CST1510 CW 2: Project Guide
Multi-Domain Intelligence Platform

Table of Contents

Part 1: Project Scenario and Assessment Overview

Project Goal: The Unified Intelligence Platform

You are tasked with designing and developing a **Multi-Domain Intelligence Platform**—a unified web application built with Python and Streamlit. This platform must serve three distinct user groups: Cybersecurity Analysts, Data Scientists, and IT Administrators, by providing **high-value analysis, insights, and operational capabilities** to address specific, real-world problems within each domain.

The Domain-Specific Problems (The "Why")

Your application must be designed to solve the following critical issues for each user group:

Domain	Core Problem to Address	High-Value Insight Students Must Deliver
Cybersecurity	**Incident Response Bottleneck:** The security team is facing a **recent, targeted surge in Phishing incidents**, leading to a growing backlog of high-severity, unresolved cases.	**Identify the specific threat trend** (Phishing spike) and analyze the **response bottleneck** (e.g., which threat category has the longest resolution time).
Data Science	**Data Governance & Discovery:** The team must manage a growing catalog of large datasets uploaded by various departments (IT, Cyber), requiring quality checks and resource management.	**Analyze dataset resource consumption** (size/rows) and **data source dependency** to recommend data governance and archiving policies.
IT Operations	**Service Desk Performance:** The IT support team is struggling with slow resolution times, and the manager suspects a **staff performance anomaly** and process inefficiencies.	**Pinpoint the staff member or process stage** (e.g., "Waiting for User" status) that is causing the greatest delay in ticket resolution.

Tiered Technical Feature Requirements (Grading Structure)

The final grade is determined by the scope of the application you successfully implement. **All students must implement the core features (Authentication, DB, OOP, AI) for at least one domain.**

Tier	Requirement	Scope of Work	Target Grade Range
Tier 1 (Pass/Merit)	**One Complete Domain Dashboard**	Full implementation of all mandatory features (Authentication, DB, OOP, AI, Visualizations) for **ONE** chosen domain (Cybersecurity **OR** Data Science **OR** IT Operations).	**50% - 69%**
Tier 2 (Merit/Distinction)	**Two Complete Domain Dashboards**	Full implementation of all mandatory features for **TWO** domains. This requires significantly more OOP refactoring and integration work.	**70% - 84%**

Tier	Requirement	Scope of Work	Target Grade Range
Tier 3 (High Distinction)	**Three Complete Domain Dashboards**	Full implementation of all mandatory features for **ALL THREE** domains. This demonstrates mastery of integration and complex software architecture.	**85% - 100%**

Continuous Development (GitHub Engagement)

Your GitHub repository is a mandatory component of your assessment. You are expected to commit and push your work **at the end of every lab session** (or at least once per week). The history of your repository must clearly demonstrate the incremental development process outlined in Part 2.

Part 2: 5-Week Incremental Lab Specification

This plan covers the core development period from Week 7 to Week 11.

Week 7: Security & File Persistence (Hashing)

Section	Focus	Implementation Tasks	GitHub Deliverable
Theoretical Focus	**Secure Authentication Principles:** Password hashing vs. encryption. The role of salts. Introduction to bcrypt and basic file I/O in Python.	N/A	N/A

Section	Focus	Implementation Tasks	GitHub Deliverable
Practical Focus	**Implement Secure Login (Text-Based)**	1. **Setup:** Create the project directory and initial Python file (auth.py). 2. **Hashing:** Install bcrypt. Implement a function to hash a password and a function to verify a password against a hash. 3. **Registration:** Implement a command-line function to register a new user, storing the username, password_hash, and role in a simple text file (users.txt). 4. **Login:** Implement a command-line function that takes a username and password, retrieves the hash from the file, and verifies the login.	A functional command-line script (auth.py) that allows secure registration and login, with user data stored in a local file (users.txt).

Week 8: Data Pipeline & CRUD (SQL)

Section	Focus	Implementation Tasks	GitHub Deliverable
Theoretical Focus	**Relational Databases & SQL:** Introduction to SQLite. Database schema design (Normalization). Full CRUD operations and parameterized queries (SQL Injection prevention).	N/A	N/A

Section	Focus	Implementation Tasks	GitHub Deliverable
Practical Focus	**Database Migration & CRUD for All Domains**	1. **Database Manager:** Create a DatabaseManager class to handle SQLite connection and cursor operations. 2. **Schema Design:** Create the users table and the tables for **ALL THREE** domains (cyber_incidents, datasets_metadata, it_tickets). 3. **Data Migration:** Write a script to read the data from users.txt (Week 7) and insert it into the new SQLite users table. 4. **CRUD Implementation:** Implement all four CRUD operations (Create, Read, Update, Delete) for **ALL THREE** domain tables. 5. **Data Loading:** Use pandas to read the provided sample data (cyber_incidents.csv, etc.) and load it into the respective SQLite tables.	A command-line script that uses the SQLite database for all user and domain data. Full CRUD functionality demonstrated via the command line for all three domains.

Week 9: Web Interface, MVC & Visualization

Section	Focus	Implementation Tasks	GitHub Deliverable
Theoretical Focus	**Web Application Architecture:** Streamlit fundamentals, session state management, multi-page apps. Data Visualization with Plotly/Matplotlib and embedding charts in Streamlit.	N/A	N/A

Section	Focus	Implementation Tasks	GitHub Deliverable
Practical Focus	**Streamlit Conversion & Master One Domain**	1. **Streamlit Setup:** Convert the application to a Streamlit multi-page structure (app.py, pages/Login.py, pages/Dashboard.py). 2. **Login Page:** Implement the login page (Login.py) using Streamlit widgets and integrate the Week 8 database functions. Use st.session_state to manage the logged-in user and role. 3. **Dashboard Conversion:** Create the dashboard page for the student's **Primary Domain**. Convert the Week 8 CRUD operations into interactive Streamlit forms and tables. 4. **Visualization:** Implement the required analytical functions and **All Visualizations** for the chosen domain (e.g., Time-Series Plot, Bar Chart) using Plotly and embed them using st.plotly_chart().	A fully functional Streamlit web application with: Secure Login, Session Management, and **ONE** complete, interactive domain dashboard featuring all required visualizations.

Week 10: Final Dashboards & AI Integration

Section	Focus	Implementation Tasks	GitHub Deliverable
Theoretical Focus	**API Integration:** REST APIs, API keys, Environment Variables. Introduction to the OpenAI API and context-aware prompting.	N/A	N/A

Section	Focus	Implementation Tasks	GitHub Deliverable
Practical Focus	**Complete Remaining Dashboards & Integrate AI**	1. **Remaining Dashboards:** Implement the remaining analytical functions and visualizations for the **Secondary/Tertiary Domains** (for Tier 2/3 students). 2. **AI Integration:** Set up the OPENAI_API_KEY environment variable. Create a helper function to call the ChatGPT API. 3. **AI Assistant:** Integrate the AI Assistant feature into **at least one** domain dashboard (e.g., a chat box that provides security advice or statistical explanations). Ensure proper error handling.	A complete, fully functional Multi-Domain Intelligence Platform that meets all mandatory requirements (Authentication, DB, Visualizations, AI).

Week 11: Software Architecture & Polish

Section	Focus	Implementation Tasks	GitHub Deliverable
Theoretical Focus	**Software Refactoring:** Benefits of OOP. Review of class design, constructors, methods, and attributes. Final project documentation and submission requirements.	N/A	N/A
Practical Focus	**OOP Refactoring & Final Documentation**	1. **Class Creation:** Create the core entity classes (User, SecurityIncident, Dataset, ITTicket, etc.). 2. **Refactoring:** Refactor the procedural code from Weeks 7-10 to use the new OOP class structure. Move business logic into class methods. 3. **Documentation:** Write the final README.md and ensure all code has clear comments and docstrings. 4. **Report Preparation:** Finalize the required diagrams and begin drafting the technical report.	The final project code, fully refactored into a clean, maintainable OOP structure. All documentation is complete, and the repository is ready for submission.

Part 3: Report Template

Word Count: 1000-1500 Words

Section 1: Introduction and Project Scope

- **Student Information:** [Full Name, Student ID, Degree Program]
- **Project Goal:** Briefly introduce the Multi-Domain Intelligence Platform and its purpose.
- **Tiered Scope:** Clearly state your chosen primary domain and the tier of implementation (Tier 1, 2, or 3).

Section 2: System Architecture and Implementation

This section should provide a narrative overview of how you built the application, referencing the key concepts learned.

- **Data Layer and Security:** Describe your approach to **secure user authentication** (Hashing) and your **database design** (SQL).
- **Application Structure (MVC and Data Flow):**
    - Include a simple **Data Flow Diagram (DFD)** or **MVC Diagram** to illustrate how the components interact (Model, View, Controller).
    - Use the diagram to explain the flow of data from user input to the database and back to the Streamlit interface.
- **Object-Oriented Programming (OOP) Design:**
    - Include a simple **UML Class Diagram** or **Entity-Relationship (ER) Diagram** showing your core classes and their relationships.
    - Explain how the **OOP Refactoring** (Week 11) improved your code's organization.
- **Feature Integration:** Briefly describe the implementation of the **interactive visualizations** and the **ChatGPT AI Assistant**.

Section 3: High-Value Analysis and Insights

This is the core analytical section. Focus on the real-world problem your implemented domain(s) address.

- **Problem Statement:** Restate the specific, high-value problem you chose to solve (e.g., Phishing surge / Staff performance anomaly).
- **Analysis and Findings:** Present the key insights derived from your data and visualizations. Use screenshots of your charts to support your findings.

- **Actionable Recommendations:** Based on your analysis, provide clear, actionable recommendations that the end-user would take to solve the problem.

Section 4: Reflection and Conclusion

- **Learning Reflection:** Reflect on your personal learning journey over the 5-week project. What were the most valuable technical concepts you mastered?
- **Challenges:** What were the biggest technical hurdles you faced, and how did you overcome them?
- **Conclusion & Future Work:** Conclude by summarizing the platform's success in solving the core problem. Suggest one or two features you would implement next.

Appendix (Optional)

- Running Instructions (as a simple list of commands).
- Supporting code snippets or documentation.