CSPM STD-

Almog Asia - 325553543

Hadar Nino - 314712464

Maor Hadad Levy - 212708077

**1. Introduction**

This Software Test Design document provides detailed test cases for verifying the correct functioning of the CSPM (Cloud Security Posture Management) project. The objective is to ensure the proper operation of the security log analysis system, including the React frontend, Python Flask backend, and machine learning components for risk assessment and alerting. The system processes security logs from AWS, Azure, and GCP, analyzing them using an Isolation Forest model trained on 2 million unsupervised log entries.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Description | Preconditions | Test Steps | Expected Result | Actual Result |
| TC-001 | Ingest AWS security log and process | Backend API running, AWS test account available | 1. Upload sample AWS log file 2. Trigger log processing endpoint | Log is parsed, validated, and stored in database | OK |
| TC-002 | Validate log data format | Backend API running | 1. Upload log file with invalid format 2. Trigger log processing | System rejects log and returns validation error | OK |
| TC-003 | Run risk assessment on ingested log | Log data available in database | 1. Initiate risk assessment via API 2. Observe risk score output | Risk score is calculated and returned | OK |
| TC-004 | Risk level categorization | Risk assessment module running | 1. Process log with known risk pattern | Correct risk level (Low/Medium/High) is assigned | OK |
| TC-005 | Alert generation for high risk | High-risk log entry present | 1. Process high-risk log 2. Check alert system | Alert is generated and displayed | OK |
| TC-006 | No alert for low risk | Low-risk log entry present | 1. Process low-risk log 2. Check alert system | No alert is generated | OK |
| TC-007 | Display risk level on dashboard | Frontend and backend running, risk data available | 1. Log in to dashboard 2. Navigate to risk overview | Risk level is displayed correctly | OK |
| TC-008 | Dashboard data refresh | Frontend running, new data ingested | 1. Ingest new log 2. Refresh dashboard | Dashboard updates with new risk data | OK |
| TC-009 | API endpoint for log retrieval | Backend API running, logs available | 1. Call log retrieval endpoint | Logs are returned in correct format | OK |
| TC-010 | API endpoint for risk score retrieval | Backend API running, risk data available | 1. Call risk score endpoint | Risk scores are returned in correct format | OK |
| TC-011 | Frontend-backend integration | Both frontend and backend running | 1. Perform action on frontend that triggers backend API | Data flows correctly between frontend and backend | OK |
| TC-012 | Error handling: missing log fields | Backend API running | 1. Upload log with missing required fields | System returns descriptive error message | System returned error message, but it was missing field details |
| TC-013 | Error handling: backend unavailable | Frontend running, backend stopped | 1. Attempt to fetch data from frontend | User sees error notification about backend unavailability | OK |
| TC-014 | ML model inference | ML model deployed, log data available | 1. Submit log for inference | Model returns risk prediction | OK |
| TC-015 | ML model threshold testing | ML model deployed | 1. Submit logs with varying risk levels | Model assigns correct risk category based on thresholds | OK |
| TC-016 | Performance: large log batch | Backend and database running | 1. Upload large batch of logs | System processes logs within acceptable time | OK |
| TC-017 | Performance: dashboard with large dataset | Frontend and backend running, large dataset available | 1. Load dashboard | Dashboard remains responsive and displays all data | OK |
| TC-018 | Edge case: empty log file | Backend API running | 1. Upload empty log file | System returns error or warning about empty file | OK |
| TC-019 | Edge case: duplicate log entries | Backend API running | 1. Upload duplicate logs | System handles duplicates appropriately (e.g., ignores or flags) | System flagged duplicates, but did not provide a warning message to the user |
| TC-020 | Log data sanitization | Backend API running | 1. Upload log with malicious input | System sanitizes input and prevents injection | OK |
| TC-021 | Risk level update after new log | Risk data available, new log ingested | 1. Ingest new log 2. Trigger risk assessment | Risk level is recalculated and updated | OK |
| TC-022 | API returns error for invalid request | Backend API running | 1. Call API with invalid parameters | API returns appropriate error code and message | OK |
| TC-023 | Visualization: risk trend over time | Frontend and backend running, historical data available | 1. View risk trend chart on dashboard | Chart displays correct trend based on data | OK |