Software Architecture and Design Specification

Project: ATM System

Version: 1.0

Authors: Example Team

Date: 09-09-2025

Status: Draft

# Revision History

Version | Date | Author | Change Summary

# Approvals

Role | Name | Signature/Date

# 1. Introduction

1.1 Purpose  
This document specifies the architecture and design of the ATM System.

1.2 Scope  
Covers ATM user services: cash withdrawal, balance inquiry, and fund transfer.

1.3 Audience  
Developers, QA engineers, security auditors, instructors, and maintenance teams.

1.4 Definitions  
ATM, PIN, ADR, PCI-DSS, TLS.

# 2. Document Overview

2.1 How to use this document  
Provides architectural deliverables including UML diagrams, ADRs, threat models, and API design.

2.2 Related Documents  
SRS, STP, RTM.

# 3. Architecture

**3.1 Goals & Constraints**  
Goals: secure, reliable, 99.9% availability. Constraints: PCI-DSS compliance, ATM hardware limits.

**3.2 Stakeholders & Concerns**  
Customers: security, availability  
Bank Ops: maintainability  
Regulators: compliance  
Developers: modularity.

## 3.3 Component (UML) Diagram

<< **Insert UML Class Diagram here**>>

## 3.4 Component Descriptions

- ATM UI: handles customer input/output.  
- Auth Service: validates PIN.  
- Transaction Processor: processes withdrawals, deposits, transfers.  
- Bank Server API: backend integration.  
- Database: account and transaction data storage.

## 3.5 Chosen Architecture Pattern and Rationale

Layered architecture chosen for clear separation of concerns. Microservices rejected as overly complex for ATM scale.

## 3.6 Technology Stack & Data Stores

Java, Spring Boot APIs, MySQL DB, TLS encryption. ATM terminals use secure modules.

## 3.7 Risks & Mitigations

Risk: ATM downtime due to network failure → Mitigation: redundant connections.

## 3.8 Traceability to Requirements

R1 (PIN validation) → Auth Service  
R2 (Cash withdrawal) → Transaction Processor

## 3.9 Security Architecture

Threat Modeling (STRIDE): Spoofing → chip validation; Tampering → secure boot; Info disclosure → TLS; DoS → rate limiting; Elevation of Privilege → RBAC.

# 4. Design

## 4.1 Design Overview

ATM designed with layered modules for separation of concerns and easier maintenance.

## 4.2 UML Sequence Diagrams

At least 2 sequence diagrams covering 2 flows specific to your project

<< **Insert Sequence Diagram here**>>

## 4.3 API Design

Interface definition for at least 2 components.

Endpoint: /withdraw  
Method: POST  
Request: {cardId, pin, amount}  
Response: {status, balance}  
Errors: 401 Invalid PIN, 402 Insufficient funds.

## 4.4 Error Handling, Logging & Monitoring

Standardized error messages. No sensitive info in logs. Monitoring includes transaction failure rate, ATM uptime.

## 4.5 UX Design

ATM UI with accessible font sizes, language selection, and voice assistance.

## 4.6 Open Issues & Next Steps

Future enhancements: biometric authentication, QR-based mobile withdrawals.

# 5. Appendices

5.1 Glossary: ATM, PIN, PCI-DSS, TLS, ADR.  
5.2 References: IEEE 42010, OWASP, NIST SP 800-160.  
5.3 Tools: PlantUML, draw.io, Swagger.