Systems Design

IniTech ITAM Revitalization Project

Brandon Trinkle

Dr. Rhonda Johnson

June 23, 2024

Table of Contents

[Background 3](#_Toc170036230)

[Problem Statement 3](#_Toc170036231)

[Technology Solution 3](#_Toc170036232)

[Process Maps 4](#_Toc170036233)

[Functions and Requirements Table 6](#_Toc170036234)

[User Interface 7](#_Toc170036235)

[Wireframe 1: Inventory Dashboard 7](#_Toc170036236)

[Description 7](#_Toc170036237)

[Wireframe 2: New Transaction Form 8](#_Toc170036238)

[Description 8](#_Toc170036239)

[Infrastructure Architecture 9](#_Toc170036240)

[Network Topology 9](#_Toc170036241)

[Information Architecture 10](#_Toc170036242)

[Entity Relationship Diagram 10](#_Toc170036243)

[Security and Privacy Architecture 11](#_Toc170036244)

[Programming 15](#_Toc170036245)

[References 20](#_Toc170036246)

# Background

## Problem Statement

  The inefficiencies in our current asset management system are multifaceted, stemming from an outdated framework incapable of accurately tracking the lifecycle of assets across various departments and geographical locations. This system's shortcomings have led to frequent instances of duplicated entries, asset misplacements, and a significant lack of accountability. Such issues compromise operational efficiency and directly impact the financial standing of the organization by inflating costs and diverting resources from critical areas of operation. The absence of real-time data and automated processes exacerbates these challenges, hindering effective decision-making and strategic asset planning.

## Technology Solution

We propose the development of an integrated asset management system designed to unify serialized asset tracking with advanced stockroom management capabilities. This new system will harness state-of-the-art technology to deliver real-time asset tracking and automated inventory management. By integrating advanced algorithms and machine learning models, the system will predict maintenance needs and optimize stock levels, thereby reducing operational downtime and minimizing costs. Enhanced security measures, including biometric access and encrypted data storage, will safeguard sensitive information, ensuring compliance with global data protection regulations and reinforcing system integrity against cyber threats.

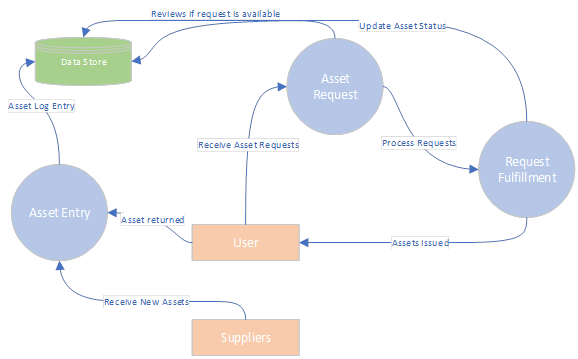
## Process Maps

 Business Function 1: Serialized Asset Tracking

Business Function 2: Stockroom Management



Business Function 3: Asset Entry and Request



Business Function 4: Security System



## Functions and Requirements Table

|  |  |
| --- | --- |
| **FUNCTION** | **REQUIREMENTS** |
| *Serialized Asset Tracking*: Maintain accurate and real-time records of all assets | * + Registration and tagging of each asset   + Real-time tracking of location, status, and condition   + Assignment tracking   + Maintenance scheduling   + Reporting features |
| *Stockroom Management*: Efficiently manage inventory of IT assets | * + Track multiple locations   + Real-time inventory tracking   + Secure access control   + Detailed logs of asset movements   + Reporting functionalities |
| *Asset Entry and Request*: Ensure seamless transition of assets from acquisition | * + Comprehensive interface for asset entry   + Real-time asset availability   + Automated alerts for new requests   + Detailed logs of asset entries and requests |
| *Security System*: Safeguard the asset management system against unauthorized access | * + Integration with existing SSO   + Two-factor authentication   + Role-based access controls   + Comprehensive logging   + Regular security audits |

# User Interface

## Wireframe 1: Inventory Dashboard

### Description

This screen provides an overview of current stock levels, recent transactions, and alerts for low stock. Users can search for specific items and view detailed information about each one. The dashboard includes sections for asset summaries, quick actions, and notification panels, offering a user-friendly interface to manage inventory efficiently.

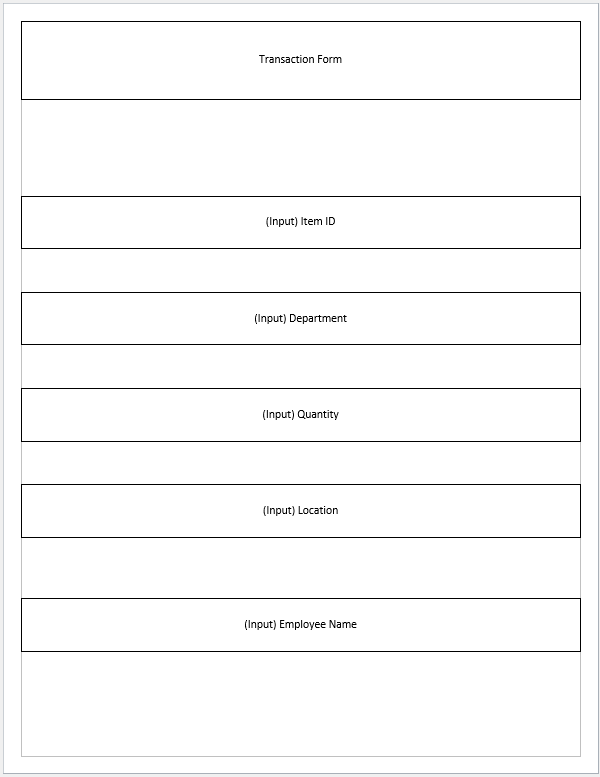
A white sheet of paper with black text

Description automatically generated

## Wireframe 2: New Transaction Form

### Description

This form allows users to add new inventory transactions, including stock additions, removals, and transfers. The form includes fields for item ID, quantity, date, and notes. It also features drop-down menus for selecting asset categories and locations, ensuring accurate and standardized data entry.



# Infrastructure Architecture

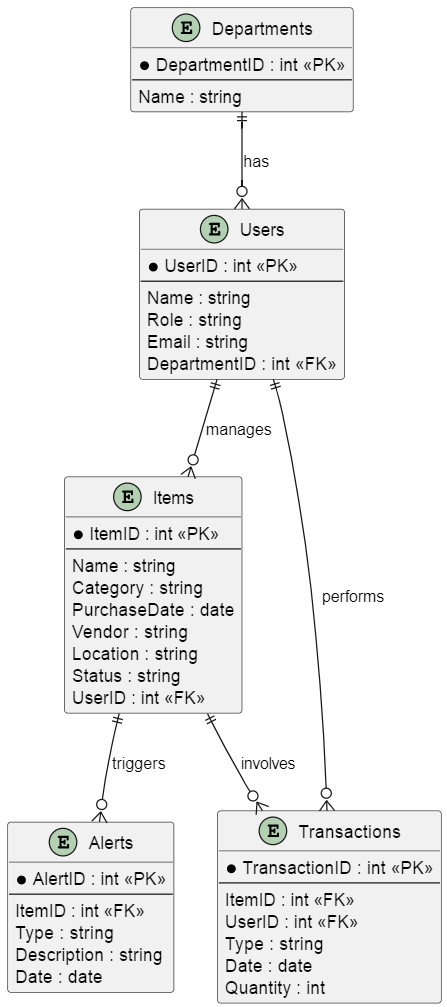
## Network Topology

A diagram of a server

Description automatically generated

# Information Architecture

## Entity Relationship Diagram



# Security and Privacy Architecture

The security and privacy architecture for the IT Asset Management (ITAM) system at IniTech Solutions is designed to provide robust protection against unauthorized access, data breaches, and other security threats. This comprehensive approach incorporates advanced authentication methods, strict authorization protocols, data encryption, and regular security audits to ensure the integrity, confidentiality, and availability of the system and its data.

**Authentication**

*Single Sign-On (SSO) Integration:*

* Purpose: Simplify user authentication by allowing users to log in once and gain access to multiple applications without needing to re-enter credentials.
* Implementation: The system will integrate with enterprise SSO solutions such as Azure Active Directory or Okta, using industry-standard protocols like SAML (Security Assertion Markup Language) or OAuth.
* Benefits: SSO reduces password fatigue, minimizes the risk of password-related security incidents, and improves user productivity by streamlining the login process (Dubin, 2018).

*Two-Factor Authentication (2FA):*

* Purpose: Enhance security by requiring two forms of verification—something the user knows (password) and something the user has (physical token or mobile app-generated code).
* Implementation: Users will need to enter their password and a one-time code sent to their mobile device or generated by an authenticator app such as Google Authenticator or Microsoft Authenticator.
* Benefits: 2FA significantly reduces the risk of unauthorized access due to stolen or guessed passwords by adding an extra layer of security.

**Authorization**

*Role-Based Access Control (RBAC):*

* Purpose: Ensure that users have access only to the resources necessary for their role, reducing the risk of accidental or malicious access to sensitive data.
* Implementation: The system will define various roles (e.g., Admin, Manager, Employee) and assign permissions based on these roles. Permissions are managed centrally and can be adjusted as needed.
* Benefits: RBAC simplifies permission management and enhances security by enforcing the principle of least privilege, ensuring users only access what they need for their job functions.

*Configurable Permissions:*

* Purpose: Adapt to the changing needs of the organization by allowing administrators to configure permissions dynamically.
* Implementation: Permissions can be adjusted through an administrative interface, ensuring flexibility and responsiveness to organizational changes.
* Benefits: This flexibility ensures the system remains aligned with evolving organizational policies and requirements, enhancing security and operational efficiency.

**Privacy**

*Data Encryption:*

* Purpose: Protect data from unauthorized access both at rest and in transit.
* Implementation:
  + Data at Rest: Encrypt all sensitive data stored in the database using AES-256 encryption. (Rimkienė, 2022)
  + Data in Transit: Encrypt data transmitted between clients and servers using TLS (Transport Layer Security).
* Benefits: Encryption ensures that even if data is intercepted or accessed without authorization, it remains unreadable and secure.

*Comprehensive Logging:*

* Purpose: Enable auditing, monitoring, and incident response by maintaining detailed logs of user activities and security events.
* Implementation:
  + Activity Logs: Record all user activities, such as logins, data access, and administrative actions.
  + Security Logs: Record security-related events, such as failed login attempts and changes to security settings.
* Benefits: Logging helps track user actions, identify potential security breaches, and ensure compliance with regulatory requirements.

*Regular Security Audits:*

* Purpose: Ensure the effectiveness and up-to-dateness of security measures through regular reviews and assessments.
* Implementation: The system will generate detailed security audit reports, which will be reviewed by the security team to identify vulnerabilities and areas for improvement.
* Benefits: Continuous monitoring and improvement of security measures enhance the system's resilience against threats and ensure ongoing compliance with security standards.

*Best Practices for Security:*

* Least Privilege: Ensure users have the minimum level of access necessary for their role.
* Segregation of Duties: Separate critical functions among different users to reduce the risk of fraud or error.
* Incident Response Plan: Develop and maintain a plan to respond to security incidents, including data breaches and system compromises.
* User Training: Regularly train users on security best practices, such as recognizing phishing attempts and creating strong passwords.

# Programming

The development of the ITAM system will utilize a variety of programming languages, frameworks, and tools chosen for their ability to meet the system’s requirements in terms of functionality, scalability, and security. Choosing the right tools and technologies for programming is critical to building a robust and scalable ITAM system. This section provides a detailed overview of the development tools, programming languages, and best practices that will be employed to ensure efficient and secure software development. Our goal is to leverage modern technologies to create a reliable and maintainable system that meets all functional requirements.

**Development Tools**

*Visual Studio Code:*

* Purpose: A powerful and versatile code editor that supports various programming languages and frameworks.
* Features: Extensions for language support, debugging tools, integrated Git control, and customizable workspaces.
* Benefits: Enhances productivity through features like IntelliSense, code navigation, and integrated terminal.

*Git:*

* Purpose: A version control system that tracks changes in source code during software development.
* Features: Branching and merging, commit history, collaboration tools, and integration with platforms like GitHub and GitLab.
* Benefits: Facilitates collaborative development, enables rollback of changes, and supports multiple development workflows.

**Programming Languages**

*Python:*

* Purpose: Used for backend development, data processing, and scripting.
* Features: Readable syntax, extensive libraries (e.g., Django for web development, Pandas for data analysis), and strong community support.
* Benefits: Python’s simplicity and robustness make it ideal for developing scalable and maintainable backend services.

*JavaScript (React):*

* Purpose: Used for frontend development to create dynamic and responsive user interfaces.
* Features: Component-based architecture, virtual DOM for improved performance, and a rich ecosystem of libraries and tools.
* Benefits: React allows for the efficient management of UI state and lifecycle, resulting in a smooth and interactive user experience.

*Node.js:*

* Purpose: Used for backend development to handle concurrent processes and real-time data processing.
* Features: Non-blocking, event-driven architecture, support for asynchronous programming, and a large library of modules (npm).
* Benefits: Node.js’s architecture makes it suitable for real-time applications and services that require high performance and scalability.

**COTS Platforms**

*Oracle Database:*

* Purpose: A reliable and scalable database management system.
* Features: Advanced data handling capabilities, robust security features, support for complex queries, and high availability.
* Benefits: Ensures data consistency and integrity, handles large volumes of data efficiently, and provides extensive support for enterprise applications.

*Microsoft Power BI:*

* Purpose: A business analytics tool for visualizing and sharing insights from data.
* Features: Interactive dashboards, real-time data updates, advanced analytics, and integration with various data sources.
* Benefits: Enables the creation of dynamic and insightful reports, supports data-driven decision-making, and improves visibility into key performance indicators.

**Best Practices for Programming**

*Modular Design:*

* Purpose: Breaks down the system into manageable, self-contained modules.
* Implementation: Each module handles a specific aspect of the system, such as user management, asset tracking, or reporting.
* Benefits: Enhances maintainability, allows for parallel development, and makes it easier to test and debug individual components.

*Code Reviews:*

* Purpose: Ensure code quality and adherence to coding standards.
* Implementation: Regular peer reviews of code changes before merging into the main branch.
* Benefits: Identifies potential issues early, promotes knowledge sharing, and maintains code consistency.

*Continuous Integration/Continuous Deployment (CI/CD):*

* Purpose: Automates the integration and deployment of code changes.
* Implementation: Use tools like Jenkins or GitHub Actions to automate build, test, and deployment processes.
* Benefits: Reduces the time to release new features, ensures that code changes are thoroughly tested, and improves overall software quality.

*Automated Testing:*

* Purpose: Ensures that the system works as expected and prevents regressions.
* Implementation: Use frameworks like pytest for Python and Jest for JavaScript to write unit tests, integration tests, and end-to-end tests.
* Benefits: Increases confidence in code changes, reduces the likelihood of bugs, and speeds up the development process by catching issues early.

*Security Practices:*

* Secure Coding: Follow best practices for secure coding, such as input validation, output encoding, and avoiding the use of deprecated functions.
* Dependency Management: Regularly update dependencies to patch known vulnerabilities and use tools like Dependabot to automate this process.
* Static Analysis: Use tools like SonarQube to analyze code for security vulnerabilities and code quality issues.

*Documentation:*

* Purpose: Provides clear and comprehensive information about the system.
* Implementation: Document code, APIs, and system architecture using tools like Sphinx for Python and JSDoc for JavaScript.
* Benefits: Facilitates onboarding of new developers, supports ongoing maintenance, and ensures that stakeholders understand the system’s capabilities and limitations.

# References

Dubin, J. (2018, Feb 28). *Single sign-on best practices: How can enterprises get SSO right?* Retrieved from Tech Target: https://www.techtarget.com/searchsecurity/answer/Best-practices-for-deploying-enterprise-single-sign-on-SSO

Rimkienė, R. (2022, Aug 29). *What is AES encryption and how does it work?* Retrieved from Cybernews: https://cybernews.com/resources/what-is-aes-encryption/