SENTINELGUARD: AI-DRIVEN INCIDENT RESPONSE CONSOLE

Course: CCGC 5003 Application Programming
Phase: I – Project Inception
Submission Date: October 27, 2025

GROUP DETAILS
• Group Name: CyberSentinel Labs
• Group Members: Gurmatsingh Sour (solo project team)

---

1. PROJECT OVERVIEW
SentinelGuard is a Python-based cybersecurity operations console that helps security analysts triage, investiga

2. CORE USE CASE SCENARIOS
1. Log Intake & Asset Linking: Analysts register new security events by selecting the affected asset, attaching ra
2. AI Risk Assessment: On demand, the AI module evaluates the ingested event, assigns a severity score, and
3. Threat Intelligence Enrichment: Analysts trigger enrichment to compare the event indicators (IP, hash, domai
4. Incident Task Orchestration: Analysts create and track remediation tasks (containment, eradication, recovery
5. Compliance & Reporting: When incidents close, the system compiles AI summaries, analyst actions, and tim

Each scenario launches from the shared dashboard, and the state captured in one flow (e.g., AI severity score)

3. DATABASE TABLES (MODEL LAYER CANDIDATES)
• analysts – registered security team members managing incidents and tasks.
• assets – catalog of protected systems and their business owners.
• security_events – normalized representation of incoming alerts/logs.
• ai_assessments – AI-generated severity scores, summaries, and recommended actions for each event.
• incident_tasks – remediation activities tied to specific security events.
• threat_intel_matches – cross-reference table tracking enrichment hits against known campaigns.
• compliance_reports – finalized incident summaries for governance and audit requirements.

(At least one table supports every scenario above; some scenarios share tables to maintain referential integrity.

4. KEY FIELDS AND DATA TYPES
analysts
• analyst_id INTEGER PRIMARY KEY
• full_name TEXT NOT NULL
• email TEXT UNIQUE NOT NULL
• role TEXT CHECK(role IN ('Tier 1','Tier 2','Tier 3','Manager'))
• on_call BOOLEAN DEFAULT 0
• created_at DATETIME DEFAULT CURRENT_TIMESTAMP

assets
• asset_id INTEGER PRIMARY KEY
• hostname TEXT NOT NULL
• ip_address TEXT NOT NULL
• business_owner TEXT NOT NULL
• criticality TEXT CHECK(criticality IN ('Low','Moderate','High','Critical'))

• last_patch_date DATE

security_events
• event_id INTEGER PRIMARY KEY
• asset_id INTEGER REFERENCES assets(asset_id)
• ingest_time DATETIME DEFAULT CURRENT_TIMESTAMP
• source TEXT NOT NULL
• raw_log TEXT NOT NULL
• status TEXT CHECK(status IN ('New','Triaged','In Progress','Resolved'))

ai_assessments
• assessment_id INTEGER PRIMARY KEY
• event_id INTEGER REFERENCES security_events(event_id) ON DELETE CASCADE
• model_version TEXT NOT NULL
• risk_score REAL CHECK(risk_score BETWEEN 0 AND 1)
• severity_label TEXT CHECK(severity_label IN ('Low','Medium','High','Critical'))
• recommended_action TEXT
• summary TEXT

incident_tasks
• task_id INTEGER PRIMARY KEY
• event_id INTEGER REFERENCES security_events(event_id) ON DELETE CASCADE
• assigned_to INTEGER REFERENCES analysts(analyst_id)
• task_description TEXT NOT NULL
• priority TEXT CHECK(priority IN ('Low','Medium','High','Urgent'))
• due_date DATE
• status TEXT CHECK(status IN ('Pending','In Progress','Blocked','Complete'))

threat_intel_matches
• match_id INTEGER PRIMARY KEY
• event_id INTEGER REFERENCES security_events(event_id)
• indicator_type TEXT CHECK(indicator_type IN ('IP','Domain','Hash','URL'))
• indicator_value TEXT NOT NULL
• threat_actor TEXT
• confidence REAL CHECK(confidence BETWEEN 0 AND 1)

compliance_reports
• report_id INTEGER PRIMARY KEY
• event_id INTEGER REFERENCES security_events(event_id)
• generated_by INTEGER REFERENCES analysts(analyst_id)
• generated_at DATETIME DEFAULT CURRENT_TIMESTAMP
• regulation TEXT NOT NULL
• summary_text TEXT NOT NULL
• export_path TEXT

5. TEAM ROLES & RESPONSIBILITIES
• Gurmatsingh Sour: Acts as project lead, full-stack developer, and AI engineer. Responsibilities include gatheri

6. HIGH-LEVEL PROJECT PLAN
• Phase II (Design): Finalize ERD, define controller/UI architecture, prepare training dataset, and stub out ORM

• Phase III (Implementation): Build database migrations, implement Tkinter workflows, integrate AI inference mo
• Phase IV (Testing & Delivery): Populate test data, execute end-to-end scenario validations, capture performar

The outlined scope satisfies the course requirement of integrating multiple use cases, a database-backed mode