Money Transfer System

1. Executive Summary

1.1 Project Snapshot

Attribute	Details
Project Name	Money Transfer System
Project Type	Enterprise Banking Microservice
Total Duration	30 Hours (Per Participant)
Development Approach	Training-Aligned Progressive Build
Training Modules	GIT → Advanced Java → Spring Boot → Angular → Snowflake

1.2 Objective Statement

Develop a **production-grade digital money transfer microservice** progressively aligned with the training curriculum.

Each training module will contribute specific components to the final application, allowing trainees to apply concepts immediately as they learn.

2. Project Overview

2.1 Problem Statement

Modern banking requires digital payment systems that can process fund transfers securely and reliably while maintaining complete transaction audit trails.

2.2 Solution Approach

Build the application **progressively** as each training module is completed:

Training Module	Project Component	Outcome
GIT	Repository & branching	Version-controlled project
Advanced Java	Domain models & logic	Core business layer
Spring Boot	REST APIs & services	Functional backend
Angular	User interface	Complete full-stack app
Snowflake	Analytics warehouse	Business intelligence

3. Training-Aligned Development Approach

3.1 Module Progression Overview

MODULE 1: GIT

- Create GitHub repository
- Setup branching strategy
- Initialize project structure
- Create .gitignore, README

SHOWCASE: Repository with proper structure & branching

MODULE 2: ADVANCED JAVA (Java 17)

- Domain entities (Account, TransactionLog)
- Enums (AccountStatus, TransactionStatus)
- DTOs with validation annotations
- Custom exceptions
- Business logic (debit/credit methods)
- Unit tests for domain logic

SHOWCASE: Domain layer with passing unit tests

MODULE 3: SPRING BOOT

- Spring Boot project setup
- JPA repositories
- Service layer (TransferService, AccountService)
- REST controllers
- Exception handling (@ControllerAdvice)
- Security configuration
- AOP logging aspect
- Database schema & seed data

SHOWCASE: Working REST APIs testable via Postman

MODULE 4: ANGULAR

- Angular project setup
- Login page & authentication
- Dashboard with balance display
- Transfer form with validations
- Transaction history page
- HTTP interceptor for auth
- Integration with backend APIs

SHOWCASE: Complete working full-stack application

MODULE 5: SNOWFLAKE

- Snowflake warehouse setup
- Stage configuration
- Fact & Dimension tables
- ETL pipeline (COPY INTO)
- Analytics queries

SHOWCASE: Analytics dashboard with business insights

3.2 Module Dependencies

Module	Depends On	Provides To
GIT	None	All modules (version control)
Advanced Java	GIT	Spring Boot (domain models)
Spring Boot	Advanced Java	Angular (REST APIs)
Angular	Spring Boot	Snowflake (transaction data)
Snowflake	Spring Boot	None (final analytics)

---

4. Technology Stack

4.1 Stack by Training Module

Module	Technologies
GIT	Git, GitHub, Git Bash
Advanced Java	Java 17, JUnit 5, Lombok, Bean Validation
Spring Boot	Spring Boot 3.x, Spring Data JPA, Spring Security, Spring AOP, MySQL
Angular	Angular 15+, TypeScript, Angular Material, RxJS
Snowflake	Snowflake Cloud, SQL, Stages, COPY INTO

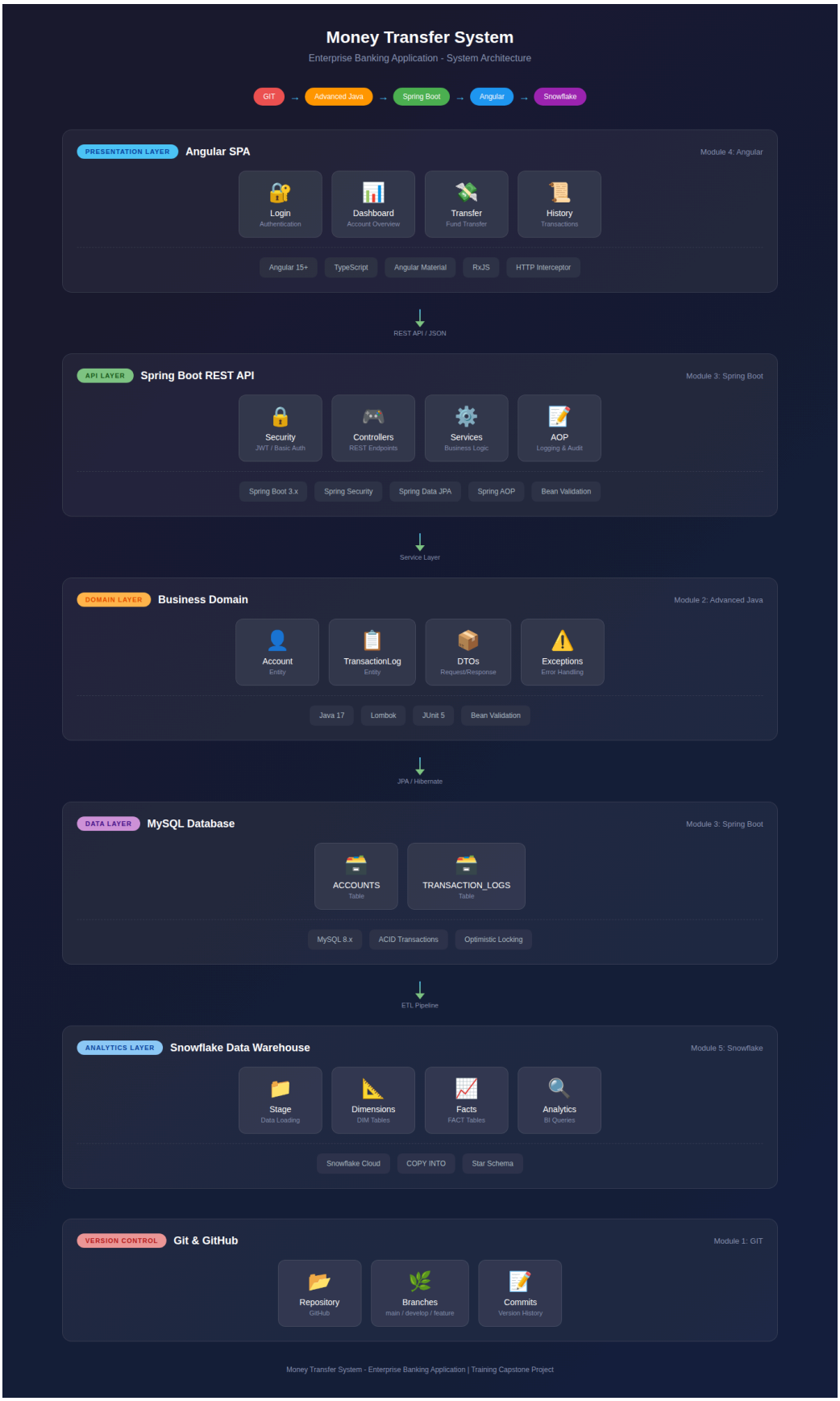
4.2 Complete Technology Matrix

Category	Technology	Version	Module
Version Control	Git	2.x	GIT
Repository	GitHub	-	GIT
Language	Java	17 LTS	Advanced Java
Testing	JUnit	5.x	Advanced Java

Category	Technology	Version	Module
Code Generation	Lombok	1.18.x	Advanced Java
Framework	Spring Boot	3.x	Spring Boot
ORM	Spring Data JPA	3.x	Spring Boot
Security	Spring Security	6.x	Spring Boot
Database	MySQL	8.x	Spring Boot
Frontend	Angular	18+	Angular
Data Warehouse	Snowflake	Cloud	Snowflake

---

5. Architecture Diagram


Money Transfer System
Enterprise Banking Application - System Architecture
GIT
Advanced Java
Spring Boot
Angular
Snowflake
PRESENTATION LAYER
Angular SPA
Module 4: Angular
Login
Authentication
Dashboard
Account Overview
Transfer
Fund Transfer
History
Transactions
Angular 15+
TypeScript
Angular Material
RxJS
HTTP Interceptor
REST API / JSON
API LAYER
Spring Boot REST API
Module 3: Spring Boot
Security
JWT / Basic Auth
Controllers
REST Endpoints
Services
Business Logic
AOP
Logging & Audit
Spring Boot 3.x
Spring Security
Spring Data JPA
Spring AOP
Bean Validation
Service Layer
DOMAIN LAYER
Business Domain
Module 2: Advanced Java
Account
Entity
TransactionLog
Entity
DTOs
Request/Response
Exceptions
Error Handling
Java 17
Lombok
JUnit 5
Bean Validation
JPA / Hibernate
DATA LAYER
MySQL Database
Module 3: Spring Boot
ACCOUNTS
Table
TRANSACTION_LOGS
Table
MySQL 8.x
ACID Transactions
Optimistic Locking
ETL Pipeline
ANALYTICS LAYER
Snowflake Data Warehouse
Module 5: Snowflake
Stage
Data Loading
Dimensions
DIM Tables
Facts
FACT Tables
Analytics
BI Queries
Snowflake Cloud
COPY INTO
Star Schema
VERSION CONTROL
Git & GitHub
Module 1: GIT
Repository
GitHub
Branches
main / develop / feature
Commits
Version History
Money Transfer System - Enterprise Banking Application | Training Capstone Project

6. Module 1: GIT - Version Control Setup

6.1 Module Overview

Attribute	Details
Training Module	GIT
Project Component	Repository Setup & Version Control
Duration	2 Hours
Prerequisites	Git installed, GitHub account

6.2 Tasks & Deliverables

#	Task	Description	Time
1	Create GitHub Repository	Initialize remote repository	0.25 hr
2	Clone & Setup	Clone repo, configure git settings	0.25 hr
3	Branching Strategy	Create develop, feature branches	0.5 hr
4	Project Structure	Create folder hierarchy	0.5 hr
5	Git Configuration	.gitignore, README.md	0.5 hr

6.3 Repository Structure to Create

```
money-transfer-system/
│
├── backend/
│   ├── src/
│   │   ├── main/
│   │   │   ├── java/
```

```
│   │   │   └── resources/
│   │   └── test/
│   └── pom.xml
│
├── frontend/
│   └── (Angular project - Module4)
│
├──database/
│   ├──schema.sql
│   └── seed-data.sql
│
├── snowflake/
│   └── (Snowflake scripts - Module5)
│
├── docs/
│   └── (Documentation)
│
├── .gitignore
└── README.md
```

6.4 Branching Strategy

Branch	Purpose	Created From
`main`	Production-ready code	-
`develop`	Integration branch	main
`feature/domain-models`	Java domain classes	develop
`feature/spring-boot-api`	REST API development	develop
`feature/angular-ui`	Frontend development	develop
`feature/snowflake-analytics`	Analytics setup	develop

6.5 Module Showcase

Showcase Item	Description
GitHub Repository	Public/Private repo with proper structure
Branch Structure	Main, develop, and feature branches
Initial Commit	Project skeleton with .gitignore and README
Commit History	Clean commit messages

---

7. Module 2: Advanced Java - Domain & Business Logic

7.1 Module Overview

Attribute	Details
Training Module	Advanced Java (Java 17)
Project Component	Domain Models & Business Logic
Duration	6 Hours
Prerequisites	Module 1 complete

7.2 Tasks & Deliverables

#	Task	Description	Time
1	Account Entity	Create Account class with fields & methods	1.0 hr
2	TransactionLog Entity	Create TransactionLog class	0.75 hr
3	Enums	AccountStatus, TransactionStatus	0.25 hr
4	DTOs	TransferRequest, TransferResponse, etc.	1.0 hr
5	Validation Annotations	Add Bean Validation to DTOs	0.5 hr

#	Task	Description	Time
6	Custom Exceptions	Create exception hierarchy	0.75 hr
7	Business Logic	Debit/Credit methods with rules	1.0 hr
8	Unit Tests	JUnit tests for domain logic	0.75 hr

7.3 Java 17 Features to Demonstrate

Feature	Usage in Project
Records	DTO classes (optional)
Sealed Classes	Exception hierarchy (optional)
Pattern Matching	instanceof checks
Text Blocks	Multi-line strings in tests
Enhanced Switch	Status-based logic

7.4 Domain Classes to Create

Entities

Class	Fields	Methods
Account	id, holderName, balance, status, version, lastUpdated	debit(), credit(), isActive()
TransactionLog	id, fromAccountId, toAccountId, amount, status, failureReason, idempotencyKey, createdOn	-

Enums

Enum	Values
AccountStatus	ACTIVE, LOCKED, CLOSED
TransactionStatus	SUCCESS, FAILED

DTOs

Class	Purpose	Validations
TransferRequest	Input for transfer API	@NotNull, @DecimalMin
TransferResponse	Output for transfer API	-
AccountResponse	Output for account API	-
ErrorResponse	Error output	-

Exceptions

Exception	Use Case
AccountNotFoundException	Account ID doesn't exist
AccountNotActiveException	Account is LOCKED/CLOSED
InsufficientBalanceException	Balance < transfer amount
DuplicateTransferException	Idempotency key reused

7.5 Unit Tests to Write

Test Class	Test Cases
AccountTest	testDebit_Success, testDebit_InsufficientBalance, testCredit_Success
TransferRequestValidationTest	testValidRequest, testInvalidAmount, testNullFields

7.6 Module Showcase

Showcase Item	Description
Domain Package	Complete entity classes with JPA annotations ready
DTO Package	Request/Response objects with validation
Exception Package	Custom exception hierarchy
Unit Tests	All tests passing (green)
Git Commit	Feature branch merged to develop

8. Module 3: Spring Boot - REST API Development

8.1 Module Overview

Attribute	Details
Training Module	Spring Boot
Project Component	REST APIs, Services, Data Access
Duration	12 Hours
Prerequisites	Module 2 complete

8.2 Tasks & Deliverables

#	Task	Description	Time
1	Spring Boot Setup	Initialize project, dependencies	1.0 hr
2	Database Configuration	MySQL connection, application.yml	0.5 hr
3	JPA Repositories	AccountRepository, TransactionLogRepository	0.5 hr
4	TransferService	Core transfer logic with transaction management	2.5 hr
5	AccountService	Account operations	1.0 hr
6	Idempotency Logic	Duplicate prevention mechanism	1.0 hr
7	TransferController	Transfer REST endpoints	1.0 hr
8	AccountController	Account REST endpoints	0.75 hr
9	Global Exception Handler	@ControllerAdvice implementation	1.0 hr
10	Security Configuration	Basic Auth or JWT setup	1.5 hr
11	AOP Logging Aspect	Method logging, execution time	0.75 hr
12	Database Scripts	Schema DDL, seed data	0.5 hr

8.3 Spring Boot Components

Configuration Files

File	Purpose
application.yml	Database, server, logging config
pom.xml	Maven dependencies

Dependencies Required

Dependency	Purpose
spring-boot-starter-web	REST API
spring-boot-starter-data-jpa	Database access
spring-boot-starter-validation	Input validation
spring-boot-starter-security	Authentication
spring-boot-starter-aop	Aspect programming
mysql-connector-java	MySQL driver
lombok	Code generation
spring-boot-starter-test	Testing

8.4 REST Endpoints to Implement

Endpoint	Method	Description
`/api/v1/transfers`	POST	Execute fund transfer
`/api/v1/accounts/{id}`	GET	Get account details
`/api/v1/accounts/{id}/balance`	GET	Get account balance
`/api/v1/accounts/{id}/transactions`	GET	Get transaction history

8.5 Service Layer Implementation

Service	Methods
TransferService	transfer(TransferRequest), validateTransfer(), executeTransfer()
AccountService	getAccount(id), getBalance(id), getTransactions(id)

8.6 Cross-Cutting Concerns

Concern	Implementation
Transaction Management	@Transactional annotation
Exception Handling	@ControllerAdvice, @ExceptionHandler
Logging	AOP @Around aspect
Security	SecurityFilterChain, authentication

8.7 Module Showcase

Showcase Item	Description
Running Application	Spring Boot app starts successfully
Postman Collection	All APIs tested and working
Database	Tables created, seed data loaded
Security	Protected endpoints, auth required
Logs	AOP logging visible in console
Git Commit	Feature branch merged to develop

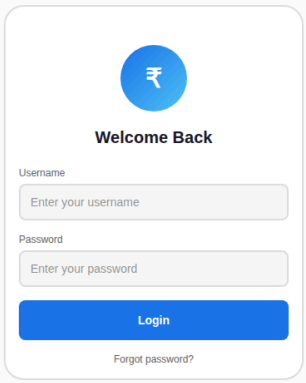
9. Module 4: Angular - Frontend Application

Money Transfer System

UI Wireframes - Angular Frontend Screens

1 Login Screen

Authentication

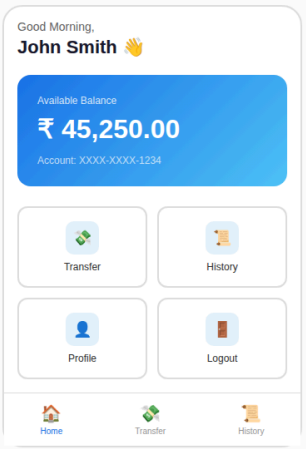


SCREEN FEATURES

- Username & Password fields with validation
- Login button triggers authentication API
- Error message display for invalid credentials
- JWT token stored on successful login
- Redirect to Dashboard on success

2 Dashboard Screen

Account Overview

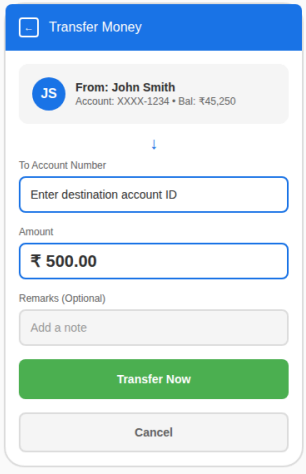


SCREEN FEATURES

- Display account holder name from API
- Show current balance (real-time)
- Quick action buttons for navigation
- Bottom navigation bar
- Logout clears session & redirects

3 Transfer Screen

Fund Transfer

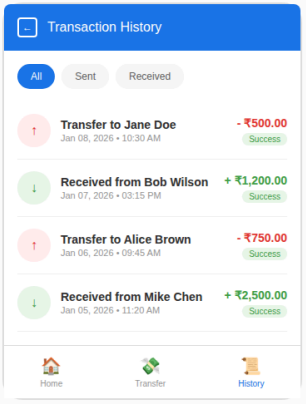


SCREEN FEATURES

- Source account shown (read-only)
- Destination account input with validation
- Amount field with min value check
- Confirmation before API call
- Success/Error message display
- Generates unique idempotency key

4 History Screen

Transaction History



SCREEN FEATURES

- Filter tabs: All / Sent / Received
- Transaction list with debit/credit icons
- Color coded amounts (red/green)
- Date & time formatting
- Status badge for each transaction
- Sorted by date (newest first)

Money Transfer System - UI Wireframes | Angular Frontend | Module 4

9.1 Module Overview

Attribute	Details
Training Module	Angular
Project Component	Single Page Application (UI)
Duration	8 Hours
Prerequisites	Module 3 complete

9.2 Tasks & Deliverables

#	Task	Description	Time
1	Angular Setup	New project, routing, material	0.5 hr
2	Auth Service	Login logic, token storage	1.0 hr
3	Login Page	Login form, validation, error handling	1.0 hr
4	HTTP Interceptor	Auth header injection	0.5 hr
5	Dashboard Page	Balance display, navigation	2.0 hr
6	Transfer Page	Transfer form, validation, API call	2.0 hr
7	History Page	Transaction table, formatting	1.0 hr

9.3 Angular Components

Component	Features
LoginComponent	Form, validation, error messages, redirect
DashboardComponent	Welcome message, balance card, action buttons
TransferComponent	Form, validation, confirmation modal, result display
HistoryComponent	Table, date formatting, status badges

9.4 Angular Services

Service	Methods
AuthService	login(), logout(), isAuthenticated(), getToken()
AccountService	getAccount(), getBalance(), getTransactions()
TransferService	transfer()

9.5 Screen Specifications

Login Screen

Element	Type	Details
Username	Input	Required, text
Password	Input	Required, password type
Login Button	Button	Submits form
Error Alert	Alert	Shows on failure

Dashboard Screen

Element	Type	Details
Welcome Banner	Text	"Welcome, {holderName}"
Balance Card	Card	Current balance, large font
Transfer Button	Button	Navigate to transfer
History Button	Button	Navigate to history
Logout Button	Button	Clear session

Transfer Screen

Element	Type	Details
From Account	Display	Current account (readonly)
To Account	Input	Destination account ID
Amount	Input	Transfer amount
Transfer Button	Button	Submit transfer
Cancel Button	Button	Return to dashboard

Element	Type	Details
Result Message	Alert	Success/Error message

History Screen

Element	Type	Details
Transaction Table	Table	List of transactions
Date Column	Column	Formatted date/time
Type Column	Column	DEBIT (red) / CREDIT (green)
Amount Column	Column	Formatted currency
Status Column	Column	Badge (Success/Failed)

9.6 Module Showcase

Showcase Item	Description
Running Angular App	ng serve working
Login Flow	Authentication working
Dashboard	Balance displayed from API
Transfer Flow	End-to-end transfer working
History View	Transactions displayed
Full Stack Demo	Frontend + Backend integrated
Git Commit	Feature branch merged to develop

10. Module 5: Snowflake - Data Analytics

10.1 Module Overview

Attribute	Details
Training Module	Snowflake

Attribute	Details
Project Component	Data Warehouse & Analytics
Duration	2 Hours
Prerequisites	Module 3 complete (transaction data)

10.2 Tasks & Deliverables

#	Task	Description	Time
1	Snowflake Setup	Database, warehouse, schema	0.25 hr
2	Stage Configuration	Internal/External stage	0.25 hr
3	Dimension Tables	DIM_ACCOUNT, DIM_DATE	0.5 hr
4	Fact Tables	FACT_TRANSACTIONS	0.5 hr
5	ETL Pipeline	COPY INTO commands	0.25 hr
6	Analytics Queries	Business intelligence queries	0.25 hr

10.3 Snowflake Objects to Create

Database Structure

Object	Name	Purpose
Database	MONEY_TRANSFER_DW	Data warehouse
Schema	ANALYTICS	Analytics objects
Warehouse	COMPUTE_WH	Query processing

Dimension Tables

Table	Columns
DIM_ACCOUNT	account_key, account_id, holder_name, status, effective_date
DIM_DATE	date_key, full_date, day, month, year, quarter

Fact Tables

Table	Columns
FACT_TRANSACTIONS	transaction_key, account_from_key, account_to_key, date_key, amount, status

10.4 Analytics Queries

Query	Purpose
Daily Transaction Volume	Count and sum by date
Account Activity	Most active accounts
Success Rate	Percentage of successful transfers
Peak Hours	Busiest transaction times
Average Transfer Amount	Mean transfer value

10.5 Module Showcase

Showcase Item	Description
Snowflake Objects	Database, schema, warehouse created
Dimensional Model	Fact and dimension tables
Data Loaded	Transaction data in warehouse
Analytics Queries	Working queries with results
Git Commit	Scripts committed to repository

11. Functional Requirements

11.1 Core Features

ID	Feature	Module	Description
FR-01	Fund Transfer	Spring Boot	Transfer funds between accounts

ID	Feature	Module	Description
FR-02	Account Validation	Spring Boot	Verify account existence/status
FR-03	Balance Check	Spring Boot	Ensure sufficient funds
FR-04	Transaction Logging	Spring Boot	Record all transfers
FR-05	Idempotency	Spring Boot	Prevent duplicates
FR-06	Balance Inquiry	Spring Boot + Angular	View current balance
FR-07	Transaction History	Spring Boot + Angular	View past transactions
FR-08	User Authentication	Spring Boot + Angular	Secure login
FR-09	Analytics	Snowflake	Business insights

12. Non-Functional Requirements

ID	Requirement	Module	Implementation
NFR-01	ACID Compliance	Spring Boot	@Transactional
NFR-02	Concurrency Control	Spring Boot	Optimistic locking
NFR-03	Security	Spring Boot	Auth + Validation
NFR-04	Audit Trail	Spring Boot	AOP logging
NFR-05	Version Control	GIT	Git branching

13. Domain Model

13.1 Account Entity

Field	Type	Description
id	Long	Primary key
holderName	String	Account holder name
balance	BigDecimal	Current balance
status	AccountStatus	ACTIVE/LOCKED/CLOSED
version	Integer	Optimistic lock
lastUpdated	Timestamp	Last modified

13.2 TransactionLog Entity

Field	Type	Description
id	UUID	Primary key
fromAccountId	Long	Source account
toAccountId	Long	Destination account
amount	BigDecimal	Transfer amount
status	TransactionStatus	SUCCESS/FAILED
failureReason	String	Error if failed
idempotencyKey	String	Unique request ID
createdOn	Timestamp	Transaction time

14. Database Design

14.1 OLTP Tables (MySQL)

ACCOUNTS Table

Column	Type	Constraints
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT
holder_name	VARCHAR(255)	NOT NULL
balance	DECIMAL(18,2)	NOT NULL

Column	Type	Constraints
status	VARCHAR(20)	NOT NULL
version	INT	DEFAULT 0
last_updated	TIMESTAMP	AUTO UPDATE

TRANSACTION_LOGS Table

Column	Type	Constraints
id	VARCHAR(36)	PRIMARY KEY
from_account	BIGINT	FOREIGN KEY
to_account	BIGINT	FOREIGN KEY
amount	DECIMAL(18,2)	NOT NULL
status	VARCHAR(20)	NOT NULL
failure_reason	VARCHAR(255)	NULLABLE
idempotency_key	VARCHAR(100)	UNIQUE
created_on	TIMESTAMP	DEFAULT NOW

14.2 Analytics Tables (Snowflake)

FACT_TRANSACTIONS

Column	Type	Description
transaction_key	NUMBER	Surrogate key
account_from_key	NUMBER	FK to DIM_ACCOUNT
account_to_key	NUMBER	FK to DIM_ACCOUNT
date_key	NUMBER	FK to DIM_DATE
amount	DECIMAL(18,2)	Transfer amount
status	VARCHAR(20)	Transaction status

15. API Specification

15.1 Transfer API

Endpoint: `POST /api/v1/transfers`

Request:

```
{
"fromAccountId":1,
"toAccountId":2,
"amount":500.00,
"idempotencyKey":"uuid-string"
}
```

Success Response (200):

```
{
"transactionId":"TRX-uuid",
"status":"SUCCESS",
"message":"Transfer completed",
"debitedFrom":1,
"creditedTo":2,
"amount":500.00
}
```

Error Response (4xx):

```
{
"errorCode":"TRX-400",
"message":"Insufficient balance"
}
```

15.2 Account APIs

Endpoint	Method	Response
`/api/v1/accounts/{id}`	GET	Account details
`/api/v1/accounts/{id}/balance`	GET	Balance only
`/api/v1/accounts/{id}/transactions`	GET	Transaction list

---

16. Business Rules

#	Rule	Error Code
1	Accounts must be different	VAL-422
2	Source account must exist	ACC-404
3	Destination account must exist	ACC-404
4	Source account must be ACTIVE	ACC-403
5	Destination account must be ACTIVE	ACC-403
6	Amount must be > 0	VAL-422
7	Source balance >= amount	TRX-400
8	Idempotency key must be unique	TRX-409
9	Debit before credit	-
10	Log every transfer	-

---

17. Error Handling

17.1 Error Catalog

Code	Message	HTTP
ACC-404	Account not found	404

Code	Message	HTTP
ACC-403	Account not active	403
TRX-400	Insufficient funds	400
TRX-409	Duplicate transfer	409
VAL-422	Invalid input	422

---

18. Security Implementation

Feature	Implementation
Authentication	Basic Auth or JWT
Protected Routes	All /api/** endpoints
Input Validation	Bean Validation
Password Storage	BCrypt encoding

---

19. Effort Estimation

19.1 Summary by Training Module

Module	Component	Hours	%
GIT	Repository & Version Control	2	7%
Advanced Java	Domain Models & Business Logic	6	20%
Spring Boot	REST APIs & Backend Services	12	40%
Angular	Frontend Application	8	27%
Snowflake	Data Analytics	2	6%
Total		**30**	**100%**

19.2 GIT Module (2 Hours)

Task	Hours
Create GitHub repository	0.25
Clone & configure	0.25
Branching strategy	0.5
Project structure	0.5
.gitignore & README	0.5
Subtotal	**2.0**

19.3 Advanced Java Module (6 Hours)

Task	Hours
Account entity	1.0
TransactionLog entity	0.75
Enums	0.25
DTOs with validations	1.0
Validation annotations	0.5
Custom exceptions	0.75
Business logic (debit/credit)	1.0
Unit tests	0.75
Subtotal	**6.0**

19.4 Spring Boot Module (12 Hours)

Task	Hours
Project setup	1.0
Database configuration	0.5
JPA repositories	0.5
TransferService	2.5
AccountService	1.0

Task	Hours
Idempotency logic	1.0
TransferController	1.0
AccountController	0.75
Exception handler	1.0
Security configuration	1.5
AOP logging aspect	0.75
Database scripts	0.5
Subtotal	**12.0**

19.5 Angular Module (8 Hours)

Task	Hours
Project setup	0.5
Auth service	1.0
Login page	1.0
HTTP interceptor	0.5
Dashboard page	2.0
Transfer page	2.0
History page	1.0
Subtotal	**8.0**

19.6 Snowflake Module (2 Hours)

Task	Hours
Snowflake setup	0.25
Stage configuration	0.25
Dimension tables	0.5
Fact tables	0.5
ETL pipeline	0.25
Analytics queries	0.25

Task	Hours
Subtotal	**2.0**

20. Deliverables by Module

20.1 Module-wise Deliverables

Module	Deliverables	Showcase
GIT	GitHub repo, branches, project structure	Repository walkthrough
Advanced Java	Domain classes, DTOs, exceptions, unit tests	Tests passing
Spring Boot	REST APIs, services, security, DB scripts	Postman demo
Angular	SPA with 4 screens, API integration	Full-stack demo
Snowflake	DW schema, ETL, analytics queries	Query results

20.2 Final Deliverables

#	Deliverable	Description
1	GitHub Repository	Complete source code
2	Backend Application	Working Spring Boot app
3	Frontend Application	Working Angular app
4	Database Scripts	DDL + seed data
5	Snowflake Scripts	DW setup + queries
6	Unit Tests	Passing test suite
7	Documentation	README with setup guide

21. Evaluation Criteria

21.1 Module-wise Evaluation

Module	Weight	Criteria
GIT	10%	Repo structure, branching, commits
Advanced Java	20%	Domain model, OOP, tests
Spring Boot	35%	API design, services, security
Angular	25%	UI components, integration
Snowflake	10%	DW design, analytics
Total	**100%**	