Multi-Domain Intelligence Platform

CST1510 Coursework 2 – Technical Report

Student Details:
Name: **Mohamedkazim khaku**
Student ID: **M01093420**

GitHub Repository: https://github.com/kaz1mk/CW2_CST1510_M01093420

Contents

1. Introduction and Project Scope

Modern organisations increasingly rely on multiple operational domains such as cybersecurity operations, data governance, and IT service management. Traditionally, these domains are managed using separate tools and dashboards, resulting in fragmented visibility, duplicated effort, and delayed decision-making. The purpose of the Multi-Domain Intelligence Platform is to address this challenge by consolidating operational data from all three domains into a single, integrated system that supports monitoring, analysis, and informed action.

The system is implemented as a Streamlit-based web application that provides authenticated access to multiple operational dashboards using Python 3 [1, 2]. It enables users to record, manage, and analyse cybersecurity incidents, organisational data assets, and IT service requests within a unified environment. By centralising this information, the platform supports cross-domain awareness, allowing patterns and operational risks to be identified more effectively than with isolated systems.

This project represents a Tier-3 solution. It goes beyond basic data display by incorporating persistent storage using SQLite [3], secure authentication with hashed passwords using bcrypt [4], object-oriented service layers, structured database schemas, and an AI operations assistant integrated via OpenRouter [5]. The inclusion of role-aware authentication, multi-table relational data, and analytical visualisations demonstrates a level of complexity appropriate for advanced undergraduate coursework and realistic enterprise-style systems.

Figure 1: Login page after successful authentication

2. System Architecture and Implementation

2.1 Security and Database

SQLite was selected as the database technology due to its simplicity, portability, and suitability for small-to-medium scale applications [3]. It requires no external server, making it appropriate for coursework deployment while still supporting relational data structures and parameterised SQL queries.

The database schema includes tables for user accounts, cybersecurity incidents, data asset metadata, and IT service tickets. The accounts table stores usernames, password hashes, and user roles. The cybersecurity incidents table records incident type, severity, status, and reporting information. The datasets metadata table tracks governance attributes such as source, record count, and file size. The IT tickets table stores service request details, including priority, status, and assignment.

Security is a central design concern. User passwords are never stored in plaintext. Instead, the system uses bcrypt to hash passwords before storage [4]. During login, submitted passwords are verified against stored hashes, which reduces the risk of credential disclosure if the database is compromised.

The users.txt file is used for initial persistence and migration during early development. During system bootstrapping, these users are migrated into the SQLite database, after which the database becomes the authoritative source. This supports incremental development while maintaining secure password handling.

2.2 System Structure (MVC and Data Flow)

The platform follows a layered architecture that separates responsibilities across distinct components. The four primary layers are the user interface layer, the logic layer, the core services layer, and the data layer. This design improves modularity and maintainability.

The user interface layer is implemented using Streamlit and is responsible for rendering dashboards, forms, and visual analytics [2]. User input collected through the interface is passed to the logic layer rather than interacting directly with the database.

The logic layer contains domain-specific modules that implement business rules and validation. For example, when a user submits a new cybersecurity incident, the logic layer validates the input before passing it to the core data services.

The core layer provides foundational services shared across the system. These include database connectivity, authentication operations, and bootstrapping workflows. The data layer consists of an SQLite database and seeded CSV files.

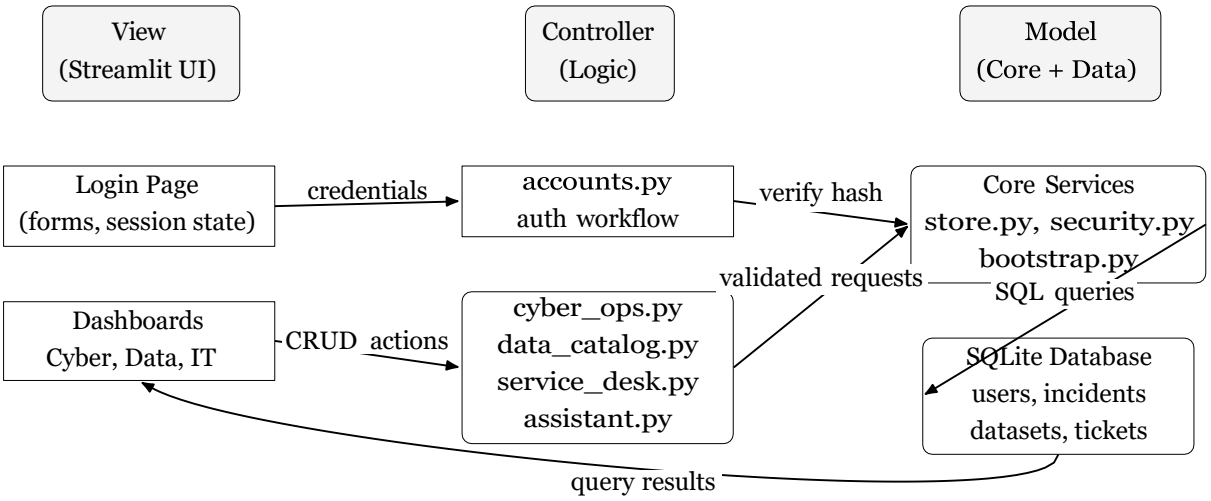

Figure 2: MVC-style data flow for authentication and dashboard CRUD operations.

2.3 Code Organisation (OOP Refactoring)

The system adopts an object-oriented design to structure domain logic consistently. Each operational domain is represented by a service class, including CyberOps, DataCatalog, and ServiceDesk. These classes encapsulate domain-specific operations such as creating incidents, registering datasets, or managing IT tickets.

The Store class acts as a database abstraction layer. It manages connections to SQLite and provides reusable methods for executing queries. Domain service classes depend on the Store rather than interacting directly with the database, reducing duplication and enforcing consistent access.

The Assistant component interacts with all three service classes, reflecting its cross-domain analytical role rather than a standalone domain.

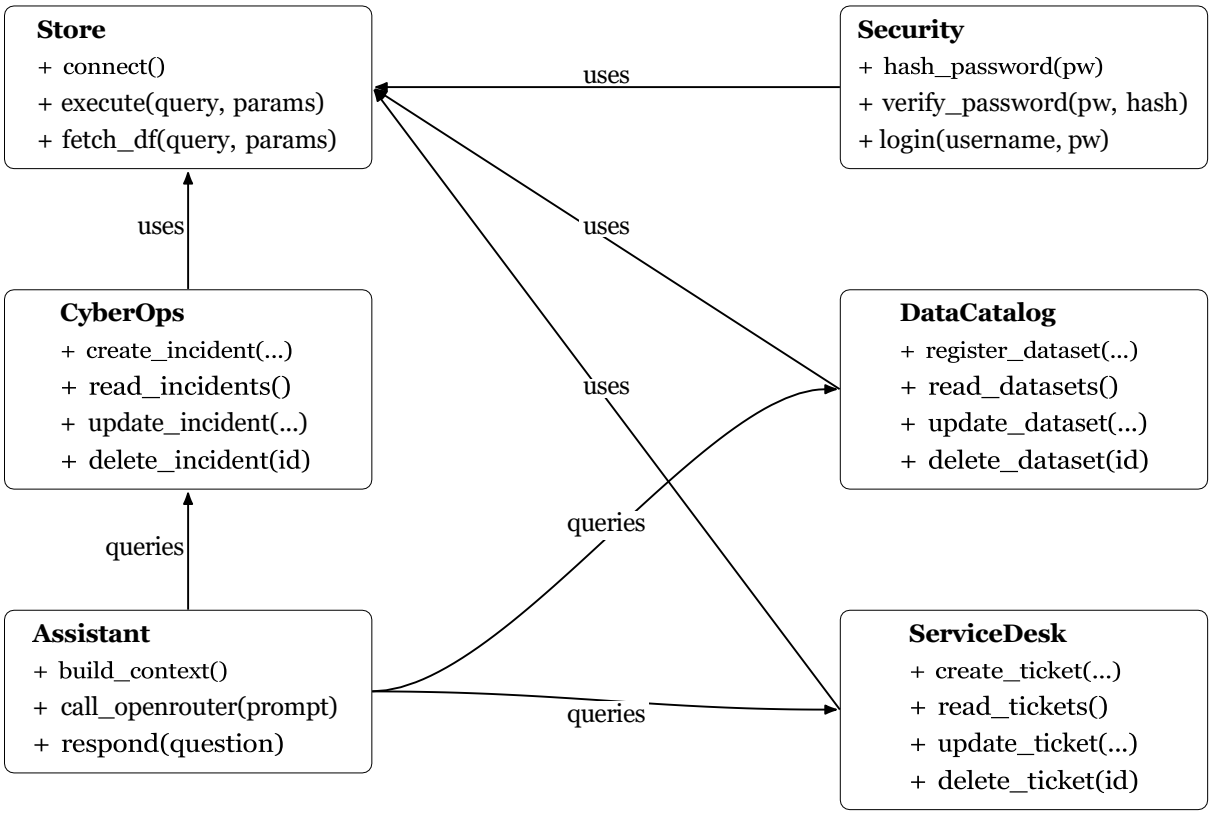

Figure 3: UML class diagram showing shared database access via Store and cross-domain Assistant interaction.

2.4 Key Features (Dashboards, Visualisations, AI)

The cybersecurity dashboard allows users to create, view, update, and delete incident records. It provides visual summaries of incidents by severity and status, supporting rapid assessment of operational risk.

The data governance dashboard focuses on organisational datasets. It displays metadata such as dataset size, source, and record count, enabling oversight of data assets.

The IT service desk dashboard manages service requests. Users can submit tickets, track their status, and review backlog levels. Visualisations highlight unresolved or high-priority tickets, supporting workload management and service quality monitoring.

The AI operations assistant provides natural language analysis across domains. Context from incidents, datasets, and IT tickets is combined and sent to Gemini for inference [5]. API keys are handled through environment variables to reduce the risk of accidental credential exposure [5].

Figure 4: Cybersecurity dashboard: incidents table and severity breakdown chart

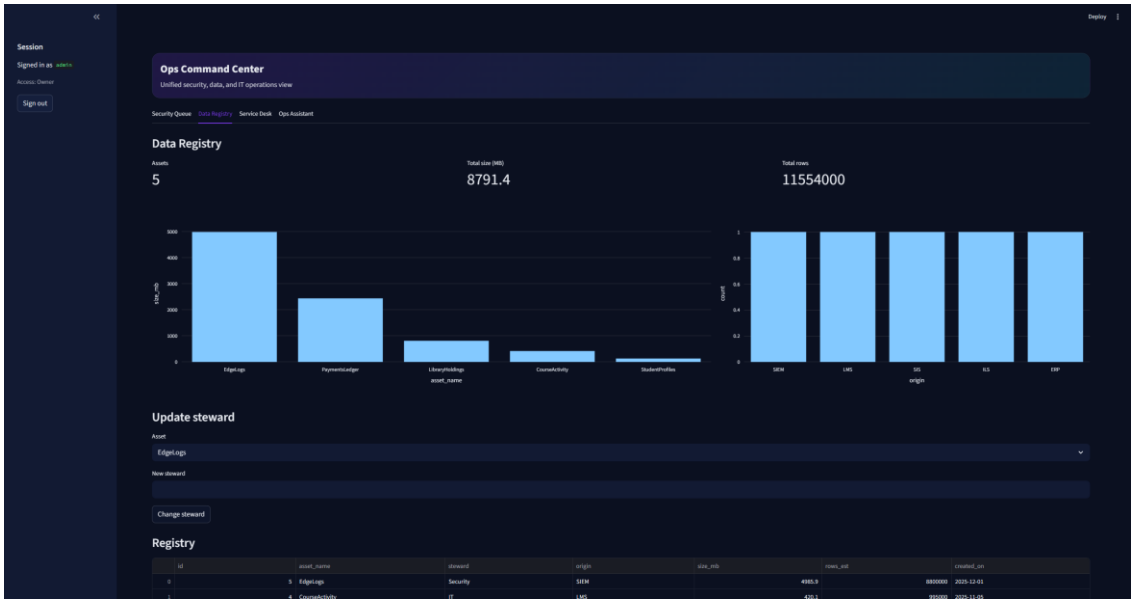

Figure 5: Data governance dashboard: dataset catalog and size/records visualisations

Figure 6: IT service desk dashboard: ticket backlog and priority/status charts

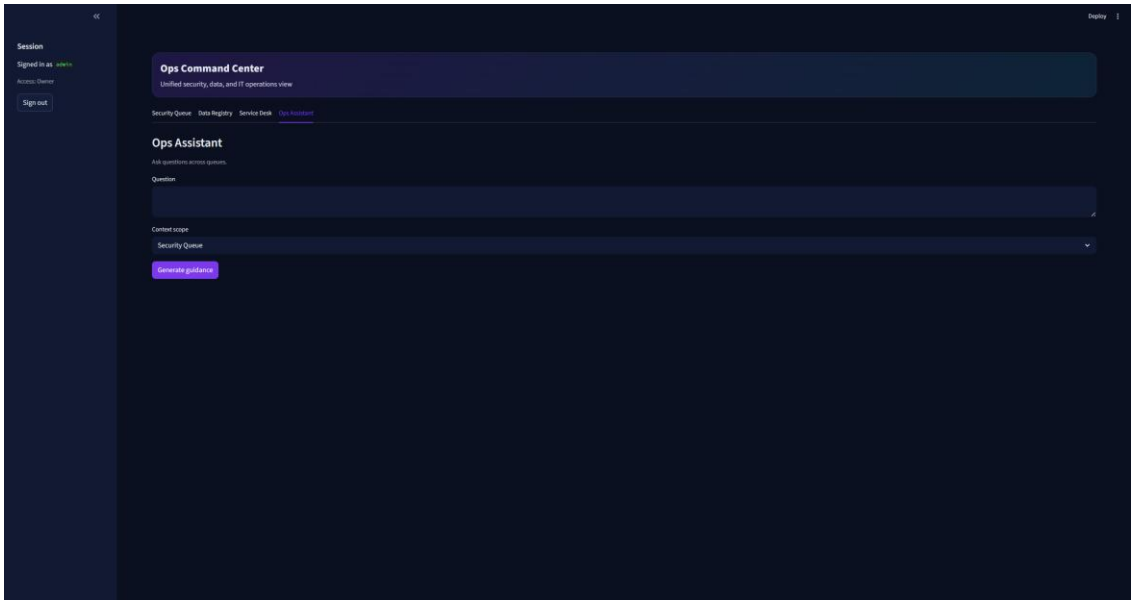

Figure 7: AI assistant panel showing a prompt and response based on operational context

3. High-Value Analysis and Insights

The integrated dashboards enable meaningful operational insights. Cybersecurity visualisations reveal trends such as an increase in high-severity incidents or prolonged resolution times. Data governance metrics highlight large or outdated datasets that may require review. IT service analytics expose ticket backlogs or recurring issue categories.

Visualisation supports decision-making by transforming raw records into interpretable patterns. A spike in unresolved incidents combined with a growing IT ticket backlog can indicate resource constraints and operational bottlenecks.

4. Reflection and Conclusion

This project developed practical skills in full-stack Python development, database design, secure authentication, and system architecture. Implementing bcrypt-based security and layered architecture reinforced the importance of security and maintainability in real-world systems.

Key challenges included coordinating data flow across multiple domains and ensuring consistent database access. These were addressed through the Store abstraction and separation of concerns.

The system meets Tier-3 criteria by integrating persistent storage, authentication, object-oriented design, analytical dashboards, and AI-assisted analysis. Future improvements could include role-based access control, advanced analytics, and deployment on a cloud-based database.

In conclusion, the Multi-Domain Intelligence Platform demonstrates how integrated operational intelligence can improve visibility and decision-making across cybersecurity, data governance, and IT service management.

5. References

[1] Python Software Foundation. *Python 3 Documentation*. https://docs.python.org/3/.

[2] Streamlit Inc. *Streamlit Documentation*. https://docs.streamlit.io/.

[3] SQLite Consortium. *SQLite Documentation*. https://www.sqlite.org/docs.html.

[4] Niels Provos and David Mazières. *A Future-Adaptable Password Scheme*. Proceedings of the USENIX Annual Technical Conference, 1999

[5] Google. (2024). *Google AI Python SDK: Python client for Google's generative AI models*. https://ai.google.dev/

A. Appendix (Optional): Running Instructions

- Create and activate a virtual environment.
- Install dependencies: `pip install -r requirements.txt`
- Run Streamlit: `streamlit run command_center.py`
- Ensure environment variables (API key) are set in `.env` or system environment.