Middlesex University Mauritius

School of Science and Technology

CST1510 Coursework 2

Multi-Domain Intelligence Platform

Technical Report

Name:	Mustafa Qasim
Student ID:	M01003450
Programme:	BSc Computer Science
GitHub Repository:	________________________________

Contents

1 Introduction

The Multi-Domain Intelligence Platform is a unified, data-driven application designed to support three core operational functions within an organisation: Cybersecurity, Data Science, and IT Operations. Its purpose is to provide teams with an integrated system for monitoring incidents, analysing datasets, tracking IT tickets, and generating insights from real-time data.

The platform is implemented as a multi-page Streamlit web application [1]. After securely logging in, a user can navigate to any dashboard depending on their role. Each dashboard includes interactive tables, filtering features, charts, and full CRUD functionality. The Cybersecurity dashboard additionally contains an AI-powered assistant capable of analysing filtered incident data and producing contextual recommendations using an OpenAI-compatible chat completion API when configured [6].

This system satisfies **Tier 3** of the coursework requirements by implementing all three domain dashboards. Each domain includes CRUD support, data visualisations, metrics, and domain-specific insights. The platform also integrates authentication, database persistence, and a modular architecture that separates presentation, business logic, and data access.

The project was developed iteratively from Week 7 to Week 11, starting from authentication and database schema design, then adding CRUD and dashboards, and finally refactoring into a service layer and integrating the AI assistant.

2 How I Built the System

2.1 Security and Database

User authentication was the first component implemented. Passwords are never stored in plain text; instead, the `bcrypt` hashing library was used to generate salted password hashes [3]. These were initially written to a text file and later migrated into the SQLite `users` table. During login, the entered password is hashed again and compared with the stored hash, ensuring that the actual password is never exposed.

SQLite was chosen as the relational database management system because it is lightweight, file-based, and easy to bundle with a Streamlit app [2]. Four key tables support the platform:

- **users** – secure authentication records.
- **cyber_incidents** – cybersecurity incident logs.
- **datasets_metadata** – dataset catalogue information.
- **it_tickets** – IT support desk tickets.

All SQL queries are parameterised to reduce the risk of injection attacks. A Python loader script ingests CSV seed data, validates column names, and inserts rows into the correct tables.

2.2 System Structure

The platform follows a three-layer architecture that separates presentation, business logic, and data access. The system flow is:

1. The user logs in using the Streamlit login page.

2. Login credentials are verified through `UserService`; on success, the username is stored in `st.session_state`.

3. The main navigation page displays links to the Cybersecurity, Data, and IT dashboards.

4. Dashboards interact with domain-specific service classes to request, insert, update, or delete records.

5. The service layer communicates with the database using a shared `DatabaseManager` and CRUD utilities.

6. Streamlit renders tables, metrics, and charts based on the returned data [4, 5].

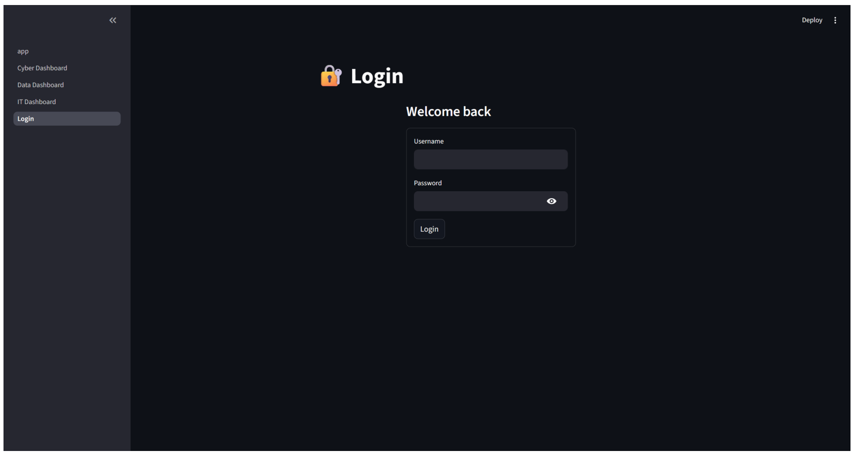
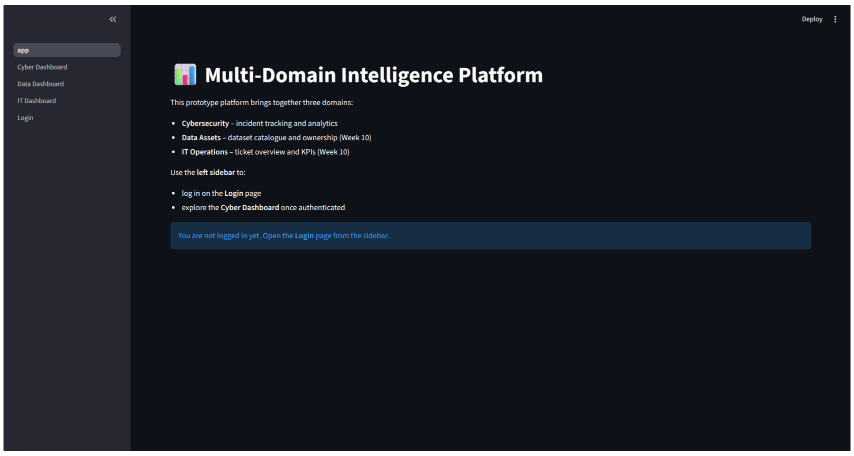


Figure 1: Login page used to authenticate users before accessing any dashboard.



Multi-Domain Intelligence Platform

This prototype platform brings together three domains:

- **Cybersecurity** – incident tracking and analytics
- **Data Assets** – dataset catalogue and ownership (Week 10)
- **IT Operations** – ticket overview and KPIs (Week 10)

Use the **left sidebar** to:

- log in on the **Login** page
- explore the **Cyber Dashboard** once authenticated

You are not logged in yet. Open the **Login** page from the sidebar.

Figure 2: Main navigation page showing links to the three domain dashboards.

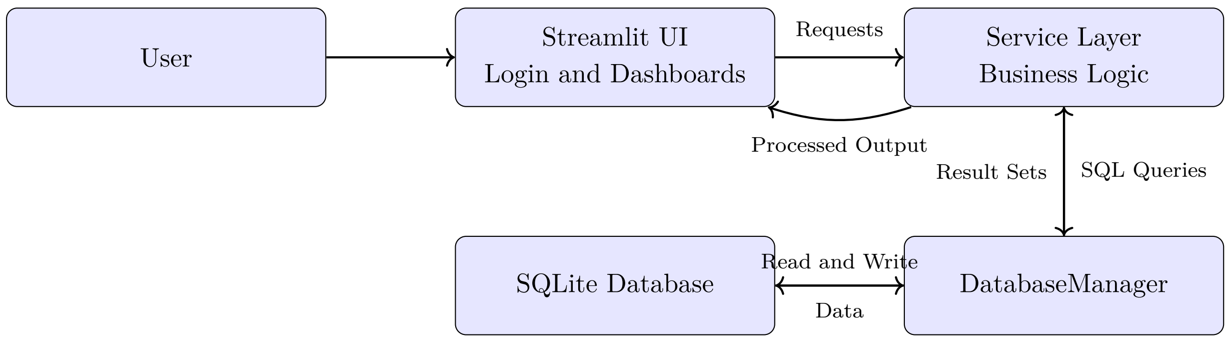


Figure 3: Data flow from user input through Streamlit, services, and SQLite.

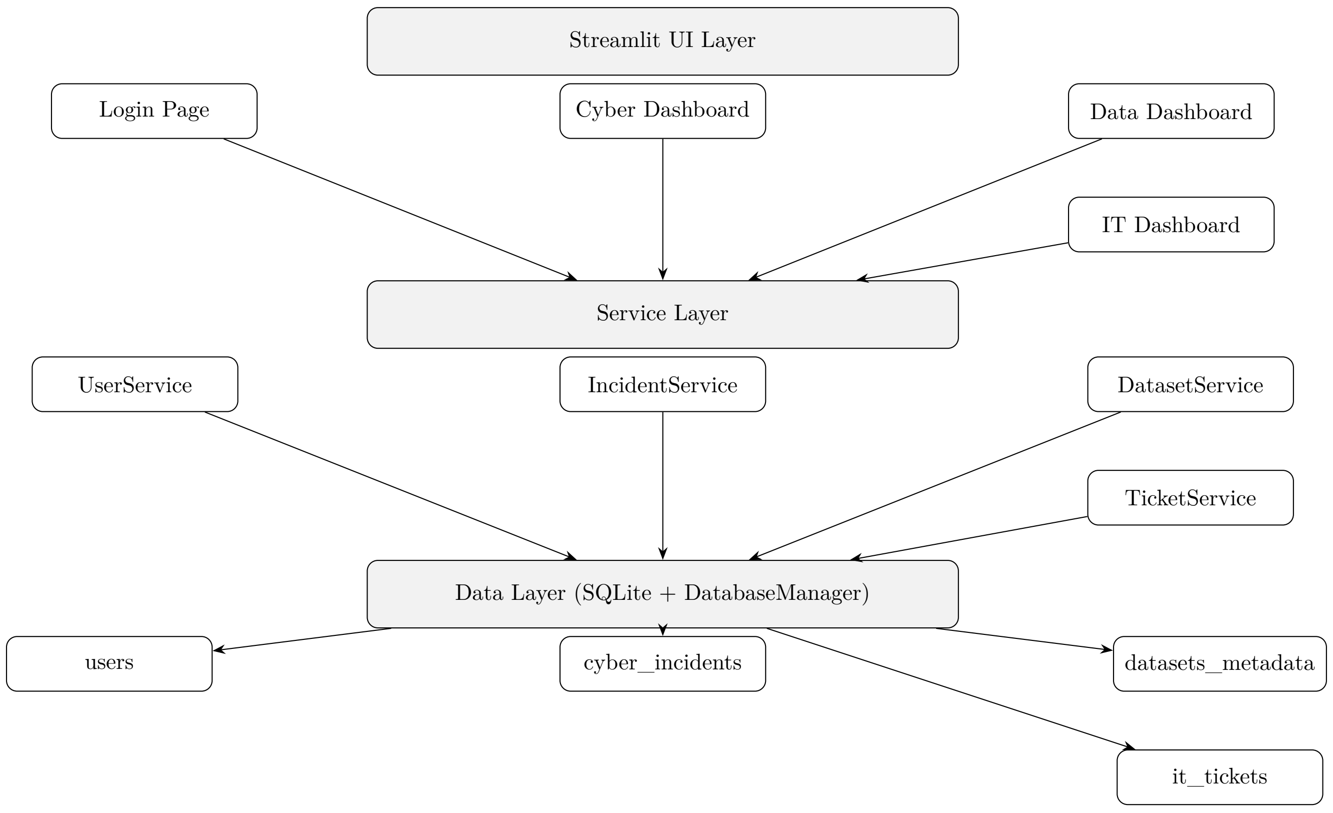


Figure 4: Three-layer architecture linking Streamlit UI, service layer, and SQLite tables.

2.3 Code Organisation (OOP Refactoring)

During Week 11, the application was refactored into a clearer object-oriented structure. The Streamlit pages now call service classes instead of issuing SQL directly. The main service classes are:

- **UserService** – authentication and user lookup.
- **IncidentService** – CRUD operations for `cyber_incidents`.
- **DatasetService** – CRUD operations for `datasets_metadata`.

- **TicketService** – CRUD operations for `it_tickets`.

All of these depend on a shared `DatabaseManager` that encapsulates connection handling and provides a single point of access to the SQLite database.

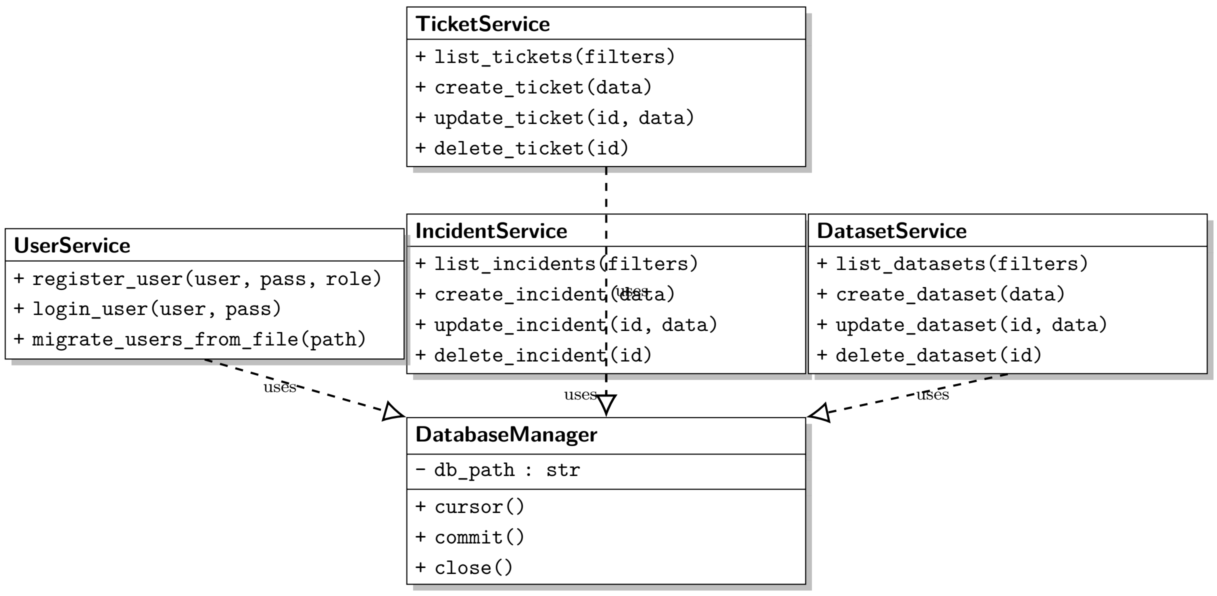
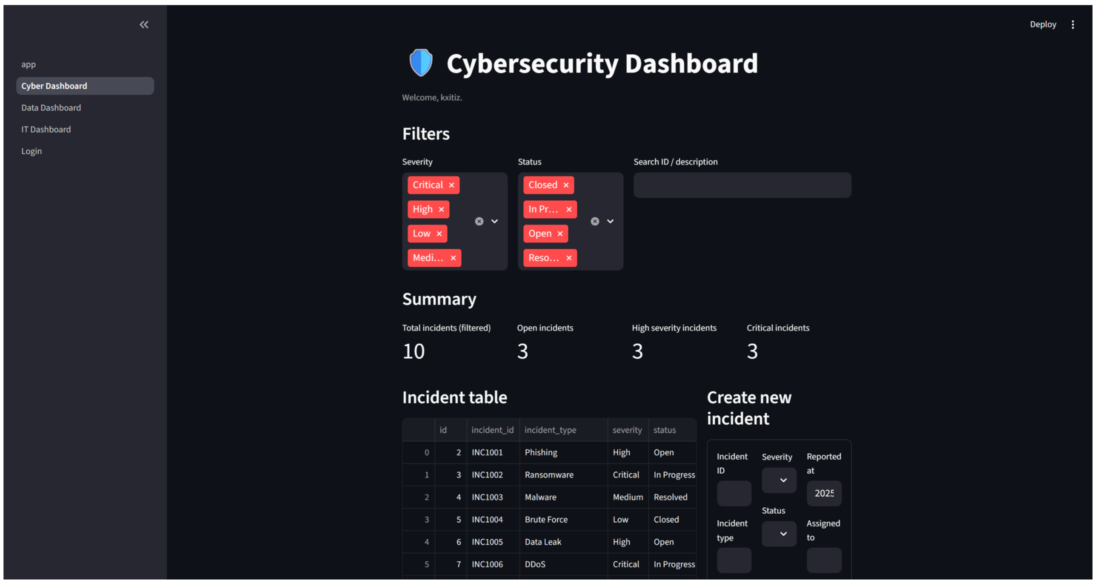


Figure 5: UML class diagram for the service layer and shared `DatabaseManager`.

2.4 Key Features

Cybersecurity Dashboard

The Cybersecurity dashboard allows analysts to filter incidents by severity, status, or keyword, and to create, update, and delete records. Visualisations show severity distribution, status breakdown, and simple time trends [5, 4].



Cybersecurity Dashboard

Welcome, kxitiz.

Filters

Severity: Critical ×, High ×, Low ×, Medi… ×

Status: Closed ×, In Pr… ×, Open ×, Reso… ×

Search ID / description

Summary

Total incidents (filtered) 10

Open incidents 3

High severity incidents 3

Critical incidents 3

Incident table

	id	incident_id	incident_type	severity	status
0	2	INC1001	Phishing	High	Open
1	3	INC1002	Ransomware	Critical	In Progress
2	4	INC1003	Malware	Medium	Resolved
3	5	INC1004	Brute Force	Low	Closed
4	6	INC1005	Data Leak	High	Open
5	7	INC1006	DDoS	Critical	In Progress

Create new incident

Incident ID

Severity

Reported at: 2025

Status

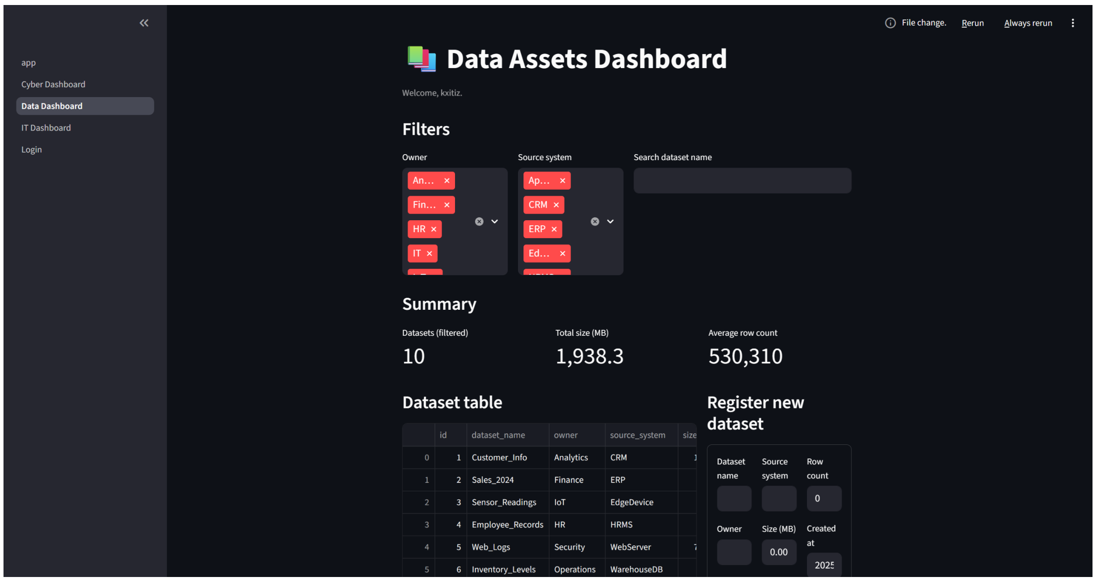
Incident type

Assigned to

Figure 6: Cybersecurity dashboard showing incident table, filters, and analytics.

Data Science Dashboard

The Data dashboard manages metadata for internal datasets. Users can register new datasets, modify owners or descriptions, and delete entries that are no longer needed.



Data Assets Dashboard

Welcome, kxitiz.

Filters

Owner: An… × Fin… × HR × IT ×

Source system: Ap… × CRM × ERP × Ed… ×

Search dataset name

Summary

Datasets (filtered)	Total size (MB)	Average row count
10	1,938.3	530,310

Dataset table

	id	dataset_name	owner	source_system	size
0	1	Customer_Info	Analytics	CRM	1
1	2	Sales_2024	Finance	ERP	
2	3	Sensor_Readings	IoT	EdgeDevice	
3	4	Employee_Records	HR	HRMS	
4	5	Web_Logs	Security	WebServer	7
5	6	Inventory_Levels	Operations	WarehouseDB	

Register new dataset

Dataset name

Source system

Row count: 0

Owner

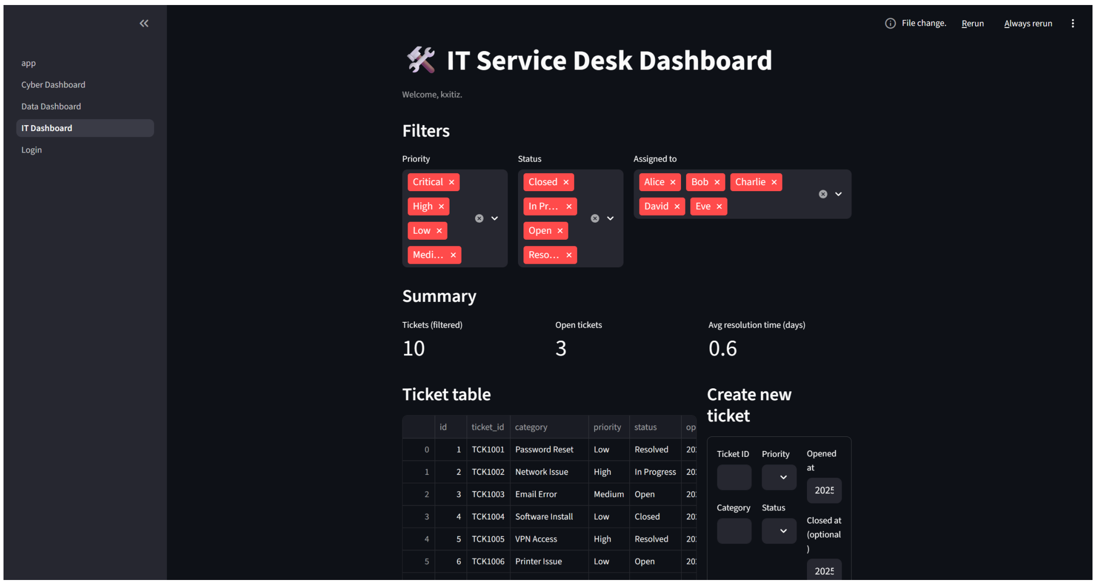
Size (MB): 0.00

Created at: 2025

Figure 7: Data Science dashboard visualising dataset metadata and storage.

IT Operations Dashboard

The IT dashboard tracks support tickets. Each ticket records status, priority, assignee, and timestamps. Visualisations show the distribution of tickets by priority and status.



IT Service Desk Dashboard

Welcome, kxitiz.

Filters

Priority: Critical × High × Low × Medi… ×

Status: Closed × In Pr… × Open × Reso… ×

Assigned to: Alice × Bob × Charlie × David × Eve ×

Summary

Tickets (filtered)	Open tickets	Avg resolution time (days)
10	3	0.6

Ticket table

	id	ticket_id	category	priority	status	op
0	1	TCK1001	Password Reset	Low	Resolved	20
1	2	TCK1002	Network Issue	High	In Progress	20
2	3	TCK1003	Email Error	Medium	Open	20
3	4	TCK1004	Software Install	Low	Closed	20
4	5	TCK1005	VPN Access	High	Resolved	20
5	6	TCK1006	Printer Issue	Low	Open	20

Create new ticket

Ticket ID

Priority

Opened at: 2025

Category

Status

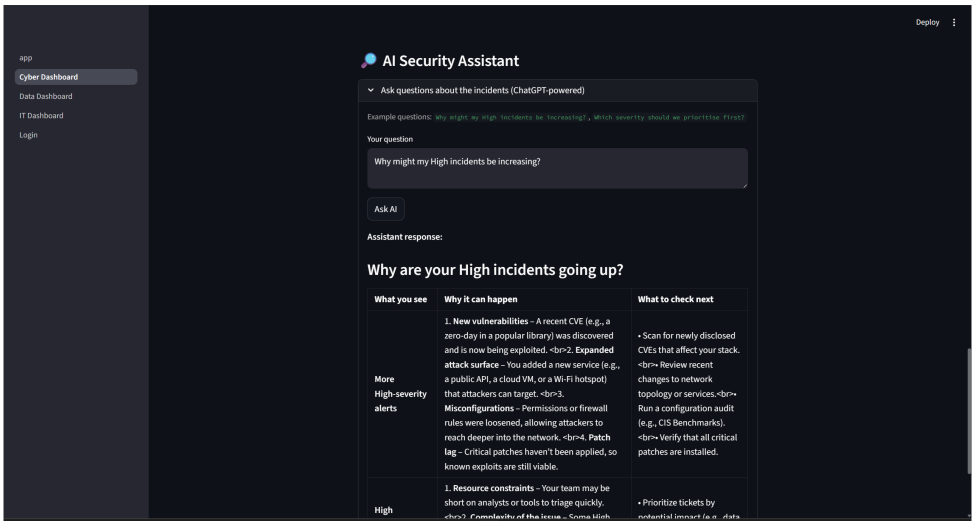
Closed at (optional): 2025

Figure 8: IT service desk dashboard with ticket list and summary charts.

AI Assistant

The AI assistant analyses the subset of incidents currently visible on the Cybersecurity dashboard. If an OpenAI-compatible API key is configured, the module calls the online model [6].

Otherwise, it falls back to a deterministic rule-based explanation, so the dashboard remains usable when the external API is unavailable.



AI Security Assistant

Ask questions about the incidents (ChatGPT-powered)

Example questions: `Why might my High incidents be increasing?`, `Which severity should we prioritise first?`

Your question

Why might my High incidents be increasing?

Assistant response:

Why are your High incidents going up?

What you see	Why it can happen	What to check next
More High-severity alerts	1. **New vulnerabilities** – A recent CVE (e.g., a zero-day in a popular library) was discovered and is now being exploited. 2. **Expanded attack surface** – You added a new service (e.g., a public API, a cloud VM, or a Wi-Fi hotspot) that attackers can target. 3. **Misconfigurations** – Permissions or firewall rules were loosened, allowing attackers to reach deeper into the network. 4. **Patch lag** – Critical patches haven't been applied, so known exploits are still viable.	• Scan for newly disclosed CVEs that affect your stack. • Review recent changes to network topology or services. • Run a configuration audit (e.g., CIS Benchmarks). • Verify that all critical patches are installed.
High	1. **Resource constraints** – Your team may be short on analysts or tools to triage quickly.	• Prioritize tickets by

Figure 9: AI assistant answering questions about the filtered incident set.

3 Reflection and Conclusion

3.1 Learning Reflection

This project required combining several skills into one coherent application. I gained practical experience with password hashing, relational schema design, and Streamlit state management [1]. Working with real data in SQLite improved my understanding of how back-end storage supports interactive interfaces [2]. The refactoring into a service layer showed the benefits of separating UI concerns from business logic and data access.

Developing the dashboards strengthened my ability to use pandas and Plotly to produce visual summaries [4, 5]. Integrating the AI assistant required careful error handling and sensible fallbacks [6].

3.2 Challenges Faced

Circular import errors occurred when service classes and Streamlit pages referenced each other; this was solved by moving all database access into dedicated modules and keeping imports one-directional. Streamlit's rerun behaviour caused forms to reset until I handled state through `st.session_state`. Loading CSV data exposed mismatches between column names and table definitions, which required validation and cleaning.

3.3 Conclusion

The final Multi-Domain Intelligence Platform meets the Tier 3 specification by integrating three domain dashboards with secure authentication, persistent storage, CRUD operations, visual analytics, and an optional AI assistant. The layered architecture and service-based design make the codebase easier to extend and maintain.

A Running Instructions

1. Install dependencies:

```
pip install streamlit pandas bcrypt plotly
```

2. Initialise the SQLite database:

```
python -m DB.main
```

3. Run the Streamlit app:

```
streamlit run app.py
```

References

[1] Streamlit Inc. (2024) *Streamlit Documentation*. Available at: https://docs.streamlit.io (Accessed: 12 December 2025).

[2] SQLite Consortium (2024) *SQLite Documentation*. Available at: https://www.sqlite.org/docs.html (Accessed: 12 December 2025).

[3] PyPI (2024) *bcrypt Python Package*. Available at: https://pypi.org/project/bcrypt/ (Accessed: 12 December 2025).

[4] The pandas development team (2024) *pandas Documentation*. Available at: https://pandas.pydata.org/docs/ (Accessed: 12 December 2025).

[5] Plotly Technologies Inc. (2024) *Plotly Python Graphing Library*. Available at: https://plotly.com/python/ (Accessed: 12 December 2025).

[6] OpenAI (2024) *Chat Completions API Reference*. Available at: https://platform.openai.com/docs/api-reference/chat (Accessed: 12 December 2025).