Middlesex University Dubai

Project Report (CST1510)

Programming for Data Communication and Networks

Mr. Santhosh Menon

Aisha Mahmood Nyako

M01027193

Project Report

Name: Aisha Mahmood Nyako
Student ID: M01027193
Program: Cybersecurity and Digital Forensics

GitHub Repository: https://github.com/aishamhn/CW2_CST1510

Disclaimer

It is important to disclose that AI tools were utilised for debugging errors, organising code structure, and clarifying complex technical concepts throughout this project.

Introduction

A Multi-Domain Intelligence Platform is a system that combines, analyses, and visualizes data from multiple domains of an organization. Its primary purpose is not just to let users edit data **(CRUD),** but to also generate insights and spot patterns regarding security threats or operational issues.

The Multi-Domain Intelligence Platform's job is straightforward. It basically acts as the central hub that makes sense of data from different parts of a company. It also helps integrate data and finds hidden risks. I chose to do all 3 domains (Cybersecurity, Data Science and IT Operations) which is tier 3. However, Only the Cybersecurity tab allows the user to perform CRUD (Create, Read, Update and Delete) operations while the other two domains are read only.

How I Built the System

I began my project in Week 7 by creating a very simple login and sign-up application that ran directly in the terminal using a file called 'authorization.py'. I used **Bcrypt** to hash passwords because it's much safer than encryption. When a user logs in, the system hashes their entered password and only compares the two hashes. At this stage, I just stored the usernames and passwords in a file called 'users.txt'.

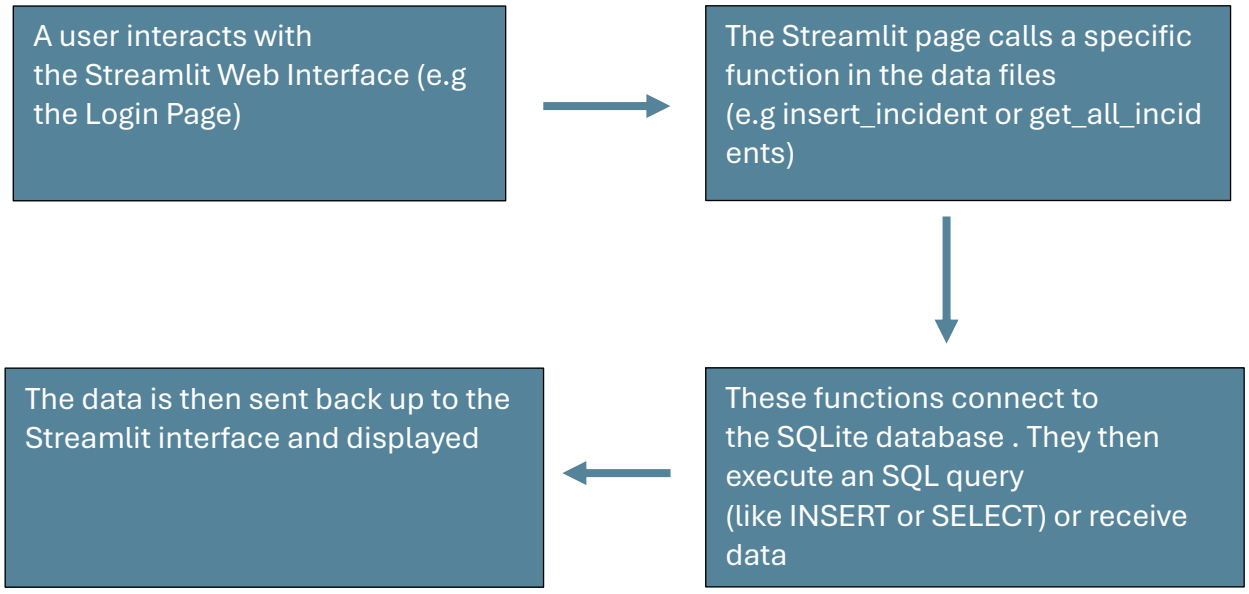
In Week 8 we started learning about Databases and **SQL**. This made me realise that storing passwords even if they're hashes, in a basic text file is not secure at all and it would be better to store it in a table in a database. The lab document gave us a lot of the

code and the file structure for the new database setup. I copied the file structure and code, but I kept running into errors and confusion trying to connect everything properly. I used AI tools here to figure things out and debug the issues. Eventually, I managed to migrate all the old users in the 'users.txt' file over to the new SQL database. I created tables for the users and downloaded the csv files for all three domains (Cyber Incidents, IT Tickets, and Datasets).

I then created all the necessary files for the **CRUD** functions. Initially, I just ran test CRUD functions in the terminal, but I hadn't yet created a way for a user to actually perform CRUD operations.

In week 9, we started using **streamlit.** I created the main file, 'app.py'. I then stopped using the command-line terminal to run my application because I could now use a browser. I then implemented different pages for navigation, used the sidebar and added simple charts.

The System Structure



The entire application runs on this structure. It moves information from user input to the database and back again to display the results.

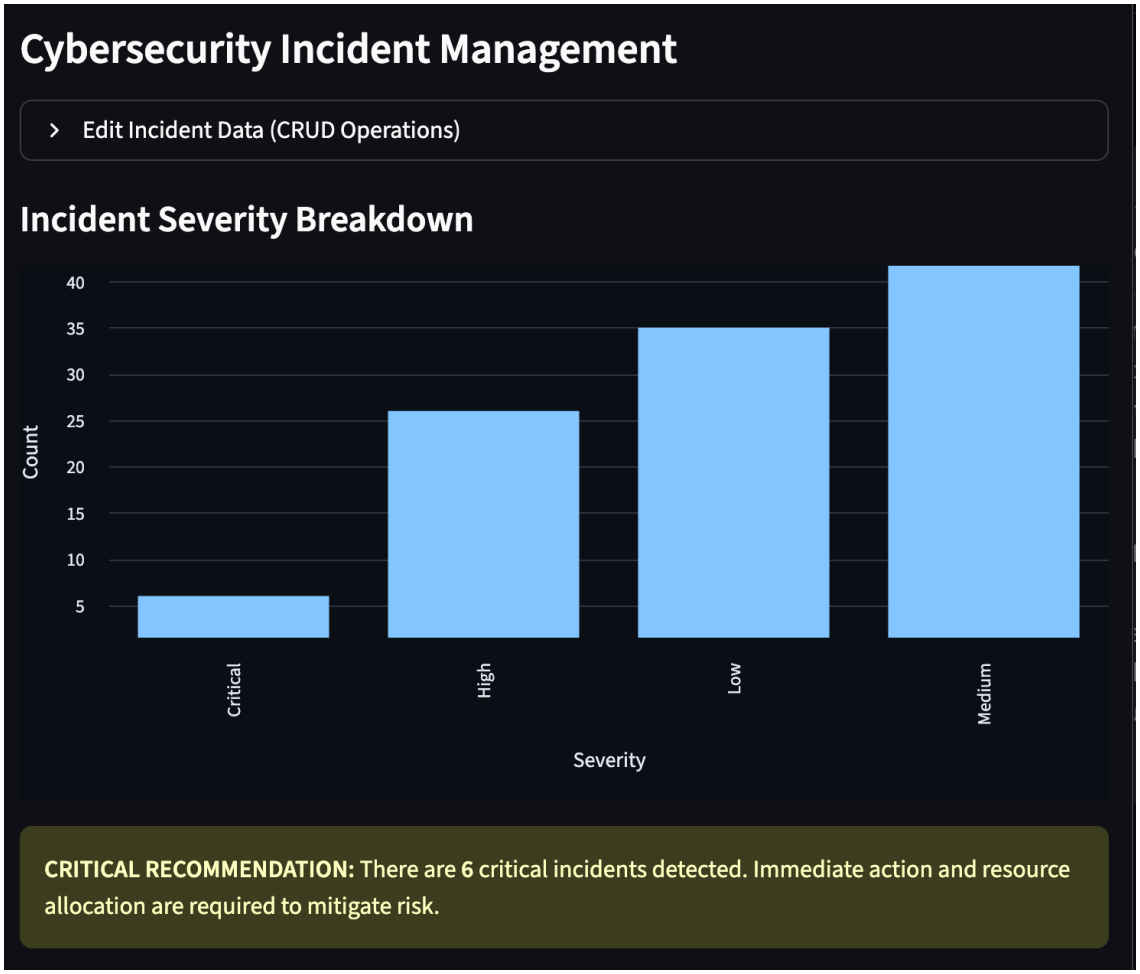
Code Organization

Since I wasn't present in the Week 11 lecture, I chose not to use classes in my code as I didn't get to fully grasp how to properly implement it. Instead, I focused on a modular and function-based approach to keep the code organised. I used:

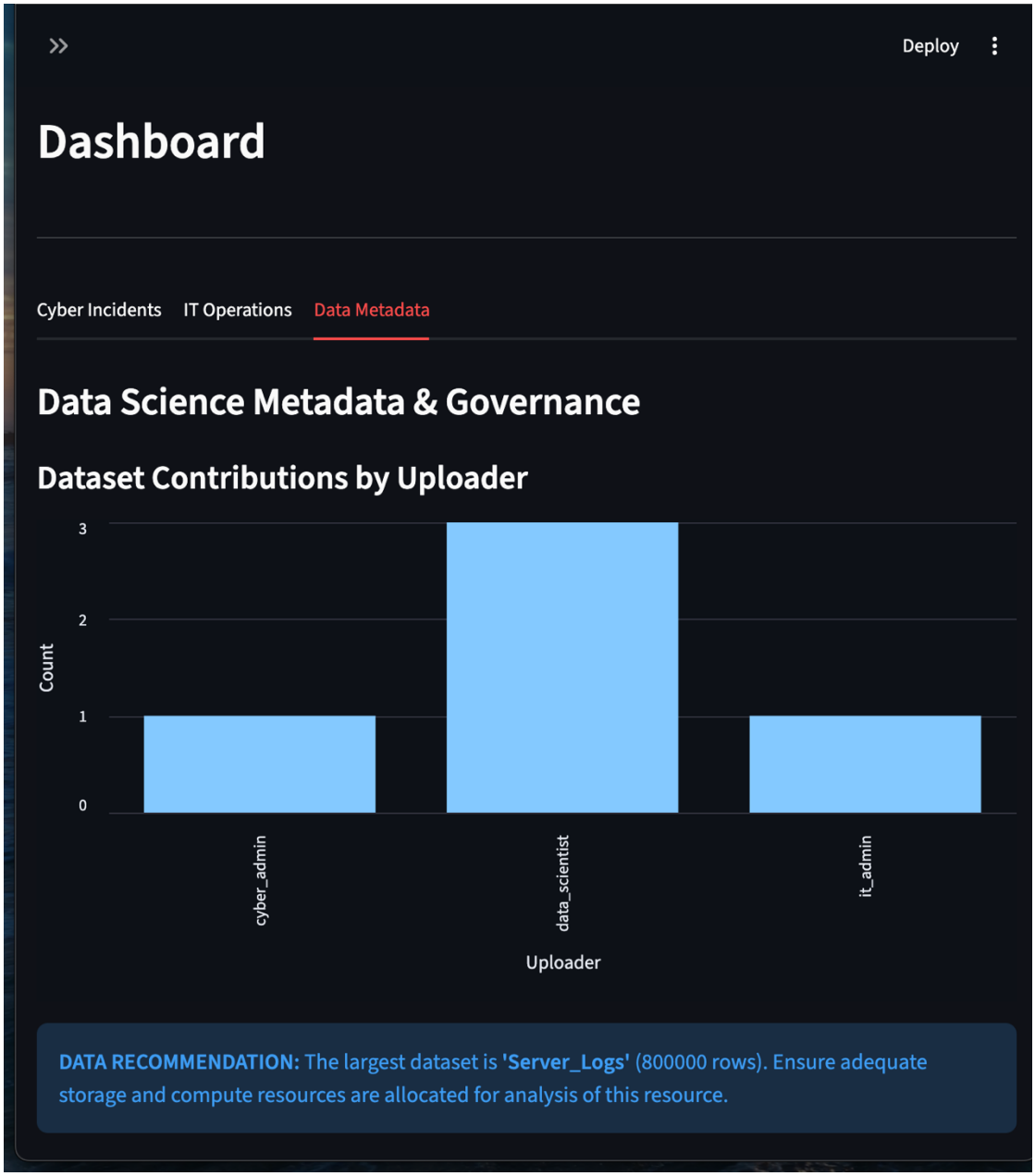
- **UI Files (pages/):** These are all the Streamlit files (like login.py, dashboard.py). Their main job is to handle the user interface and call specific data functions.
- **Data Files (app/data/):** These contain all the functions that talk directly to the database (e.g incidents.py, tickets.py). All the SQL and connection code is kept here.
- **Service Files (app/services/):** These files hold the functions for hashing/login or the AI integration.

My dashboard initially just showed read only data from the database. The lecturer pointed out that I wasn't actually letting the user create, update or delete anything. I fixed it and implemented the full user input CRUD functions for the **cyber security tab** only.

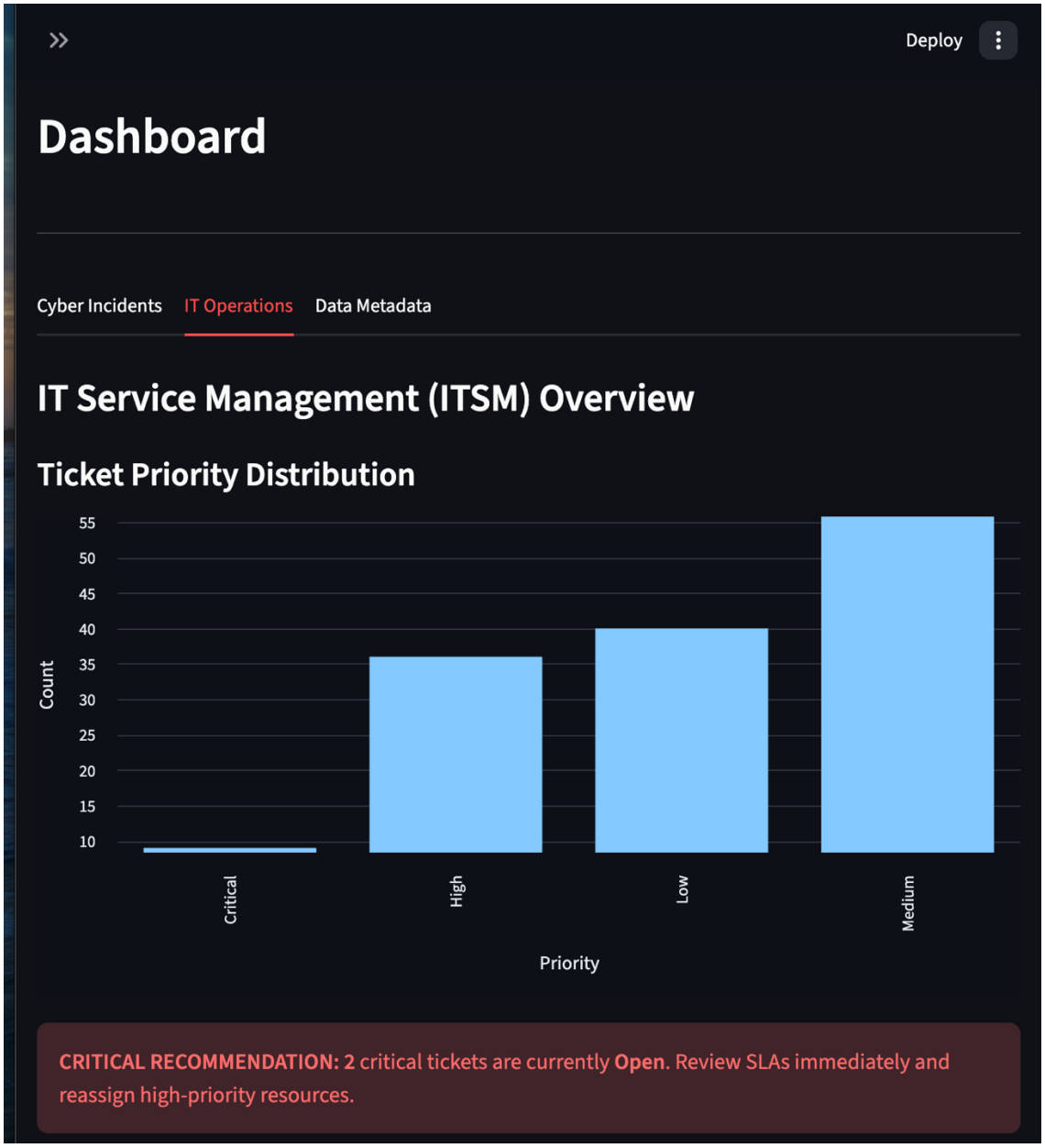
My visualizations are placed directly on the dashboard tabs.



- The **Cyber Security** Tab shows a Bar Chart of incident counts grouped by Severity. The data here is ordered by risk level, not just count. This gives the user an idea of the most serious problems. The tab also has an alert box if any critical incidents exist.

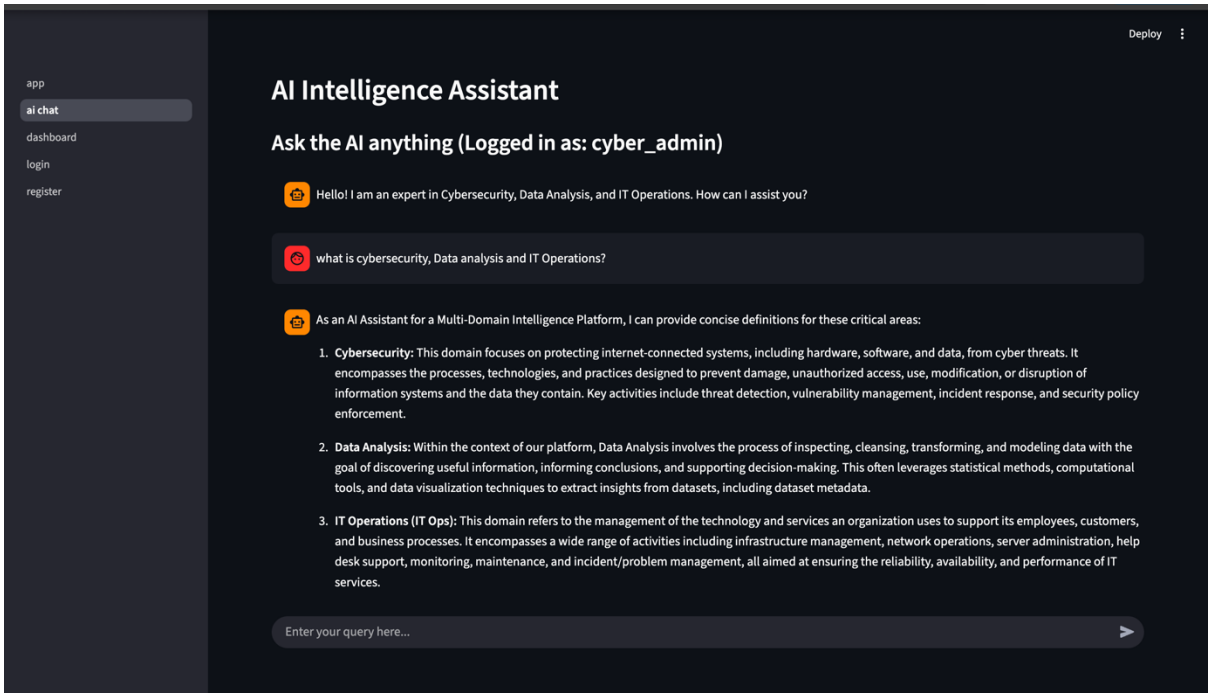


- The **Data Science Metadata** tab is read only. It has a Bar Chart that visualizes the dataset contributions by Uploader. It also has an alert box that identifies the largest dataset by rows and gives a recommendation for resource allocation.



- The **IT Operations** tab is also read only. It displays a Bar Chart of tickets grouped by **Priority**, ordered by risk. There is also an alert box that provides a recommendation if any Critical tickets are currently Open.

In Week 10/11, I managed to integrate the Google **Gemini AI** into my code. This involved creating two new files. one file “ai_service.py” imported the AI and gave it instructions to act as a multi-domain intelligence assistant. The other file “ai_chat.py” built the graphical page for the chat inside Streamlit. The AI feature is fairly basic right now. I really wanted it to be more connected with the database so that it could analyse specific data and give recommendations, but I got a technical issue and decided to keep it as a standalone assistant.



Reflection & Conclusion

Over the five weeks of working on this project, I can definitely say that it has been interesting and stressful. Besides coding skills, I'd say the most valuable skill I gained is time management. This project really pushed me to be more organized and disciplined with my tasks. I had to dedicate time to work both inside and outside of university hours. I learned how to use various new modules like **Bcrypt, SQLite** and **Streamlit.** I also gained a basic understanding of GitHub. However, I definitely need more time exploring GitHub to fully get the hang of it. In general, my understanding of the theoretical concepts on how applications and platforms work has deepened a lot. I learned how to turn simple code that ran in the terminal, into a fully functional GUI platform. I also never knew how AI was integrated into a software until now. Implementing it felt incredible. This entire process allowed me to build directly upon the knowledge I gained during IFP.

I honestly felt quite stressed, especially during the first few weeks. In my opinion, the jump from Project 1 to Project 2 was quite massive. We went from writing simple code to attempting a whole Intelligence Platform. Because of this complexity, AI became a crucial helping hand in my project. This experience taught me how to do efficient AI prompting to solve problems. I also feel that my understanding advanced more on the theoretical side than the practical technical side of building the project. This is because most of the code was provided within the lab tasks, which led to a lot of copying and pasting and, frankly, confusion. It felt like we would get a good introduction to the basic concept and code in the lecture class, but then the lab task itself would be much more complicated than the initial lesson. I faced a technical issue when trying to link the AI to the database to give more insightful recommendations. I eventually just

decided to keep the AI as a standalone assistant. I also ran into an issue where my API key had gotten exposed on github. I fixed that by moving the API key into a secret file called "**secrets.toml**".

All in all, this experience has been truly enlightening when it comes to understanding the functionality of applications, the importance of security and using github. However, I plan to do more individual studying because the process of learning the concepts felt a bit rushed. The main achievement of this project is definitely a deepened understanding of complex systems, and the importance of security in applications. The features I would like to improve on are my AI integration with the database so that it could be overall more useful. I would also like to implement classes in my code.