Banking System

a prototype Subsystem for a bank

Agenda

1- Introduction

2- Tool Description

3- UML Diagrams

4- Project Architecture

5- Design Patterns

6- Live Demo

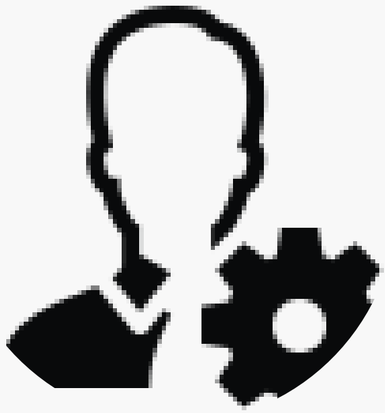
7- Future Work

Introduction

Our project focuses on designing and implementing a comprehensive banking management system to streamline operations across multiple user roles and financial services. The system will handle

- Financial Services: Transactions, Loans, and Certificates (e.g., fixed deposits).
- Operational Units: Branches and Ticket Booking (for appointment scheduling).

Bank Roles

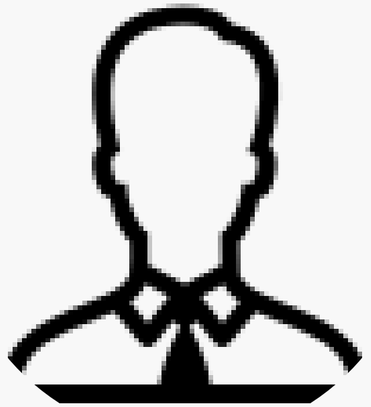


Admin

Certificate | Manager | Branch

Manager

Branch | Loan | Savings | Teller



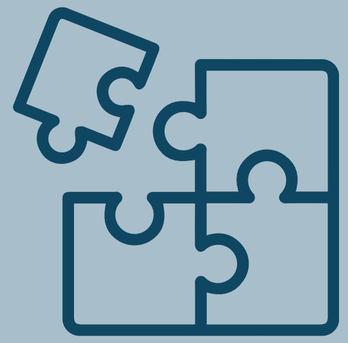
Teller

Account | Cards | Customer | Ticket

BankObjectives

Certificates

- Admin shall create a general certefiacte that the bank shall provide
- each type of the certificate has it’s own interest



Loans

- User Can apply for a Loan by adding PDFs to be verified
- Manager can either accept or reject it based on the documents.

Transactions

- each withdraw, deposit, is saved in a transaction table
- All the Banks managerial staff can access those transactions

Customer

Benefits:

The customer can book a reservation, create an account, make a card, apply for a loan, get a certificate, and withdraw, transfer, or deposit money.

What Is Gliffy ?

- Powered by Atlassian (same company as Jira).
- Easily create diagrams, ERDs, and flowcharts.
- Simply drag - drop shapes
- Fully integrated with Confluence, perfect for teams.

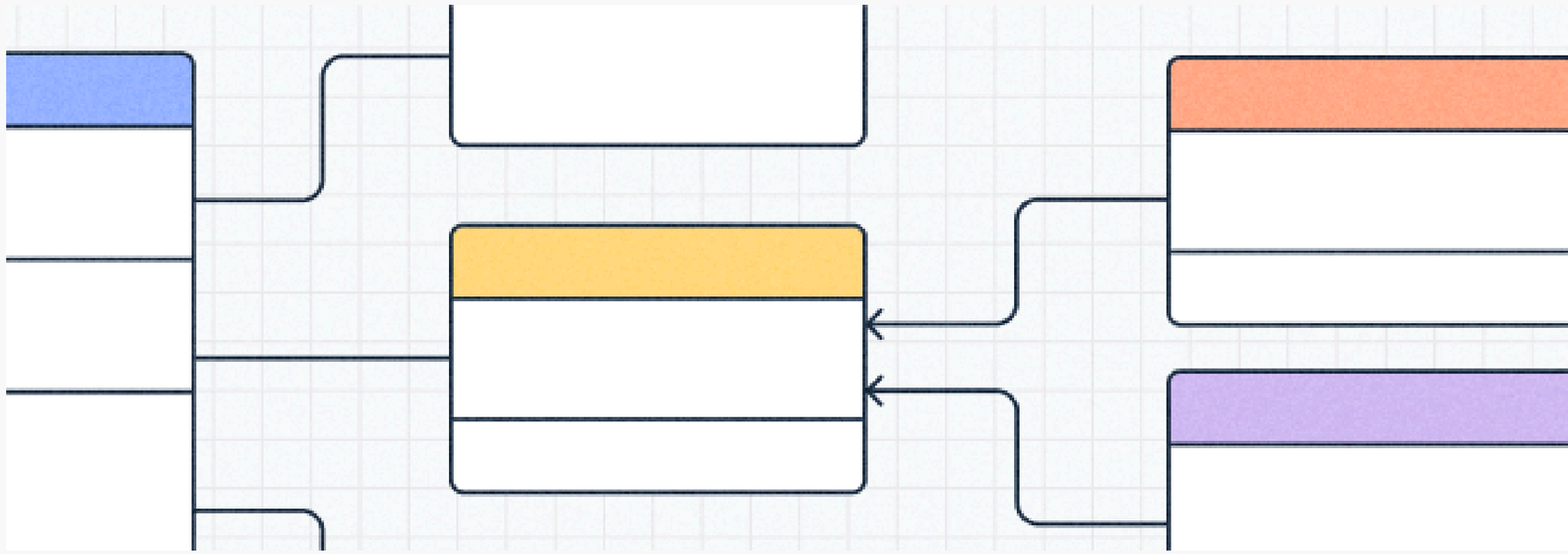
UML Diagrams

1- Use Case Diagram

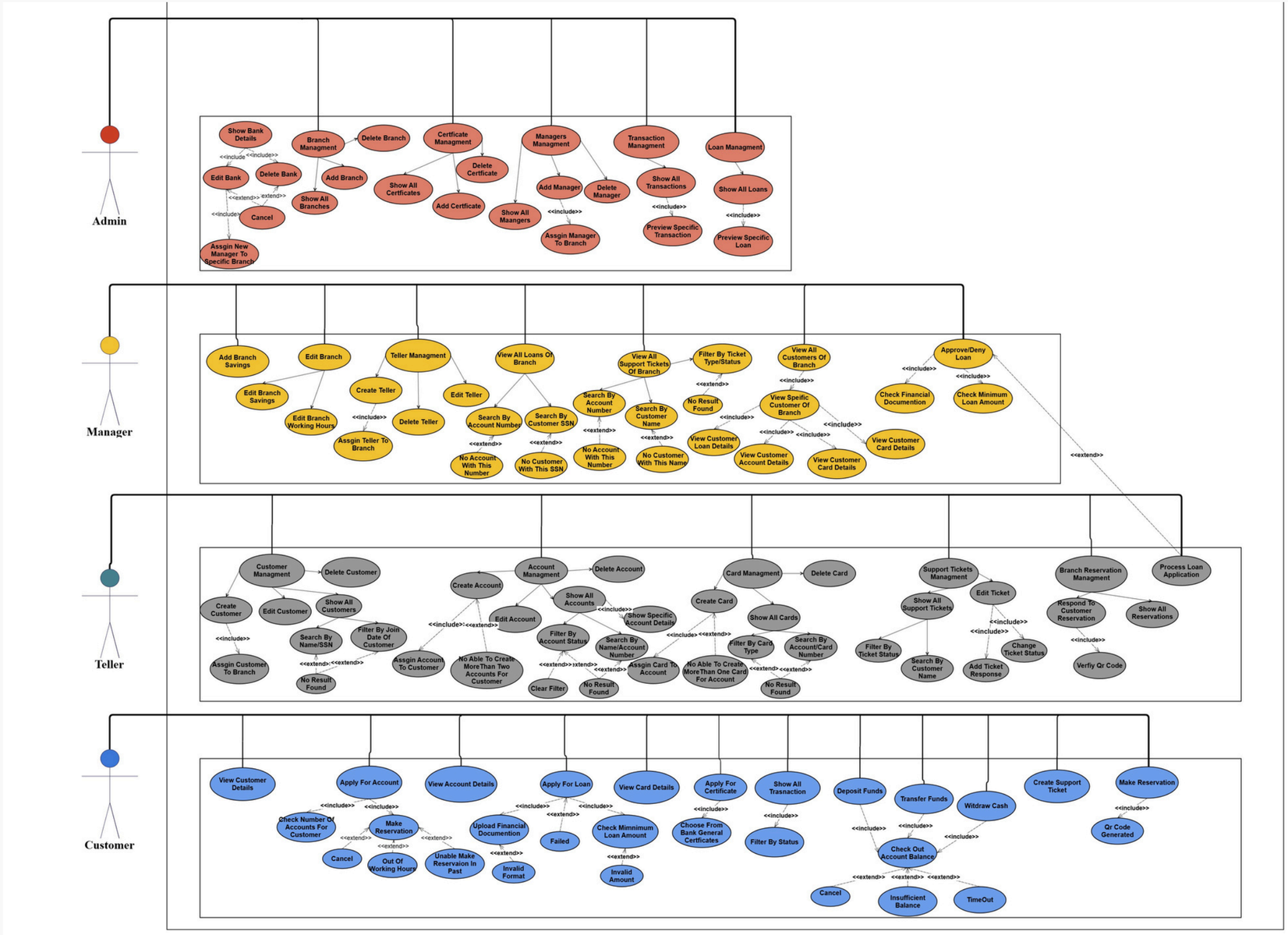
2- Sequence Diagram

3- Class Diagram

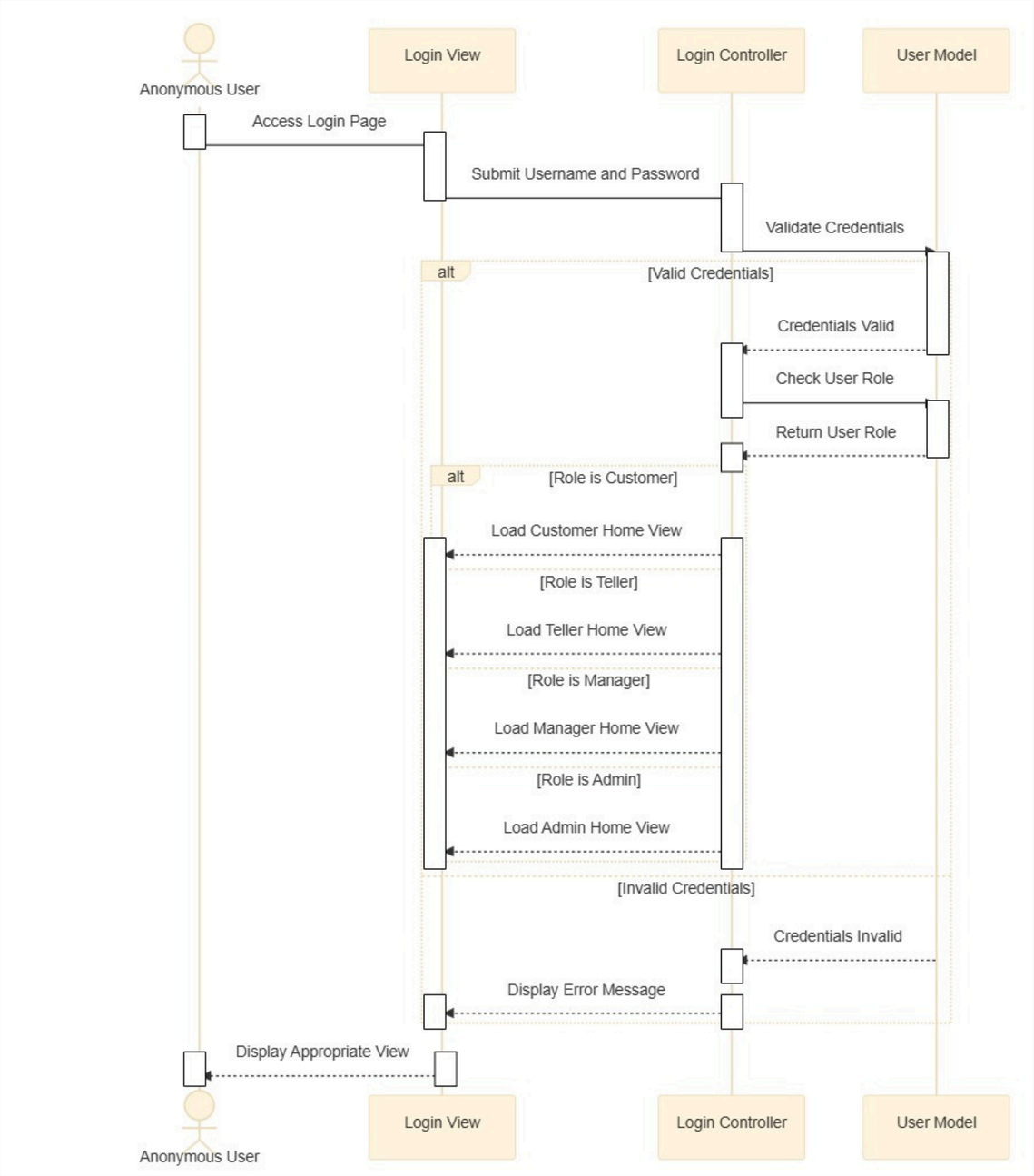
4- State Diagram



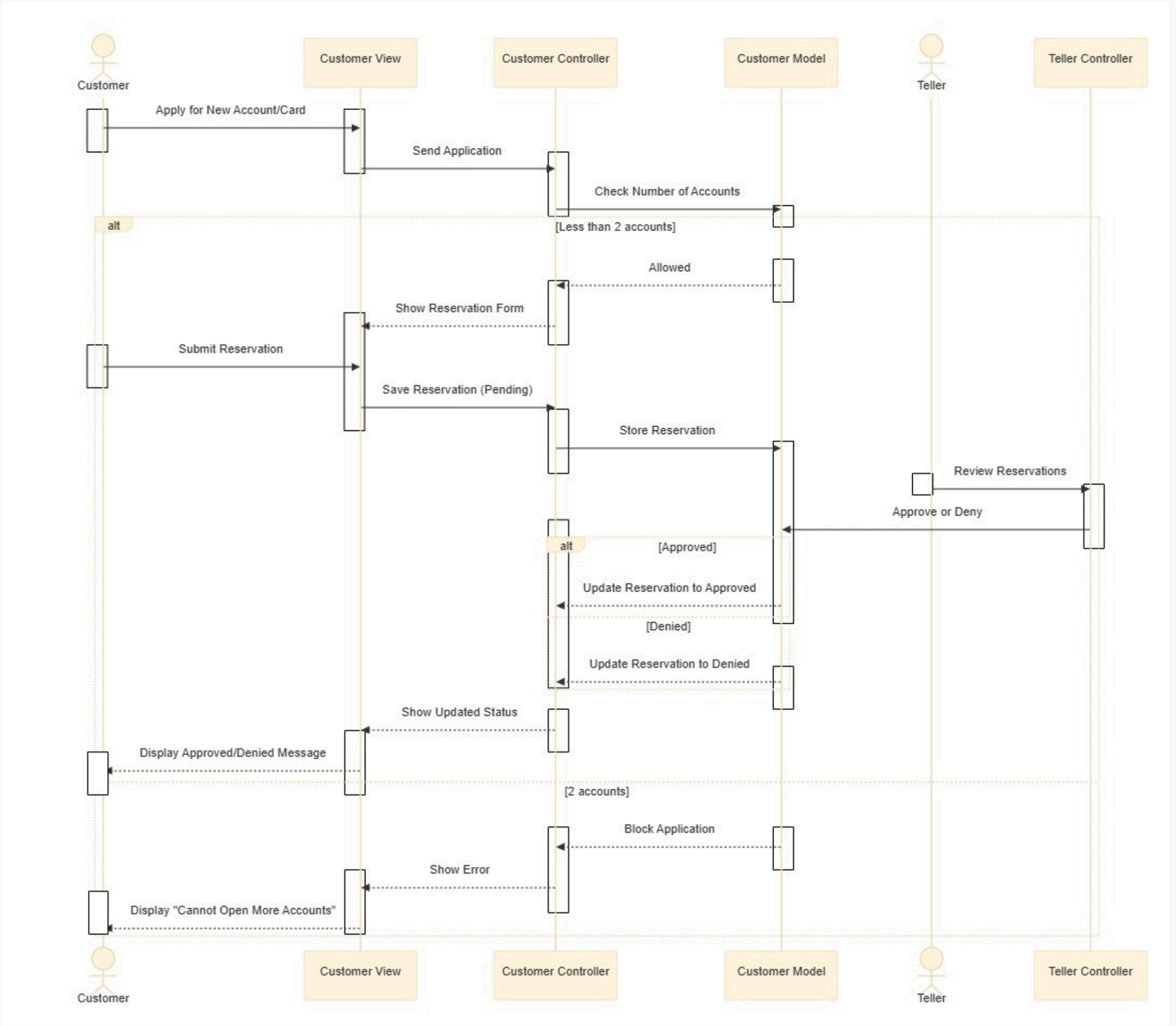
Use Case Diagram



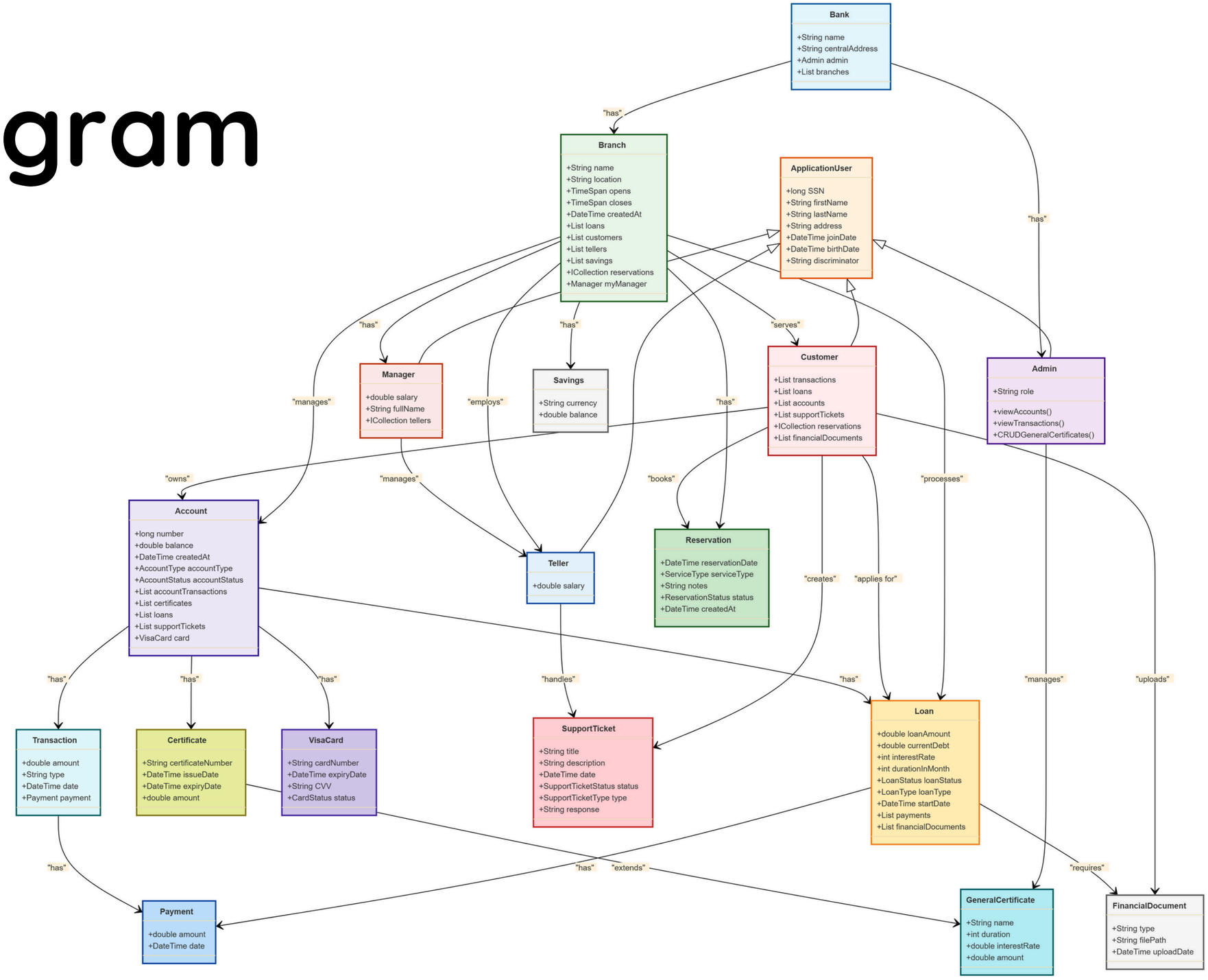
LOGIN SEQUENCE DIAGRAM



ACCOUNT & CARD SEQUENCED DIAGRAM



Class Diagram



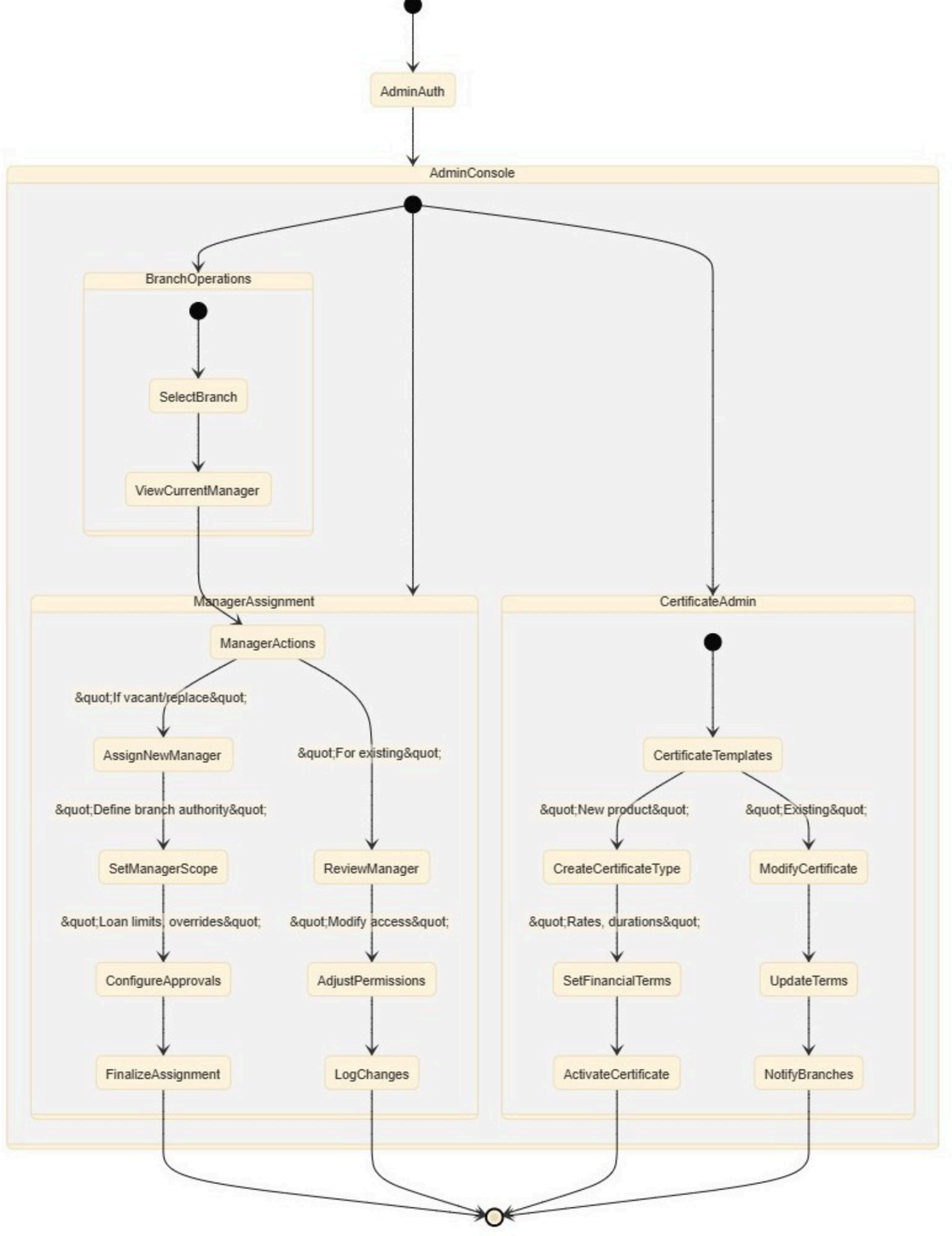
State Diagram

Admin State

1. Opening/closing branches and setting them up.
2. Assigning managers to branches and

   controlling what they can do
3. Creating and managing certificate accounts

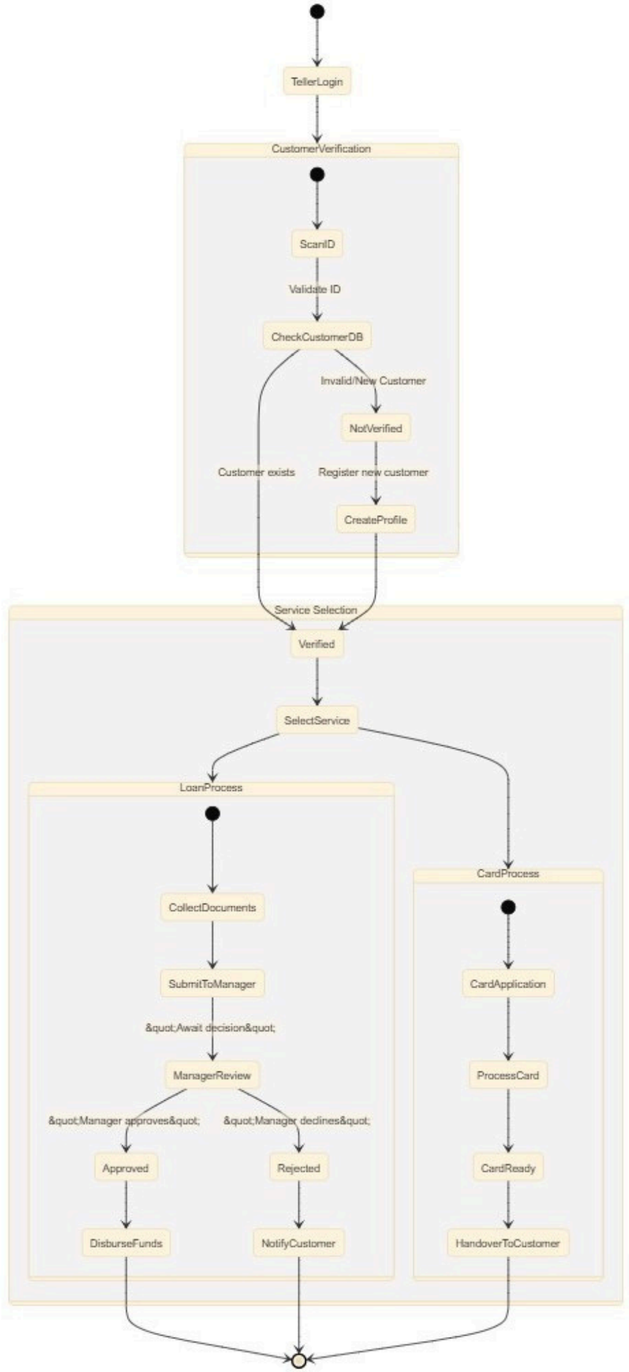
   (like fixed deposits)



State Diagram

Teller State

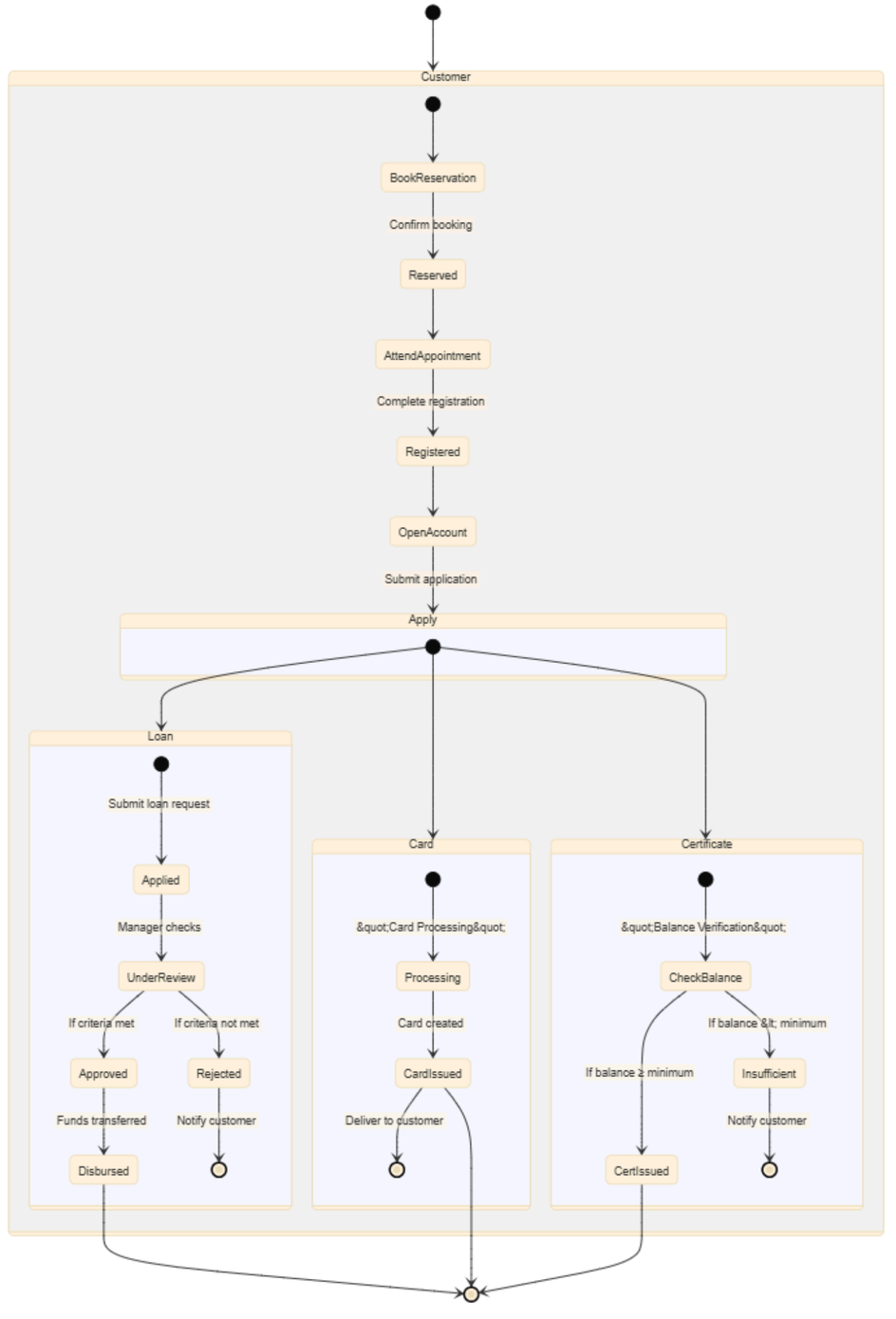
1. Account Opening – Create and verify new customer accounts (savings/current).
2. Tellers manage customer appointment bookings for banking services
3. Customer Support – Assist with inquiries, bookings, and basic issue resolution.



State Diagram

Customer State

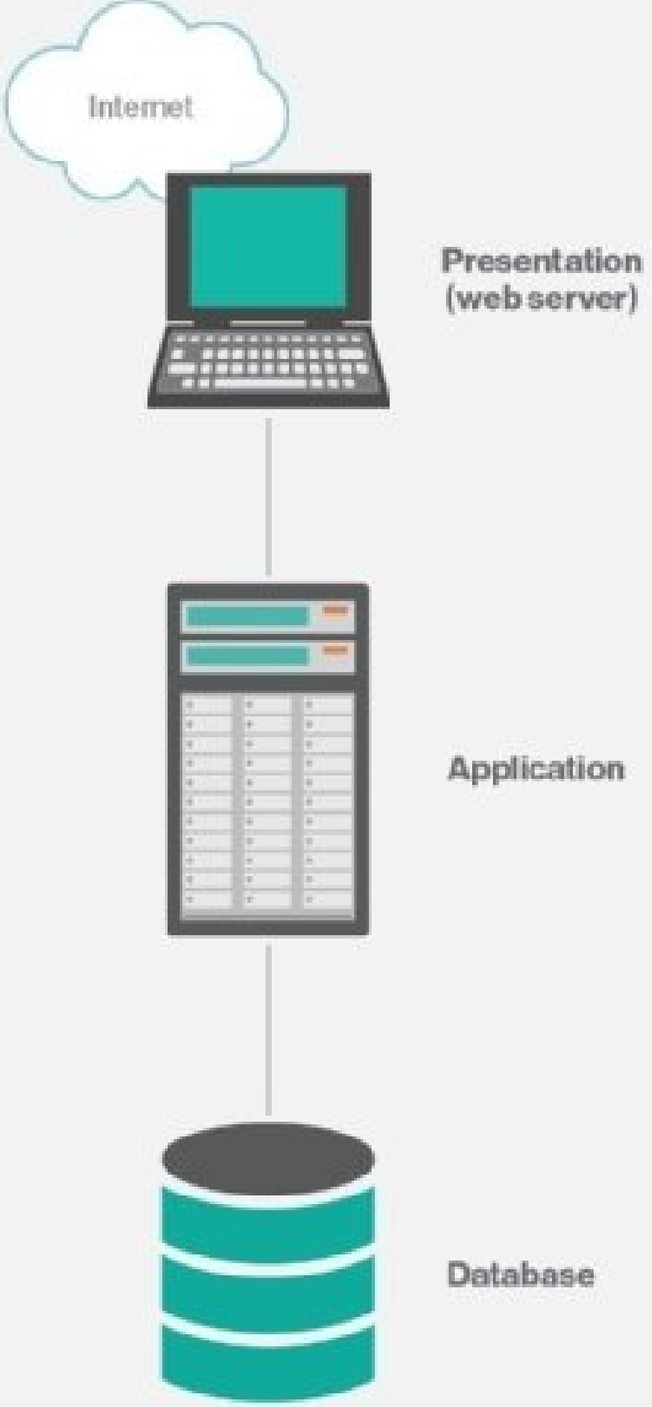
1. Booking & Registration – Customer schedules appointments and completes sign-up.
2. Service Selection – Chooses between loans (with approval checks), card requests, or balance-based certificates.
3. Result Handling – Receives automated updates on approvals/rejections or account balance issues.



Project Architecture

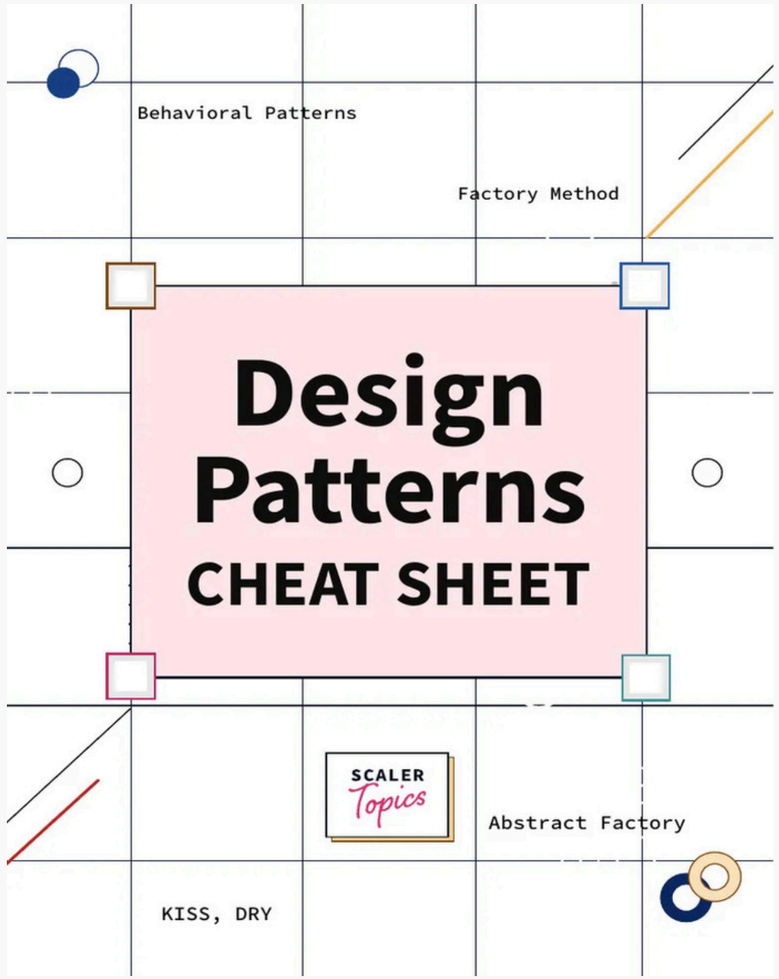
We followed the 3-layer architecture to ensure a clean separation of concerns, thiis structure enhances maintainability, scalability, and testability of our application.

1. **The Presentation Layer** handles user interactions.
2. **Business Logic Layer** processes data and applies core rules.
3. **Data Access Layer** manages database operations.



Design Patterns

- **Generic Repository Pattern**: Used to abstract and centralize all data access logic.
- **Unit of Work**: Ensures that all database operations across multiple repositories are committed in a single transaction.
- **Dependency Injection (DI)**: Applied to inject services and dependencies throughout the project in a loosely coupled way.



Live Demo

Future Work

1. Real-time Notifications.
2. Job Advertisement Management by Bank Managers
3. ATM Simulation Module.
4. Improved UI/UX Design.

Thank you