Business Requirements Document Generic Third-Party Integration Module

Prepared for: Ashish (CTO)

Date: June 11, 2025

In-House Development Team

Contents

1 Introduction

1.1 Purpose

This Business Requirements Document (BRD) outlines the requirements for developing a generic third-party integration module to facilitate seamless data exchange between the Loan Management System (LMS), Loan Origination System (LOS), and external systems, including core banking systems and co-lending partner platforms. The module aims to standardize integrations, reduce development effort for future third-party systems, and ensure compliance with regulatory guidelines.

1.2 Scope

The module will:

- Enable data exchange with third-party systems (initially two partner companies and a co-lending module).
- Support flexible endpoint configurations, authentication methods, and data formats (JSON, CSV).
- Handle asynchronous processing and callback mechanisms.
- Operate as a containerized microservice on AWS.
- Be developed in Golang with a 1-month timeline for Phase 1 (API-based).
- Include basic reporting and error handling with nightly retries for failed transactions.

1.3 Stakeholders

- **Primary Stakeholder**: Ashish (CTO)
- **Development Team**: In-house developers (including interns)
- **End Users**: Internal LMS/LOS teams and third-party partners

2 Business Objectives

- **Primary Objective**: Streamline data exchange with third-party systems to support current integrations (two partner companies) and future integrations (co-lending module and additional systems).
- **Secondary Objectives**:
  - Reduce integration development time for new third-party systems.
  - Ensure compliance with Reserve Bank of India (RBI) lending practices.
  - Provide scalability for handling up to 1,000 transactions per day per partner.

3 Functional Requirements

3.1 Data Exchange

- **Data Types**: Customer details, loan details, collateral details, documents, disbursement and payments, loan servicing requests, co-applicant, and guarantor details.
- **Data Formats**: JSON and CSV, with flexibility to support additional formats in the future.
- **Data Transformation**: Map internal LMS/LOS data to third-party-specific formats using a configuration-driven approach.
- **Transaction Volume**: Support up to 1,000 transactions per day per third-party partner (2,000 total for initial partners).

3.2 Endpoint and Authentication Management

- **Endpoint Configuration**: Store and manage third-party API endpoints, headers, and payloads in a configuration database.
- **Authentication**: Support multiple authentication methods (e.g., OAuth2, API keys, JWT, basic auth) based on partner requirements.
- **Secret Management**: Securely manage secrets (e.g., API keys, tokens) using AWS Secrets Manager.

3.3 Callback Handling

- **Mechanism**: Implement webhooks for real-time callbacks from third-party systems. Support polling as a fallback for systems without webhook capabilities.
- **Processing**: Handle asynchronous responses and update internal systems accordingly.

3.4 Error Handling and Retries

- **Error Handling**: Log all failed transactions with detailed error messages.
- **Retry Policy**: Automatically retry failed transactions during a nightly batch process.
- **Reporting**: Generate a daily failure report for review by the operations team.

3.5 Asynchronous Processing

- **Requirement**: Support asynchronous data processing to handle high-volume transactions efficiently.
- **Implementation**: Use a message queue (e.g., RabbitMQ or AWS SQS) for task queuing and processing.

3.6 API-Based Interface (Phase 1)

- **Functionality**: Expose RESTful APIs to initiate data pushes, configure integrations, and retrieve transaction statuses.
- **Documentation**: Provide OpenAPI-compliant documentation for all endpoints.

3.7 User Interface (Phase 2)

- **Future Requirement**: Develop a web-based UI for configuring third-party integrations (e.g., endpoints, mappings).
- **Scope Exclusion**: UI development is out of scope for Phase 1 (1-month timeline).

4 Non-Functional Requirements

4.1 Performance

- **Response Time**: Standard API response time (<2 seconds for 95% of requests).
- **Uptime**: 99.9% availability.
- **Scalability**: Handle up to 2,000 transactions per day initially, with the ability to scale for future partners.

4.2 Security

- **Data Security**: Encrypt sensitive data in transit (HTTPS/TLS) and at rest (AES-256). Implement role-based access control (RBAC) for API access.
- **Audit Logging**: Log all API requests, responses, and errors for compliance with RBI guidelines.
- **Secret Management**: Use AWS Secrets Manager for storing authentication credentials.

4.3 Compliance

- **Regulatory Requirements**: Adhere to RBI lending practices, including data privacy and auditability.
- **Data Retention**: Retain transaction logs for a minimum of 7 years (per RBI guidelines).

4.4 Deployment

- **Environment**: Deploy as a containerized microservice on AWS (using Docker and ECS/EKS).
- **CI/CD**: Implement continuous integration and deployment pipelines.

4.5 Monitoring and Reporting

- **Monitoring**: Monitor API uptime, transaction success/failure rates, and system performance using AWS CloudWatch.

- **Reporting**: Provide basic reports on transaction volumes, success rates, and failures. Generate daily failure reports for retry analysis.

5 Technical Requirements

- **Programming Language**: Golang for backend development.
- **Database**: PostgreSQL for storing configuration data (endpoints, mappings).
- **Message Queue**: RabbitMQ or AWS SQS for asynchronous processing.
- **Cloud Platform**: AWS for hosting and secret management.
- **Containerization**: Docker for packaging and deployment.
- **API Framework**: Use `gin` or `echo` for building RESTful APIs in Golang.

6 Constraints

- **Timeline**: Complete Phase 1 (API-based module) within 1 month.
- **Budget**: In-house development with no external vendor costs.
- **Team**: Includes intern developers, requiring simplicity in design and implementation.

7 Assumptions

- Third-party systems provide API documentation and sandbox environments for testing.
- Initial integrations focus on two partner companies, with co-lending module requirements to be defined later.
- Standard performance and security measures (e.g., <2s response time, AES-256 encryption) are sufficient.
- Internal LMS/LOS systems do not require direct integration in Phase 1.

8 Risks and Mitigation

- **Risk**: Limited experience of intern developers may delay delivery.
  - **Mitigation**: Provide clear documentation, use simple frameworks, and assign senior oversight.
- **Risk**: Third-party API changes may break integrations.
  - **Mitigation**: Implement versioned configurations and robust error handling.
- **Risk**: Tight 1-month timeline may limit testing.
  - **Mitigation**: Prioritize unit testing and use sandbox environments for integration testing.

9 Timeline and Milestones

- **Week 1**: Requirements finalization and design (architecture, database schema).
- **Week 2**: Develop configuration layer, data transformation, and API client.
- **Week 3**: Implement webhook/polling, error handling, and retry logic.
- **Week 4**: Testing, deployment on AWS, and basic reporting setup.

10 Approval

- **Stakeholder**: Ashish (CTO)
- **Approval Process**: Review and sign-off on BRD before development begins.