**FUNCTIONAL REQUIREMENTS - Money Transfer**

**Core Money Transfer Components:**

1. **Transaction Model** (Transaction.java)

* Supports DEPOSIT, WITHDRAWAL, TRANSFER transaction types
* Tracks source and destination accounts
* Transaction status management (PENDING, COMPLETED, FAILED, CANCELLED, REVERSED)
* Amount validation and currency support (INR)

1. **Transfer Service** (TransferService.java)

* Processes transfers between accounts
* Validates account status and sufficient balance
* Prevents self-transfers
* Atomic transaction processing with rollback capability

1. **Deposit Service** (DepositService.java)

* Handles money deposits to accounts
* Validates account status and amount
* Updates account balance

1. **Withdrawal Service** (WithdrawService.java)

* Processes money withdrawals
* Validates sufficient balance and account status
* Updates account balance

1. **Transaction Controller** (TransactionController.java)

* REST API endpoints for all transaction operations
* /api/transactions/deposit - Deposit money
* /api/transactions/withdraw - Withdraw money
* /api/transactions/transfer - Transfer between accounts
* /api/transactions/account/{accountNumber} - Get transaction history

1. **Account Model** (Account.java)

* Core account entity with balance management
* Business methods: canWithdraw(), deposit(), withdraw()
* Account status tracking (ACTIVE, INACTIVE, SUSPENDED, CLOSED)

1. **Account Controller** (AccountController.java)

* Account management endpoints
* Balance inquiry functionality
* Account creation and status updates

**FUNCTIONAL REQUIREMENTS - Audit Logs**

**Core Audit Logging Components:**

1. **Audit Log Model** (AuditLog.java)

* Comprehensive audit trail structure
* Tracks user actions, entity changes, timestamps
* Supports success/failure logging with error messages
* IP address and user agent tracking

1. **Audit Service** (AuditService.java)

* Dual storage: MongoDB + DynamoDB
* Asynchronous DynamoDB logging
* Multiple query methods for audit retrieval
* Success/failure logging methods

1. **Audit Controller** (AuditController.java)

* REST API for audit log access
* /api/audit/logs - Get all audit logs
* /api/audit/logs/user/{userId} - User-specific logs
* /api/audit/logs/date-range - Date range queries
* /api/audit/logs/entity/{entityType}/{entityId} - Entity-specific logs
* /api/audit/logs/action/{action} - Action-specific logs
* /api/audit/stats - Audit statistics

1. **Audit Log Repository** (AuditLogRepository.java)

* MongoDB repository with custom queries
* Advanced filtering by user, date, entity, action
* Existence checks and complex queries

**NON-FUNCTIONAL REQUIREMENTS**

**Data Persistence & Storage:**

1. **MongoDB Configuration** (MongoConfig.java)

* Primary database for accounts, transactions, customers
* Auto-index creation enabled
* Repository pattern implementation

1. **DynamoDB Configuration** (DynamoDBConfig.java)

* Secondary storage for audit logs
* AWS region configuration (ap-south-1)
* Enhanced client for advanced operations

1. **Repository Layer:**

* AccountRepository.java - Account data access
* TransactionRepository.java - Transaction data access
* CustomerRepository.java - Customer data access
* AuditLogRepository.java - Audit log data access

**Transaction Management & Reliability:**

1. **Undo/Redo Stack** (UndoRedoStack.java)

* Transaction reversal capability
* Per-account undo/redo stacks
* Thread-safe implementation

1. **Settlement Queue** (SettlementQueue.java)

* Asynchronous transaction processing
* Queue management for pending transactions
* Thread-safe blocking queue implementation

**Technology Stack & Dependencies:**

1. **Spring Boot Framework** (v3.2.0)

* Web services and REST APIs
* Data validation and transaction management
* Auto-configuration and dependency injection

1. **Database Technologies:**

* MongoDB for primary data storage
* DynamoDB for audit log backup
* Spring Data repositories

1. **AWS Integration:**

* DynamoDB SDK v2.21.29
* AWS credentials and region management

1. **Testing Framework:**

* JUnit 5 for unit testing
* Cucumber for BDD testing
* Spring Boot Test integration

1. **Additional Features:**

* JSON processing with Jackson
* CLI support with Spring Shell
* Cross-origin resource sharing (CORS)
* Input validation with Bean Validation

**Summary**

This banking system comprehensively fulfills the requirements with:

* **Complete money transfer functionality** including deposits, withdrawals, and inter-account transfers
* **Comprehensive audit logging** with dual storage (MongoDB + DynamoDB) and extensive querying capabilities
* **Robust transaction management** with undo/redo capabilities and settlement queues
* **Production-ready architecture** with proper error handling, validation, and monitoring
* **Scalable design** supporting both functional and non-functional requirements for a banking system