**Software Test Plan (STP) - CSPM Project**

Almog Asia - 325553543

Hadar Nino - 314712464

Maor Hadad Levy - 212708077

**1. Introduction**

This Software Test Plan outlines the testing strategy for the CSPM (Cloud Security Posture Management) project. The objective is to ensure the correct functioning 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.

**2. Test Items**

* React Frontend Application
* Python Flask Backend API
* Machine Learning Processing Module
* Log Processing and Analysis Engine
* Risk Assessment and Alerting System
* AWS/Azure/GCP Integration Components

**3. Features to be Tested**

* Security log data ingestion and processing
* Machine learning model inference and risk calculation
* Real-time risk level assessment and alerting
* Frontend dashboard visualization and data display
* API endpoints for data retrieval and processing
* Log data validation and sanitization
* Risk level categorization and threshold testing
* Frontend-backend integration
* Error handling and edge cases

**4. Features Not to be Tested**

**(Future versions)**

* Cloud deployment functionality
* Open-source log data collection tools
* Multi-cloud deployment scenarios
* User authentication and authorization
* Data modification capabilities
* Real-time log streaming
* Advanced analytics features
* Custom rule creation and management

**5. Testing Strategy**

**a) Unit Testing:**

* Frontend component testing using Jest and React Testing Library
* Backend API endpoint testing using PyTest
* ML model inference testing
* Log processing unit tests

**b) Integration Testing:**

* Frontend-Backend API integration
* ML model integration with processing pipeline
* Database connectivity and query testing

**c) System Testing:**

* End-to-end log processing flow
* Risk assessment pipeline
* Alert generation and display
* Performance testing with large log datasets

**d) Acceptance Testing:**

* User interface validation
* Risk level accuracy verification
* Alert system reliability

**6. Test Environment**

**Hardware Requirements:**

* Development machines with minimum 8GB RAM
* Sufficient storage for log data processing

**Software Requirements:**

* Node.js and npm for frontend testing
* Python 3.8+ for backend testing
* Flask development server
* React development environment
* AWS/Azure/GCP test accounts
* Testing frameworks:  
  + Jest and React Testing Library
  + PyTest
  + Postman for API testing

**7. Responsibilities**

Almog:

- Frontend testing and validation

- React component testing

- UI/UX testing

- Dashboard visualization testing

- Frontend-backend integration testing

Maor:

- Backend API testing

- Flask endpoint validation

- Database integration testing

- Log processing pipeline testing

- Performance testing

Hadar:

- ML model testing and validation

- Risk assessment algorithm testing

- Cloud integration testing (AWS/Azure/GCP)

- System integration testing

- Acceptance testing coordination

**8. Schedule**

**Week 1 (May 25 - May 31):**

* Unit testing setup and implementation
* Frontend component testing
* Backend API testing

**Week 2 (June 1 - June 7):**

* Integration testing
* ML model testing
* Performance testing

**Week 3 (June 8 - June 14):**

* System testing
* Acceptance testing
* Bug fixes and final validation

**9. Risks and Contingencies**

**A. Technical Risks**

**1. ML Model Performance**

* **Risk**: Model accuracy degradation with new log patterns
* **Solution**: Real-time performance monitoring and rule-based fallback system

**2. Data Processing**

* **Risk**: System slowdown during large log volume processing
* **Solution**: Implemented batch processing and multi-level caching

**3. Cloud Services**

* **Risk**: Unexpected API changes or service disruptions
* **Solution**: Comprehensive error handling and mock service implementation

**4. Frontend Performance**

* **Risk**: Dashboard responsiveness issues with large datasets
* **Solution**: Optimized rendering and intelligent data pagination

**B. Project Risks**

**1. Timeline**

* **Risk**: Testing phase delays due to integration complexity
* **Solution**: Prioritized testing pipeline and daily progress tracking

**2. Resources**

* **Risk**: Cloud service costs exceeding allocated budge
* **Solution**: Strategic use of free tiers and automated cost monitoring

**3. Security**

* **Risk**: Potential exposure of sensitive log data during testing
* **Solution**: Strict data anonymization and role-based access control