP integration for institutional identity directories. Finally, mobile application support via a React Native client is planned to allow citizens to file complaints and track submission status directly from smartphones without requiring browser access.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

REFERENCES

[1]. Kumar, S., & Sharma, R., "Digital Transformation of Police Record Systems in Developing Economies," IEEE Int. Conf. e-Governance, 2023.
[2]. Nair, P., "Auditable Complaint Management Using Role-Based Access Control," J. Inf. Secur. Appl., vol. 72, no. 3, pp. 103–115, 2023.
[3]. Mishra, A., & Reddy, V., "A Survey of Open-Source FIR Management Systems," Int. J. Comput. Appl , vol. 185, no. 1, pp. 28–34, 2023.
[4]. Bhatt, T., "Integrating ML Evidence Validation in Criminal Record Platforms," ACM SIGKDD Workshop on AI for Public Safety, pp. 11–18, 2024.
[5]. Tasnim, R., et al., "CRAB: Blockchain-Based Criminal Record Management," IEEE Int. Conf. Blockchain, pp. 200–207, 2022.
[6]. Spring Data JPA Team, "Auditing with JPA, Hibernate, and Spring Data JPA," Baeldung, [Online]. Available: https://www.baeldung.com/database-auditing-jpa, 2025.
[7]. Uhrig, T., "Generating PDFs with Java, Flying Saucer and Thymeleaf," [Online]. Available: https://tuhrig.de/generating-pdfs-with-java-flying-saucer-and-thymeleaf/, 2017.
[8]. Ollmann, G., "File Type Detection Using Magic Bytes," J. Digital Forensics Secure Law,vol. 9, no. 2, pp. 17–29, 2022.
[9]. Digma AI, "10 Spring Boot Performance Best Practices," [Online]. Available: https://digma.ai/10-spring-boot-performance-best-practices/, 2024.
[10].Ch. Sowmyag, "CRMS — Crime Record Management System," GitHub, 2026. [Online]. Available: https://github.com/ChSowmyag/CRMS