*Open Source Intelligence Application*

**Project Management Plan**



Department of Information Technology and Management

*November 2024*

**Revision History**

Note: The revision history cycle begins once changes or enhancements are requested after the document has been baselined.

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 11/25/2024 | 2.0 | Final Project Management Plan | Henil Gandhi |
| 11/11/2024 | 1.0 | Developed an Initial Project Management Plan | Henil Gandhi |

Instructions

| **Activity** | **New Capability (1)** | **Feature Enhancement (2)** |
| --- | --- | --- |
| **Field Deployment (A)** | No | No |
| **Cloud/Web Deployment (B)** | Yes | Yes |
| **Mobile Application (C)** | No | No |

**Table Of Content**

1. **Introduction**  
   1.1 Project Overview  
   1.2 Scope Statements  
   1.3 Goals and Objectives  
   1.4 Stakeholders and Key Personnel
2. **Project Organization**
3. **Acquisition Process**
4. **Monitoring and Control Mechanisms**
5. **Systems Security Plans and Requirements**
6. **Work Breakdown Structure (WBS)**
7. **Project Success Criteria**
8. **Communication Management Plan**
9. **Risk Management Plan**
10. **Software Configuration Management (SCM) Plan**
11. **Training Plan**
12. **Quality Assurance Plan**
13. **Project Measurement Plan**  
    13.1 Description  
    13.2 Performance Measurements
14. **Reference Materials**

**1. Introduction**

**1.1 Project Overview**

The project focuses on developing an Open-Source Intelligence (OSINT) web application that fetches and analyzes vulnerability data from multiple APIs, such as NVD and CVE. Users will interact with the application through predefined questions, enabling them to generate insights and reports about vulnerabilities based on parameters such as CVSS scores, attack vectors, and severity. The tool is designed to assist cybersecurity teams in prioritizing and addressing critical risks.

**1.2 Scope Statements**

* **In-Scope**:
  + Integration of four APIs to fetch and analyze vulnerability data.
  + User interface to select predefined questions and generate corresponding outputs.
  + Dynamic reporting on vulnerabilities, trends, and severity levels with visualizations (e.g., charts).
* **Out-of-Scope**:
  + Direct implementation of security patches or mitigations.
  + Real-time vulnerability detection beyond API data.

**1.3 Goals and Objectives**

* Provide a seamless and interactive platform to analyze and report vulnerabilities.
* Deliver actionable insights for risk prioritization using data fetched from trusted APIs.
* Support informed decision-making for security teams through visual trends and detailed reports.

**1.4 Stakeholders and Key Personnel**

* **Stakeholders**:
  + Cybersecurity professionals (end-users).
  + Developers and system architects (internal team).
  + API providers (NVD, CVE, etc.).
* **Key Personnel**:
  + **Project Manager**: Oversees timelines, deliverables, and communication.
  + **Web Developer**: Leads the integration and backend development efforts.
  + **Data Analyst**: Ensures data accuracy and meaningful output generation.
  + **UI/UX Designer**: Designs the web interface for optimal usability.
  + **QA Engineer**: Verifies system performance, security, and functionality.

**2. Project Organization**

* **Development Team**: Handles core functionalities like API integration, backend logic, and frontend design.
* **QA Team**: Focuses on testing for accuracy, security, and usability.
* **Project Management Office (PMO)**: Tracks progress and mitigates risks.

**3. Acquisition Process**

* **Technologies**: Use Python/Flask for backend, React for frontend, PostgreSQL for the database, and Chart.js for data visualization.
* **Third-Party APIs**: Obtain necessary permissions for accessing NVD, CVE, and other APIs.
* **Hosting Platform**: Deploy the application on cloud infrastructure such as AWS or Azure for scalability and reliability.

**4. Monitoring and Control Mechanisms**

* Weekly sprint reviews to evaluate progress and resolve issues.
* Regular status updates using tools like Jira or Trello.
* Automated testing pipelines to ensure continuous integration and delivery (CI/CD).

**5. Systems Security Plans and Requirements**

* Secure communication using HTTPS and API keys for data fetching.
* Implement role-based access control (RBAC) for restricting access.
* Regularly monitor the system for vulnerabilities and enforce secure coding practices.

**6. Work Breakdown Structure (WBS)**

**Work Breakdown Structure (WBS)**

1. **Planning**  
   1.1 Requirements gathering.  
   1.2 Stakeholder consultations.
2. **Design**  
   2.1 UI wireframing.  
   2.2 System architecture design.
3. **Development**  
   3.1 Backend development and API integration.  
   3.2 Frontend development and visualization creation.
4. **Testing**  
   4.1 Unit and integration testing.  
   4.2 User acceptance testing (UAT).
5. **Deployment**  
   5.1 Hosting setup.  
   5.2 Final rollout and feedback.

**7. Project Success Criteria**

* Accurate and timely analysis of vulnerabilities fetched from APIs.
* User-friendly interface that meets stakeholder expectations.
* Positive feedback from cybersecurity professionals during testing phases.

**8. Communication Management Plan**

* Weekly team meetings using Zoom/Meet.
* Progress reports shared with Team.
* Documentation stored in a centralized repository (e.g., Confluence or Google Drive).

**9. Risk Management Plan**

* **Key Risks**:
  + API changes or downtimes affecting data retrieval.
  + Data inconsistencies between sources.
  + Security vulnerabilities in the web application.
* **Mitigation Strategies**:
  + Implement fallback APIs for redundancy.
  + Validate and clean data before analysis.
  + Conduct regular penetration testing.

**10. Software Configuration Management (SCM) Plan**

* Use GitHub for version control.
* Follow a clear branching strategy (main, dev, and feature branches).
* Use CI/CD pipelines for build and deployment automation.

**11. Training Plan**

* User guides and video tutorials for onboarding cybersecurity teams.
* Regular webinars to train users on advanced functionalities.

**12. Quality Assurance Plan**

* Functional testing of API integrations.
* Stress and performance testing for large data loads.
* Usability testing to ensure intuitive interaction.

**13. Project Measurement Plan**

**13.1 Description**

Performance and success are measured based on technical accuracy, system usability, and user satisfaction.

**13.2 Performance Measurements**

* **API Response Accuracy**: Verify outputs align with fetched data.
* **User Satisfaction Score**: Gather feedback post-UAT.
* **System Downtime**: Ensure 95% uptime post-deployment.

**14. Reference Materials**

* Official API documentation for NVD, CVE, and other sources.
* OWASP security guidelines for web applications.
* Best practices for data visualization and user experience design.