This comprehensive guide details the system's three-tier architecture, elaborates on API endpoints and usage, explains the database schema and data dictionary, and lists third-party libraries and dependencies utilized by the Python client, providing developers and engineers a clear understanding of the system's design and technical components.

Technical Documentation

IFS325 Individual Assignment Documentation

Alyssa Krishna - 4308998

Contents

System Architecture and Design

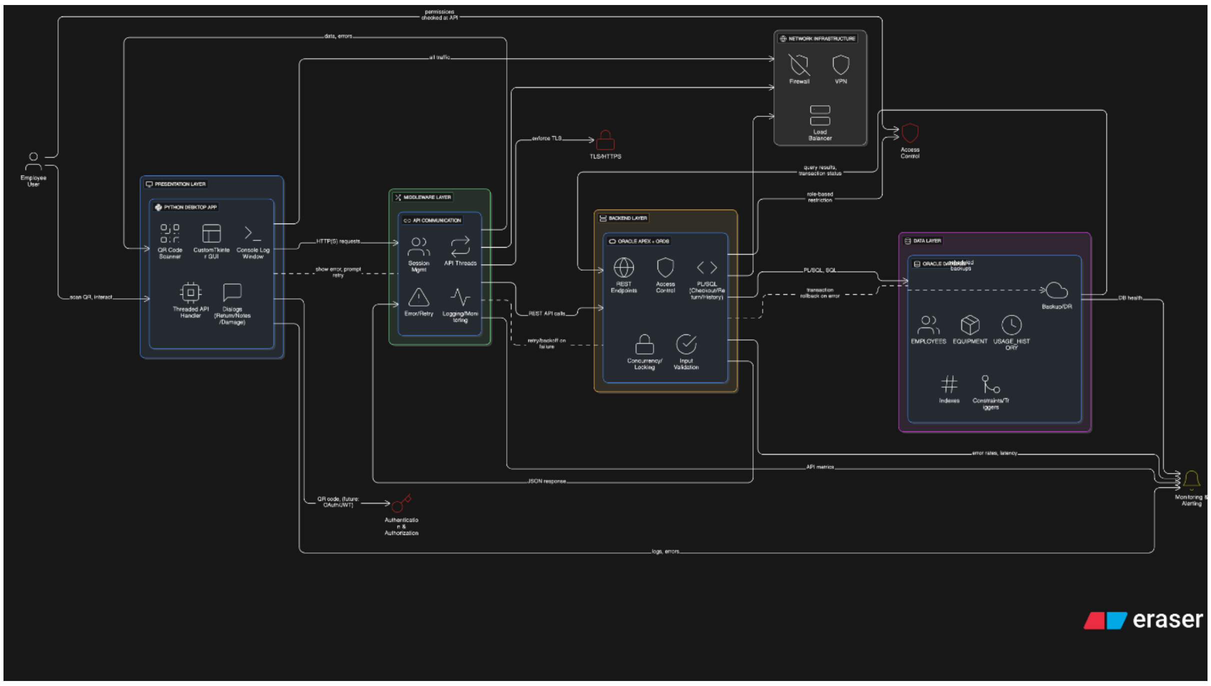
Architectural Overview

The system employs a three-tier architecture to separate concerns and ensure scalability.

- Presentation Layer:
    - Oracle APEX Application: Serves as the primary administrative interface. This web-based application provides comprehensive functionality for inventory management, reporting, and system administration.
    - Python GUI Application: A dedicated client application for employees, built using Tkinter. Its purpose is to provide a streamlined interface for viewing personal equipment history, and processing checkouts and returns.
- Application Layer:
    - Oracle REST Data Services (ORDS): Acts as the middleware, exposing a secure set of RESTful APIs. This layer facilitates all programmatic interactions between the Python client and the database, enforcing business logic and security protocols.
- Data Layer:
    - Oracle Database 21c: The single source of truth, housing all application data, including inventory stock, employee details, transaction history, and system logs.

System Architecture

Figure 1: System Architecture Diagram



Description:

- **Client Tier**: The Python Employee Client provides a dedicated interface for end-users (employees) to perform self-service returns and view history.
- **Application Tier**: The Oracle APEX Server hosts the administrative web application and the ORDS (Oracle REST Data Services) middleware. ORDS exposes RESTful APIs that facilitate all communication between the Python client and the database, encapsulating business logic.
- **Data Tier**: The Oracle Database serves as the single source of truth, persistently storing all application data.

Note: Please see attached PNGs in zipped folder for better clarity.

1.2 Data Flow

For a return operation:

1. The Python client sends an authenticated POST request to the /equipment/return/ ORDS endpoint.
2. ORDS validates the request and executes a PL/SQL API procedure.
3. The procedure updates the STATUS in NEXORA_USAGE_HISTORY and increments the QUANTITY in NEXORA_INVENTORY within a transaction.
4. A success or error JSON response is routed back through ORDS to the Python client.

5. The client displays the result to the user.

API Documentation

The system's functionality is exposed via a REST API, enabling interoperability. All requests and responses are in JSON format.

- Base URL: https://oracleapex.com/ords/nexora/api/

Endpoint	Method	Parameters (Body)	Description	Success Response (200)
/checkout	POST	{ "item_id": "RPI4-8GB", "employee_id": 1567, "qr_code": "EMP1567", "is_damaged": "N", "checkout_notes": "Test from Postman" } **Headers:** Key: content-type Value: application/json	Initiates a check-out transaction. Decrements inventory.	{"success": "Equipment checked-out successfully", "booking_id": int}
/return	POST	{ "booking_id": 101, "employee_id": 1567, "qr_code": "EMP1567",	Finalizes a check-out transaction. Updates item status and inventory. Generates QR code	{"success": "Equipment returned successfully"}

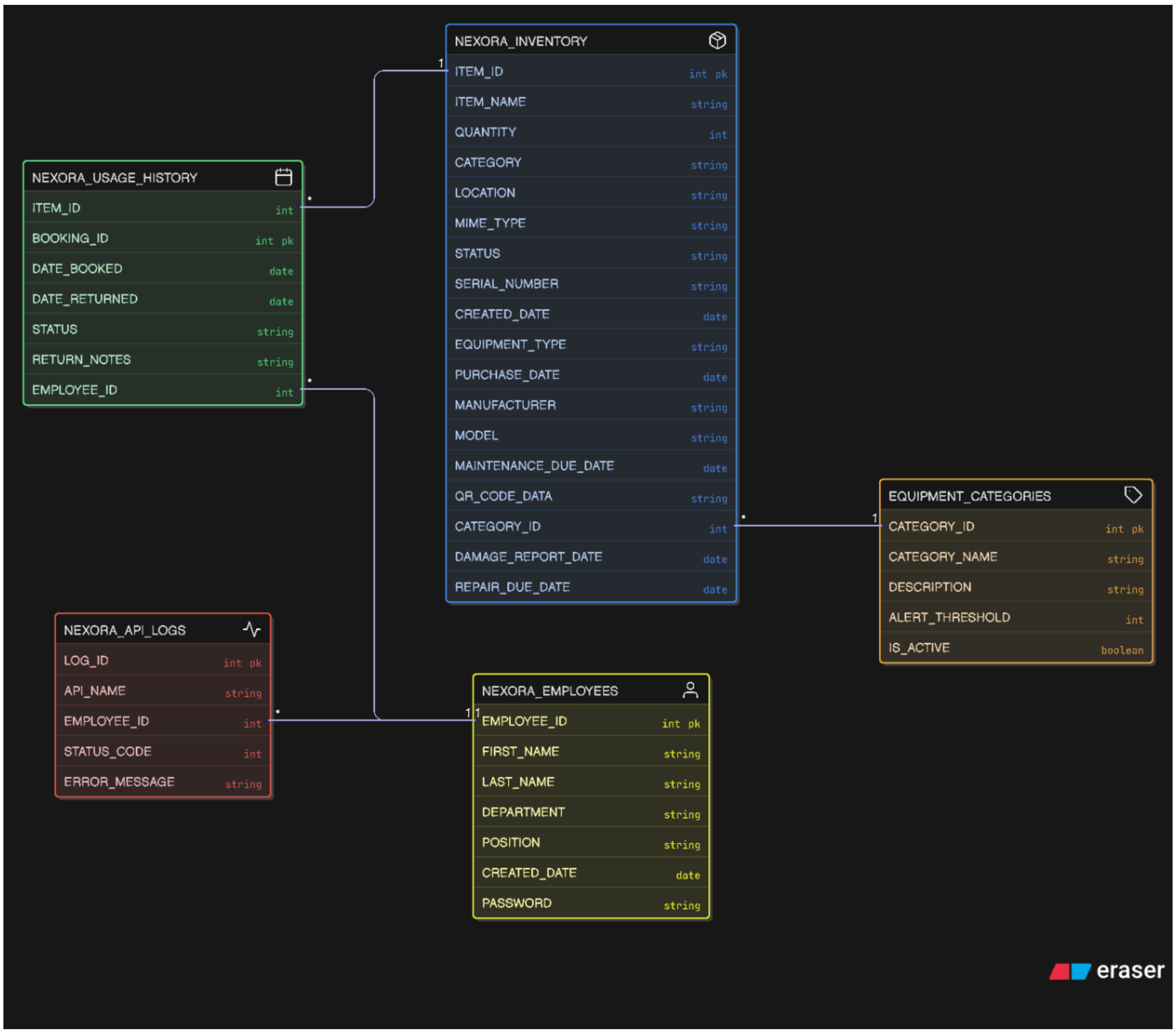
		"return_notes": "Test return from Postman", "is_damaged": "N" } **Headers:** Key: content-type Value: application/json	that Oracle APEX can understand.	
/history/{emp_id}	GET	employee_id (URL Parameter)	Retrieves historical and active transactions for a given employee.	{"items": [{booking_data}, ...]}
/inventory	GET	-	Fetches the current state of all inventory items.	{"items": [{item_data}, ...]}
/employees	GET	-	Fetches all the employee data.	{"items": [{employee_data}, ...]}

Database Schema and Data Dictionary

Entity-Relationship Diagram (ERD)

The core data model is represented by the following Entity-Relationship Diagram:

Figure 2: Simplified Entity-Relationship Diagram (ERD)



Note: Please see attached PNGs in zipped folder for better clarity.

Data Dictionary

The relational schema is built upon the provided tables, with key functional dependencies outlined below.

Core Tables:

- **NEXORA_INVENTORY**
  - ITEM_ID (PK): VARCHAR2 - Unique identifier for each equipment item (e.g., RPI4-8GB). Serves as the primary asset tag.
  - QUANTITY: NUMBER - Current available stock level. Validated against ALERT_THRESHOLD.

    - STATUS: VARCHAR2 - Operational state ('Available', 'In Use', 'Maintenance', 'Damaged').
    - CATEGORY_ID (FK): NUMBER - References EQUIPMENT_CATEGORIES(CATEGORY_ID).
    - QR_CODE_DATA: VARCHAR2 - Typically stores the ITEM_ID for generating physical QR asset tags.
- **NEXORA_USAGE_HISTORY**
    - BOOKING_ID (PK): NUMBER - System-generated unique transaction ID.
    - ITEM_ID (FK): VARCHAR2 - References NEXORA_INVENTORY(ITEM_ID).
    - EMPLOYEE_ID (FK): NUMBER - References NEXORA_EMPLOYEES(EMPLOYEE_ID).
    - STATUS: VARCHAR2 - Current transaction status ('CHECKED_OUT', 'RETURNED').
    - DATE_BOOKED, DATE_RETURNED: DATE - Timestamps for transaction lifecycle.
- **NEXORA_EMPLOYEES:** Contains employee credentials and details. Used for authentication and auditing.
- **EQUIPMENT_CATEGORIES:** Defines categories (e.g., 'Sensors', 'Computing') and their configurable low-stock thresholds (ALERT_THRESHOLD).
- **NEXORA_API_LOGS:** Audits all API interactions for troubleshooting and security monitoring.

Third-party Libraries and Dependencies

- Python Application:
    - requests: HTTP library for API communication.
    - qrcode: Library for generating QR code images for new inventory items.
    - opencv-python & pillow: Computer vision and image processing libraries for capturing and decoding QR codes via the employee application's camera interface.

- tkinter: Standard GUI library for building the application interface.

Ethical Considerations

1. **Employee Privacy and Data Protection:**
   The system collects and stores sensitive employee information (such as employee IDs, check-in/out history, and potentially location tracking via equipment usage). It is ethical and legally necessary to ensure that this personal data is collected, stored, and accessed in compliance with applicable privacy regulations (e.g., GDPR, POPIA). Access should be restricted to authorized personnel only, and employees should be informed about what data is collected and how it is used.
2. **Security of Authentication Methods:**
   Using QR codes for employee authentication necessitates strong security measures to prevent unauthorized access or cloning of QR badges. The system must ensure that QR codes cannot be easily duplicated or misused, and that there are fallback or additional authentication layers to verify user identity.
3. **Data Accuracy and Integrity:**
   Ethical responsibility includes maintaining accurate and up-to-date records so that equipment allocation and returns are reliably tracked. False or outdated data could lead to unfair penalization of employees or loss of assets.
4. **Transparency and Accountability:**
   Employees should have transparency regarding their equipment usage data and the purposes for which it is collected. There should be accountability for data handling practices and audit logs maintained to monitor system use and potential misuse.
5. **Minimization of Monitoring:**
   The system should be designed to collect only the data necessary for operational purposes, avoiding excessive monitoring of employees beyond the scope of equipment tracking, respecting their workplace autonomy.

Incorporating these ethical principles helps ensure that the system respects user rights, maintains trust, and complies with legal and moral obligations.

References

1. *Oracle APEX BD. (2024b, September 26). Oracle APEX PL/SQL Tutorial: Building Dynamic Processes and Packages [Video]. YouTube. https://www.youtube.com/watch?v=HJ1kibsXTV0*