🏦 Bank Management Application – Case Study

🔧 Tools & Technologies Used

Category	Technology Stack
Frontend	HTML, CSS, Angular (Version 18)
Backend	Spring Boot, Spring MVC, Spring Data JPA, Spring Security
Programming Language	Java
Database	MySQL / Oracle / H2 (for development & testing)
Server	Apache Tomcat 9.5

🎯 Objective

The **Bank Management System** is designed as a full-stack web application that automates manual banking operations. It enables **bank managers** and **bank employees** to efficiently manage customer accounts, perform financial transactions, and track banking activities.

This in-house solution simplifies day-to-day tasks and brings transparency, security, and accountability to banking operations.

👥 System Users

There are two primary roles in the application:

1️⃣ Bank Manager (Admin)

The **Bank Manager** acts as the superuser or admin of the system.

✅ **Responsibilities:**

- Add, edit, and update **customer bank accounts**
- Add, edit, and update **bank employees (clerks)**
- **Authorize bank employees** to perform transactions
- Monitor and manage all **online transaction logs**
- When a new customer is added:
  - A **unique account number** is auto-generated
  - Account details are securely stored

2️⃣ Bank Employee (Clerk)

The **Bank Employee** is authorized by the manager to serve customers.

✅ **Responsibilities:**

- Perform **transactions** on behalf of customers:
    - Deposit funds
    - Withdraw funds
    - Transfer funds between accounts
- **Print transaction details** on the customer’s passbook

🗃️ Key Functional Modules

Module	Description
User Management	CRUD operations for customers and employees
Account Handling	Auto-generation and management of customer account numbers
Transaction System	Real-time handling of deposits, withdrawals, and transfers
Authentication & Authorization	Role-based access using Spring Security & JWT
Audit & Logs	Track all user activities and transaction logs
Passbook Service	Generate printable transaction summary for customers

---

🔒 Security Features

- Login via **Angular + Spring Security OAuth2** (optional: Keycloak or JWT)
- Role-based access control:
    - Manager: full admin privileges
    - Clerk: restricted to customer service
- HTTPS & CORS enabled (for real-world deployment)
- Token-based session handling

📈 Scalability & Deployment

- Microservice-ready architecture (modular codebase)
- Easily extendable to support:
    - Credit cards, Loans, KYC, Branch management
- Docker and CI/CD pipeline compatible
- Can be deployed on **Tomcat, AWS EC2, Azure**, or **Spring Cloud**

📌 Future Enhancements (Optional)

- Customer self-service portal (login, view balance, request services)
- Notification system (SMS/email alerts)
- Integration with external payment gateways (UPI, NEFT)