Bank Web Application

Aim

To develop a secure, role-based, full-stack web application for online banking using modern Java-based backend (Spring Boot) and a responsive React.js frontend.

Abstract

This project implements a multi-role online banking system that includes functionality for Customers, Employees, and Administrators. The backend leverages Spring Boot (version 3.1.5) with Java 17 for a robust, secure REST API, and MySQL as the relational database. The frontend is built using React 18.2.0, styled with Material-UI, and communicates with the backend using Axios. Key features include user registration, authentication, account management, money transfer, FD and loan management, and admin functionalities like user approval and job application handling.

Technology Stack

Backend Technologies

- Java Development Kit (JDK): 17.0.8
- Spring Boot Framework: 3.1.5
- Spring Data JPA: 3.1.5
- Spring Security: 6.1.5
- Hibernate ORM (JPA Provider): 6.2.9.Final
- MySQL Server: 8.0.34
- JDBC Driver: MySQL Connector/J 8.0.33
- PDF Generation: iText 7.2.5
- JSON Processing: Jackson 2.15.3
- Apache Maven: 3.9.x
- Embedded Server: Apache Tomcat 10.1.15

Frontend Technologies

- Node.js: 18.17.0 LTS
- npm: 9.6.7 or yarn: 1.22.19
- React: 18.2.0
- React DOM: 18.2.0
- React Router DOM: 6.16.0
- Axios: 1.5.1
- Material-UI:
  - @mui/material: 5.14.10
  - @mui/icons-material: 5.14.10
  - @emotion/react: 11.11.1
  - @emotion/styled: 11.11.0

User Roles and Key Features

Customer

- Registration and login
- Dashboard overview
- Fund transfers, check balance & transaction history (PDF supported)
- Apply for Fixed Deposits (FD) & Apply for Loans

Employee

- Login/logout functionality
- Manage deposit , withdrawal , balance check & transaction history

Administrator

- Login/logout functionality
- Approve/reject customer registration
- Handle FD and loan applications
- Manage job applications
- Inherits all employee features

Public Access

- Submit job applications without login

Project Structure

Backend Project Structure

- `config/` – Security and application configuration
- `controller/` – REST API controllers
- `dto/` – Data Transfer Objects
- `model/` – Entity classes
- `repository/` – JPA repository interfaces
- `request/response/` – Request and response format classes
- `security/` – Custom user details, filters, and auth logic
- `service/` – Core business logic
- `BankingSystemApplication.java` – Main entry class

Frontend Project Structure

- `App.js` – Main application entry
- `index.js` – App rendering
- `context/AuthContext.js` – Authentication state manager
- `components/` – Reusable UI components
- `pages/` – Page-level components:
  - Customer: Transfer, Balance, Transactions, Loans, FDs
  - Admin: Dashboard, Approvals, Applications
  - Employee: Dashboard, Deposit, Withdrawal
  - General: Login, Register, Job Application
- `utils/` – Utility functions
- `styles/` – CSS and theming

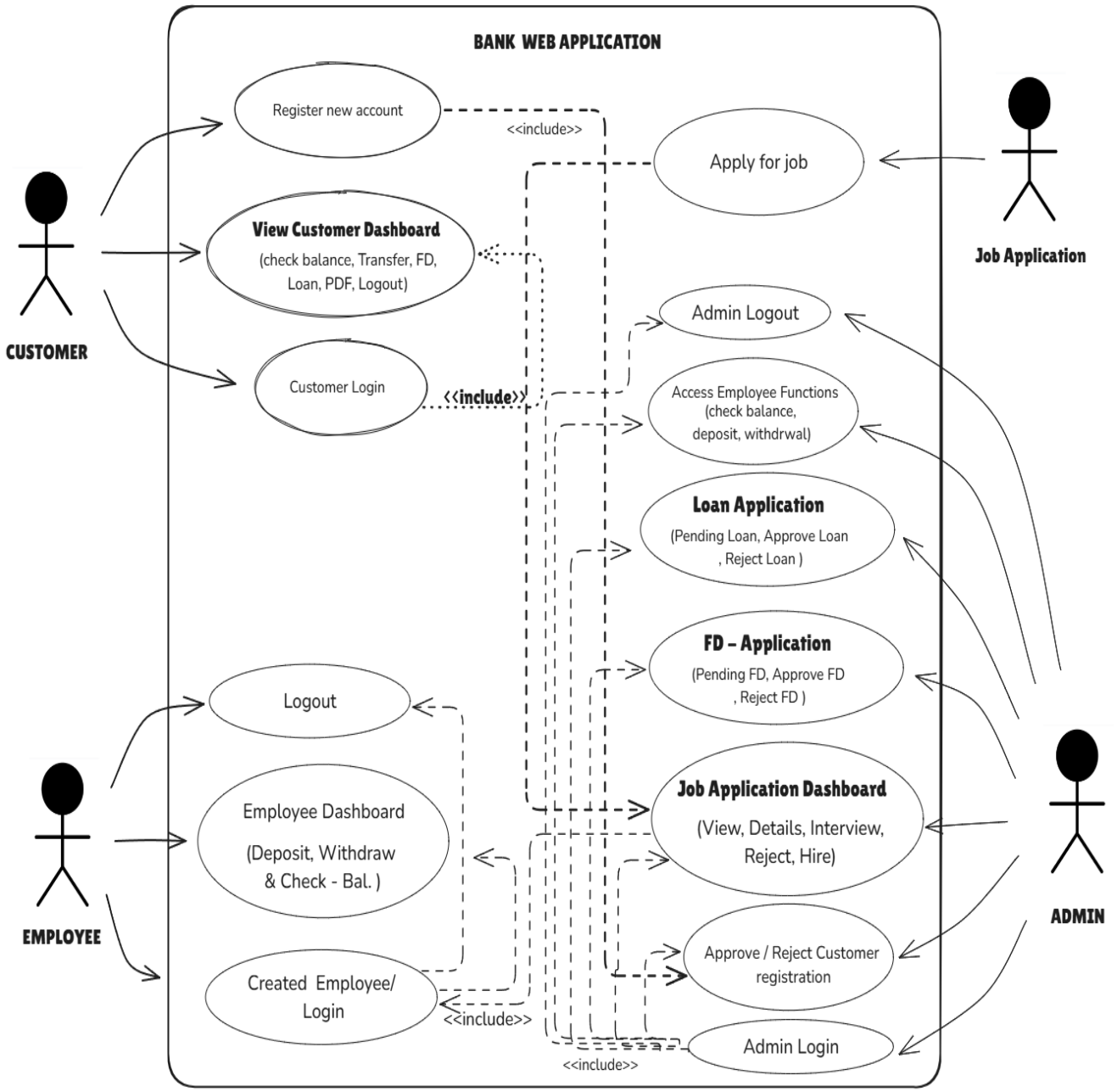

Figure 1: Usecase Diagram