**Document Overview**

* **Title:** System without any Buzzwords (SwaB)
* **Version:** 1.1.2
* **Date:** [10/21/2025]
* **Authors and Roles:**   
  [Ethan Castleman - Security,  
  Travis Clark - Dev,   
  Cameron Evans - Dev,  
  Nathanael McClure - QA,   
  Clayton Warstler - DBA]

**2. Executive Summary**

* The purpose of this project is to design and implement a secure, centralized Inventory Tracking System that allows the Internal Audit department at Buzzword Software Solutions Limited to manage and monitor all company-owned physical assets, such as servers, laptops, and other electronic resources.
* This system will ensure that all asset data exists in a single source, improving accuracy, accountability, and ease of access for both employees and managers.

**3. Business Context**

* **Background:** Buzzword Software Solutions Limited Internal Audit department has requested a software system to manage the physical assets of the company.
* **Opportunity:** A single source to track the servers, employee laptops, and other electronic assets.
* **Goals & Objectives (Use Case):**
  + The ability to query resource details
  + The ability to add/update/delete resources single or multiple
  + The ability to group resources by type
  + The ability to assign resources to employees and locations

**4. Vision Statement**

* “To create a secure, scalable, and user-friendly system that empowers Buzzword Software Solutions to efficiently track, assign, and manage all company assets. With a goal of reducing audit risk, simplifying reporting, and enhancing transparency across the organization.”

**5. Customer or Market Needs**

* **Target users:**
  + Managers
  + Employees
* **What problems are they facing?**
  + A decentralized system to check where resources and assets are located and being used.

**6. Proposed Solution**

* The proposed solution is a secure, centralized Inventory Tracking System designed as a backend service using RESTful API architecture. This system will serve as the single source of truth for managing all company-owned physical assets across Buzzword Software Solutions Limited's global offices.
* The system will allow authorized users, primarily managers and internal audit staff, to perform full lifecycle management of assets, including adding, updating, deleting, grouping, and assigning resources. Employees will have read-only access to view assets assigned to them or query general resource information.
* The backend will be built with a security-first approach, incorporating role-based access control, audit logging, and data protection mechanisms. No frontend UI will be developed in this phase; instead, the system will expose a well-documented REST API for integration with other internal tools or future frontend development.
* REST API for resource management
  + Enables CRUD operations on individual or grouped assets via secure endpoints.
* Resource assignment
  + Assets can be assigned to employees or physical locations, supporting both static and mobile resources.
* Search and Query Capabilities
  + Resources can be searched by type, location, employee, or other metadata.
* Role-Based Permissions
  + Managers can modify data; employees can view assigned assets.
* Audit Logging
  + Tracks all changes for compliance and accountability.
* Security and Privacy
  + Includes authentication, authorization, and hidden sensitive fields.
* Categorization and Metadata
  + Resources are grouped by type and include critical fields like acquisition date, location, and formatted resource ID.
* Improved Accuracy and Accountability
  + Centralized data reduces errors and ensures consistent tracking.
* Enhanced Audit Readiness
  + Built-in logging and structured data support internal and external audits.
* Scalability and Flexibility
  + Modular design allows future expansion, including frontend integration or external deployment.
* Operational Efficiency
  + Streamlined asset management reduces manual tracking and improves resource utilization.

**7. Scope**

* In Scope:
  + REST API for resource management (no frontend required)
  + Ability to add, update, delete single or multiple resources
  + Ability to assign resources to employees or locations
  + Ability to search resources by type, employee, or location
  + Role-based permissions (manager vs employee)
  + Security-first design (authentication, authorization, data protection)
  + Audit logging for all changes
* Out of Scope:
  + Frontend UI
  + Employee Management

**8. Assumptions & Dependencies**

* Assumptions:
  + Users will have an authentication token verified through another service
  + Managers are responsible for maintaining resource records
  + Resource categories (e.g., laptops, servers, peripherals) will be predefined and maintained by system administrators
  + Resources can be assigned to an employee or a location but not both
* Dependencies:
  + IT support must be available for onboarding new users and troubleshooting access or data issues.
  + Authentication and authorization services must be operational.
  + Management must provide clear role definitions to distinguish between employee and manager permissions.

**9. Success Criteria**

* **Functional Criteria**
  + **Managers can add, update, and delete resources**:
    - Verified through CRUD operation tests.
    - Measured by error-free execution and correct database updates.
  + **Employees can query resource details**:
    - Queries return accurate and complete information.
  + **Resources can be tracked by employee or location**:
    - Ability to query by employee, location, resource, and date assigned
    - Reports match actual database records.
* **Security Criteria**
  + **Database is secure against common attacks**:
    - No vulnerabilities to SQL Injection or Cross-Site Scripting (XSS).
  + **Data Protection**:
    - Sensitive fields (e.g., employee assignments) encrypted at rest and in transit.
    - Access control enforced based on user roles.
* **Quality Assurance**
  + **Passes Whole Test Suite**:
    - All unit, integration, and end-to-end tests pass successfully.
    - Test coverage includes edge cases (invalid input, duplicate IDs).
  + **Performance Benchmarks**:
    - Handles expected load without degradation.
    - Response time stays within user expectation
* **Compliance & Audit**
  + Meets internal security standards and regulatory requirements.
  + Audit logs capture all changes with timestamps and user IDs.

**10. Stakeholders**

|  |  |  |
| --- | --- | --- |
| **User Proximity** | **Role** | **Description** |
| Primary | Managers | Manage asset lifecycle: adding, updating, deleting, and assigning resources. |
| Primary | Employee | Can view assets assigned to them or query general resource information. |
| Secondary | System Administrator | Handle deployment, access control, and security management. |
| Secondary | Developers (Team Blue) | Responsible for designing, implementing, and securing the API. |
| Secondary | Internal Audit Department | Primary requestor uses the system to verify inventory and audit changes. |
| Offstage | Legal Team | Provides guidelines for regulatory and legal compliance. |

**11. Risks & Mitigation**

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Likelihood | Impact | Mitigation |
| SQL Injection attack | Medium | Critical | Use parameterized queries,  Input validation, and regular  security scans. |
| Cross-Site Scripting (XSS) | Medium | High | Encode output, sanitize user  Input, implement CSP  Headers. |
| Unauthorized access to resource data | Low | Critical | Role-based access control,  Strong authentication,  Encryption at rest & transit |
| Performance degradation under load | Medium | Moderate | Load testing, caching strategies, scalable architecture. |
| Risk | Likelihood | Impact | Mitigation |

**12. Appendices**

* Use Case Diagram
* Use Case – SwaB (1.0) – word document