System Integration Final Project Presentation

GROUP 3

Sampoerna University

Overview

Overall Goal

"Seamless integration of inventory, CRM, and financial systems to enhance operational efficiency and data-driven decision-making."

System Requirement Analysis

System Requirement Analysis

Business Needs & Integration Goals

Business Needs

- Real-time integration of core systems such as inventory, CRM, and finance.
- Scalable and future-proof solution.
- Improved interdepartmental communication and automation.

Integration Goals

- Streamlined data flow.
- Reduction of manual processes.
- Enhanced visibility across operations.

System Requirement Analysis

Business Needs & Integration Goals

1 Inventory

The inventory management system will play a crucial role in tracking product quantities, stock levels, and supplier information. It will ensure real-time monitoring of inventory movement, including the entry and exit of production supplies, and the status of organizational assets. By automating these processes, the system will help minimize human error, streamline supply chain operations, and improve overall inventory accuracy.

2 CRM

The CRM system will centralize customer data, enabling the management of interactions and improving customer service. It will assist in overseeing marketing campaigns, tracking sales activities, and supporting customer support operations. Integration with other systems will ensure that customer data is synchronized across departments, providing accurate insights into customer behavior and enhancing the overall customer experience.

3 Financial System

The financial system will automate accounting processes, payroll management, and financial reporting. This system will streamline billing and invoicing, linking them with customer data from the CRM system for precision. Additionally, it will integrate with inventory management to reflect real-time inventory changes in financial records, ensuring compliance and accuracy in financial operations.

Key Performance Metrics

Data Accuracy

Reduce errors by 90%.

Automation

Achieve 80-90% data exchange automation.

Operational Efficiency

Decrease update delays by 50%.

Scalability

Cloud-based architecture for growth.

System Architecture Design

System Boundary

Subsystems & Components

1 Inventory

- Inventory DB
- Supply Management
- Order Management
- Supply Chain Log

2 CRM

- Transaction Record
- Customer DB
- Customer Support
- Marketing Automation

3 Finance

- Accounting
- Payroll
- Budgeting
- Report Generator

Interface, Communication & Data Exchange

1 Interface

Use API Endpoints & Connections for:

- Inventory-CRM (Enable Inventory Update)
- CRM-Finance (Sync Customer & Billing Data)
- Inventory-Finance (Auto Log Transaction when Stock is Adjusted)

2 Communication Protocols

- RESTful APIs (Real-Time Data Sharing)
- Event-driven Architecture (Asynchronous Data Flows & Reduce Latency)
- Webhooks (Trigger Updates in Response to Events)
- Message Queueing (Reliable Delivery on Hi-Volume Hi-Freq Data)

3 Data Exchange Mechanism

- Data Transformation & Normalization (Ensure consistency)
- Extract, Transform, Load (ETL) (Initial Data Migration)
- Real Time Updates (Sync Key Data)
- Scheduled Updates (Reduce System Load)

Integration Strategy & Plan

First Steps

1 Define system architecture and integration requirements

- Identify key endpoints for data exchange
- Define real-time vs scheduled syncs for data transfer
- Establish integration priorities
- Document API requirements, data structures and protocols

2 Implement Data Transformation and Normalization

- Use a middleware tool to standardize data schemes like apache
- Validate data formats to prevent discrepancies across systems
- Create a transformation layer for legacy system data

Integration

1 Inventory system

1. Connect inventory database for the items list, stock level, new stock, and orders.
2. Integrate supply management, this includes automating the recording of new stock entries in the database.
3. Link order management.
4. Add supply chain log to check when stock is replenished

Middleware

2 CRM

1. Connect transaction records to log sales data from the order management system.
2. Build a customer database.
3. Integrate customer support.
4. Enable marketing automation.

Middleware

3 Financial System

1. Connect accounting system to monitor expenses from supply management and payroll.
2. Integrate payroll to automated payroll processing.
3. Enable budgeting to be sync with the inventory database to forecast supply costs.
4. Add a financial report generator.

Concluding Steps

Conduct Incremental Integration Module Testing

- Start with a single subsystem pair (i.e., Inventory-CRM)
- Perform unit testing for each interface (i.e., Stock Sync API)
- Validate Data Flow Accuracy and Error Handling

Legacy Systems

- Middleware platform will be used between the systems to act as the bridge between the systems with newer and older technology
- Transition legacy system to modern platforms over time, ensuring backward compatibility during migration
- Break down business functions into discrete services that can be reused across different applications through Service-oriented architecture (SOA)
- Developed APIs that interface with legacy systems, translating between old protocols and modern RESTful endpoints an API,

Testing and Validation Plan

Testing Plan

1. **Test individual modules like an API to synchronize the inventory stock for customer data updates in isolation**

2. **Integration Testing:**

   - Verify data flow between connected subsystems (e.g., Inventory to CRM).
   - Test middleware performance and data transformation.

3. **Perform System Testing by assessing the entire integrated system for expected behavior. Then validate business processes (e.g., automated stock updates).**

4. **Performance Testing:**

   - Measure response times of APIs and data synchronization.
   - Load and stress testing for scalability.

Testing Plan

5 Verify that new changes do not disrupt existing functionality through regression testing

6 Test for vulnerabilities in data transfer protocols and interfaces

7 Validate that the system meets end-user requirements through User Acceptance Testing (UAT)

Test Cases: Dataflow

1

Data Exchange between Inventory and CRM system

- Input: Add new stock to the inventory system.
- Expected Result: CRM reflects updated inventory in real time.

2

Financial System Integration

- Input: Record a sale in CRM.
- Expected Result: Sale data synchronized with the Financial System for Revenue Tracking.

3

Middleware Transformation

- Input: Legacy system sends outdated data format.
- Expected Result: Middleware standardized data format and transmits to modern systems.

1. Test Cases: System Performance

1

API Load Test

- Input: 1,000 concurrent users accessing inventory data.
- Expected Result: The system maintains a response time of < 2 seconds.

2

Stress Test for Scaling

- Input: Increase input to 10x normal load.
- Expected Result: The system processes data without crashing.

1. Test Cases: Interoperability

1

Legacy and Modern System Communication

- Input: Trigger a financial report request from the legacy system.
- Expected Result: Middleware bridges communication and generates an accurate report.

2

Event-Driven Communication via Webhooks

- Input: Inventory level falls below threshold.
- Expected Result: Notification triggers the supply management system.

Functional Requirements

Data synchronization and accuracy

Automated workflows

Real-time and scheduled data transfer

Non-Functional

Performance

Ensuring that API response times are quick and the system is scalable to handle increased loads.

Reliability

Error handling in middleware and APIs when failure occurs.

Security

Protocol compliance and the system's ability to withstand attacks, protecting sensitive data during integration

Compatibility

Ensures smooth communication between legacy and modern systems, allowing seamless data transfer and functionality.

Usability

Creating intuitive interfaces to ease the use for end-users, enhancing the overall user experience.

Available Tools for Testing

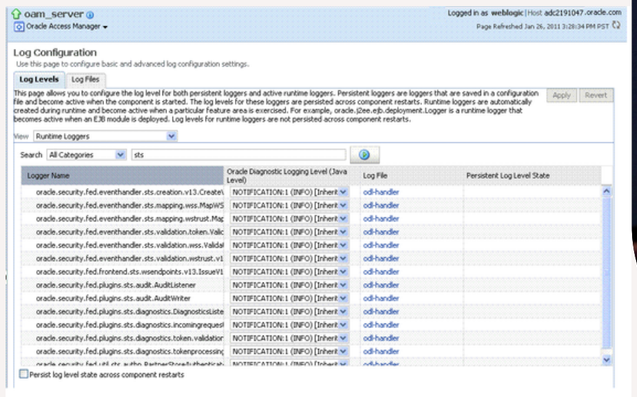
Postman for API Testing

POSTMAN

Selenium for UI Testing

Selenium

Middleware log files for Error Detection



Thank You