Legacy Banking System Architectural Documentation

Version 1.0

System Architect Team

Prepared for:

Banking Solutions Inc.

Contents

Chapter 1

Introduction

1.1 Mission

The mission of the Legacy Banking System is to provide a secure, reliable, and efficient platform for managing financial transactions and customer data. It aims to support millions of customers and process billions of transactions annually while complying with evolving regulatory requirements.

1.2 Business Goals

- Ensure high availability and reliability of banking services.
- Maintain compliance with financial regulations.
- Enhance customer experience through efficient service delivery.
- Support scalability to handle increasing transaction volumes.

1.3 Constraints

- Legacy technology stack (e.g., COBOL, IBM DB2) limits modernization efforts.
- Monolithic architecture restricts scalability and flexibility.
- High operational costs due to maintenance of legacy systems.

1.4 Stakeholders

- Banking Solutions Inc. (System Owner)
- Customers (End Users)
- Regulatory Authorities
- System Architects and Developers
- IT Operations and Maintenance Teams

Chapter 2

System Overview

2.1 Context Delineation

The Legacy Banking System operates in a highly regulated financial environment. It interacts with external payment systems, credit bureaus, and regulatory authorities. The system is designed to handle core banking functionalities, including account management, transaction processing, and loan management.

2.2 Key Domain Entities

- Customer: Represents an individual or entity with one or more accounts.
- Account: Represents a financial account (e.g., savings, checking, fixed deposit).
- Transaction: Represents a financial operation (e.g., deposit, withdrawal, transfer).
- Loan: Represents a financial product provided to customers.
- Report: Represents financial or regulatory reports generated by the system.

2.3 System Components Classification

2.3.1 Internal Components

Internal components are modules and subsystems that operate within the bank's infrastructure and are directly controlled by the banking organization:

- **Core Banking Modules**:
  - Account Management Module: Manages customer accounts and their associated data.
  - Transaction Processing Module: Handles all financial transactions within the system.
  - Loan Management Module: Processes loan applications, disbursements, and repayments.

  - Interest Calculation Module: Computes interest for deposit and loan accounts.
  - Fixed Deposit Module: Manages term deposits with fixed interest rates.
  - Foreign Exchange Module: Handles currency conversion and exchange rate management.
  - Standing Order Module: Manages recurring payments and transfers.
- **Support Systems**:
  - Customer Service Module: Provides interfaces for customer support operations.
  - Reporting Module: Generates internal and regulatory reports.
  - Data Archiving Module: Manages historical data storage and retrieval.
  - Audit and Compliance Module: Ensures regulatory compliance and maintains audit trails.
  - Document Management Module: Stores and retrieves customer documents and contracts.
  - Customer Relationship Management (CRM) Module: Manages customer interactions.
  - Branch Operations Module: Supports in-branch banking operations.
- **Security Infrastructure**:
  - Authentication and Authorization Module: Controls system access and permissions.
  - Fraud Detection Module: Monitors and flags suspicious activities.
  - Backup and Recovery Module: Ensures data integrity and business continuity.
  - Encryption Module: Manages data encryption for sensitive information.
  - Digital Signature Module: Handles electronic signatures for documents.
  - Security Incident Management Module: Tracks and responds to security breaches.
- **Technical Infrastructure**:
  - Database Management Systems: IBM DB2 and Oracle installations.
  - Application Servers: Hosts for Java and .NET applications.
  - Mainframe Systems: Hosts for COBOL applications.
  - Performance Monitoring Module: Tracks system health and performance.
  - API Gateway Module: Manages API requests and routing.
  - Enterprise Service Bus (ESB): Facilitates service-oriented architecture.
  - Load Balancing Module: Distributes workload across servers.
  - Notification Service Module: Sends alerts and notifications to customers.

2.3.2 External Components

External components are systems, services, or entities that exist outside the direct control of the banking organization but interface with the Legacy Banking System:

- **Regulatory Systems**:
  - Central Bank Reporting Systems: For mandatory financial reporting.
  - Anti-Money Laundering (AML) Monitoring Systems: For compliance with financial crime regulations.
  - Know Your Customer (KYC) Verification Services: For customer identity verification.
  - Financial Intelligence Unit (FIU) Systems: For suspicious transaction reporting.
  - Securities and Exchange Commission (SEC) Systems: For investment reporting.
- **Financial Network Interfaces**:
  - SWIFT Network: For international wire transfers.
  - National Payment Networks: For domestic transfers and clearing.
  - Credit Card Networks (Visa, MasterCard, etc.): For card transaction processing.
  - Payment Gateway Integration Module: For connecting to external payment processors.
  - Automated Clearing House (ACH) Systems: For electronic fund transfers.
  - Real-Time Gross Settlement (RTGS) Systems: For high-value transfers.
  - Electronic Fund Transfer at Point of Sale (EFTPOS) Networks: For retail transactions.
- **Third-Party Services**:
  - Credit Bureaus: For credit scoring and verification.
  - Tax Authority Systems: For tax reporting and compliance.
  - Notification Service Providers: For SMS and email communications.
  - Address Verification Services: For validating customer addresses.
  - Postal and Courier Services: For card and document delivery.
  - Credit Scoring Models: For risk assessment.
  - Property Valuation Services: For mortgage-related assessments.
- **Customer-Facing Channels**:
  - Internet Banking Portal: Web interface for customers.
  - Mobile Banking Applications: Apps for smartphones and tablets.
  - ATM Networks: For cash transactions and account inquiries.
  - Point-of-Sale (POS) Systems: For merchant transactions.

  - Interactive Voice Response (IVR) Systems: For telephone banking.
  - Call Center Systems: For customer support.
  - Branch Teller Systems: For in-person banking.
  - Self-Service Kiosks: For automated banking services.
- **Partner Systems**:
  - Insurance Provider Systems: For bancassurance products.
  - Investment and Wealth Management Platforms: For investment services.
  - Mortgage Servicing Systems: For real estate loan processing.
  - Merchant Acquiring Systems: For business banking services.
  - Foreign Exchange Partners: For international currency services.
  - Payroll Processing Systems: For corporate banking clients.
  - Digital Wallet Providers: For electronic payment solutions.

2.3.3 Integration Mechanisms

The following mechanisms facilitate integration between internal and external components:

- API Gateway Module: Centralizes and manages API traffic.
- Message Queuing Systems: Enables asynchronous communication.
- File Transfer Protocols: For batch processing and data exchanges.
- Database Links: For direct database-to-database communication.
- Enterprise Service Bus (ESB): Facilitates service-oriented architecture integration.
- Web Services: SOAP and REST interfaces for component communication.
- Event-Driven Architecture: Publish-subscribe patterns for real-time updates.
- ETL (Extract, Transform, Load) Processes: For data synchronization.

2.4 Style / Design Principles

- Modularity: The system is divided into functional modules (e.g., Account Management, Transaction Processing).
- Security: Data encryption, multi-factor authentication, and role-based access control are implemented.
- Reliability: High availability and fault tolerance are prioritized.
- Compliance: The system adheres to financial regulations and maintains audit trails.

Chapter 3

Architecture Drivers

3.1 Key Features

- Account Management: Creation, modification, and deletion of customer accounts.
- Transaction Processing: Handling deposits, withdrawals, transfers, and bill payments.
- Loan Management: Processing loan applications, disbursements, and repayments.
- Fraud Detection: Monitoring transactions for suspicious activities.
- Reporting: Generating financial and regulatory reports.

3.2 Quality Attributes

- Security: Data encryption, authentication, and authorization mechanisms.
- Reliability: High availability and fault tolerance.
- Performance: Efficient processing of transactions and queries.
- Scalability: Ability to handle increasing transaction volumes.
- Maintainability: Modular design for ease of updates and maintenance.

3.3 Operation & SLAs

- Availability: 99.9% uptime.
- Transaction Processing Time: Less than 2 seconds for 95% of transactions.
- Support: 24/7 customer support and technical assistance.

Chapter 4

System Components

4.1 Account Management Module

4.1.1 Description

Manages customer accounts, including creation, modification, and deletion of savings, checking, and fixed deposit accounts.

4.1.2 Technologies Used

- Programming Language: COBOL
- Database: IBM DB2

4.1.3 Dependencies

- Transaction Processing Module: For updating account balances.
- Customer Service Module: For retrieving account details.

4.2 Transaction Processing Module

4.2.1 Description

Handles deposits, withdrawals, transfers, and bill payments.

4.2.2 Technologies Used

- Programming Language: Java
- Framework: Spring Framework

4.2.3 Dependencies

- Account Management Module: For retrieving and updating account balances.
- Reporting Module: For generating transaction reports.

4.3 Loan Management Module

4.3.1 Description

Processes loan applications, disbursements, repayments, and interest calculations.

4.3.2 Technologies Used

- Programming Language: C#
- Framework: .NET Framework

4.3.3 Dependencies

- Account Management Module: For linking loans to customer accounts.
- Reporting Module: For generating loan-related reports.

4.4 Customer Service Module

4.4.1 Description

Provides customer support through multiple channels (in-branch, online, call centers).

4.4.2 Technologies Used

- Programming Language: Java
- Framework: Spring Framework

4.4.3 Dependencies

- Account Management Module: For retrieving account details.
- Transaction Processing Module: For resolving transaction-related queries.

4.5 Reporting Module

4.5.1 Description

Generates financial reports, transaction histories, and regulatory compliance reports.

4.5.2 Technologies Used

- Programming Language: COBOL
- Database: IBM DB2

4.5.3 Dependencies

- Transaction Processing Module: For retrieving transaction data.
- Loan Management Module: For retrieving loan-related data.

4.6 Fraud Detection Module

4.6.1 Description

Monitors transactions for suspicious activities and potential fraud.

4.6.2 Technologies Used

- Programming Language: Python
- Framework: TensorFlow, Scikit-learn

4.6.3 Dependencies

- Transaction Processing Module: For analyzing transaction data.
- Reporting Module: For generating fraud detection reports.

4.7 Interest Calculation Module

4.7.1 Description

Automates interest calculations for savings and loan accounts.

4.7.2 Technologies Used

- Programming Language: COBOL
- Database: IBM DB2

4.7.3 Dependencies

- Account Management Module: For retrieving account details.
- Loan Management Module: For calculating loan interest.

4.8 Data Archiving Module

4.8.1 Description

Archives historical data for regulatory and analytical purposes.

4.8.2 Technologies Used

- Programming Language: Java
- Database: Oracle

4.8.3 Dependencies

- Reporting Module: For archiving report data.
- Transaction Processing Module: For archiving transaction data.

4.9 Payment Gateway Integration Module

4.9.1 Description

Integrates with external payment gateways for processing payments.

4.9.2 Technologies Used

- Programming Language: Java
- Framework: Spring Framework

4.9.3 Dependencies

- Transaction Processing Module: For processing payment transactions.
- Account Management Module: For updating account balances.

4.10 API Gateway Module

4.10.1 Description

Manages API requests and routes them to the appropriate modules.

4.10.2 Technologies Used

- Programming Language: Java
- Framework: Spring Cloud Gateway

4.10.3 Dependencies

- All modules: For routing API requests.

4.11 Enterprise Service Bus (ESB)

4.11.1 Description

Facilitates service-oriented architecture and integration between components.

4.11.2 Technologies Used

- Technology: Apache Camel
- Programming Language: Java

4.11.3 Dependencies

- API Gateway Module: For service routing.
- All internal modules: For inter-module communication.

4.12 Load Balancing Module

4.12.1 Description

Distributes workload across servers to optimize performance and reliability.

4.12.2 Technologies Used

- Technology: F5 BIG-IP
- Protocol: HTTP, HTTPS

4.12.3 Dependencies

- Performance Monitoring Module: For server health monitoring.
- API Gateway Module: For traffic distribution.

4.13 Scalability Potential

The system has limited scalability due to its monolithic architecture. Modernization efforts aim to break it into microservices for better scalability.

Chapter 5

Conclusion

This document provides a comprehensive overview of the Legacy Banking System, detailing its components, technologies, functionalities, and interdependencies. It serves as a foundation for AI-based dependency mapping and migration strategies.

Chapter 6

Appendix

6.1 References

- IBM DB2 Documentation.
- Spring Framework Official Documentation.
- COBOL Programming Guide.

6.2 Glossary

- COBOL: Common Business-Oriented Language.
- MFA: Multi-Factor Authentication.
- RBAC: Role-Based Access Control.

6.3 Abbreviations

- SLA: Service Level Agreement.
- API: Application Programming Interface.
- ORM: Object-Relational Mapping.

6.4 Change Log

- Version 1.0 (Initial Release): March 26, 2025.