**File Structure**

css

CopyEdit

bank-management-system/

│

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ └── com/

│ │ │ └── bank/

│ │ │ ├── BankManagementApplication.java

│ │ │ ├── controller/

│ │ │ │ ├── AccountController.java

│ │ │ │ ├── AuthController.java

│ │ │ ├── model/

│ │ │ │ ├── Account.java

│ │ │ │ ├── BankTransaction.java

│ │ │ │ └── UserLogin.java

│ │ │ ├── repository/

│ │ │ │ ├── AccountRepository.java

│ │ │ │ ├── BankTransactionRepository.java

│ │ │ │ └── UserLoginRepository.java

│ │ │ ├── service/

│ │ │ │ ├── AccountService.java

│ │ │ │ └── AuthService.java

│ │ └── resources/

│ │ ├── application.properties

│ │ └── static/

│ └── test/

├── pom.xml

**Step-by-Step Guide**

**1. Set Up Your Project**

1. **Install Java**: Ensure you have **Java 8** or above installed on your system.
2. **Install Maven**: Ensure **Maven** is installed. You can check using:

bash

CopyEdit

mvn -v

1. **Set Up Oracle Database**: Set up **Oracle Database** (either locally or remotely).
2. **Create Tables**: Run the following SQL commands in your Oracle database to create the required tables.

**SQL Schema (Create Tables with Intern\_ Prefix):**

sql

CopyEdit

-- Create INTERN\_CUSTOMER table

CREATE TABLE INTERN\_CUSTOMER (

ID NUMBER PRIMARY KEY,

NAME VARCHAR2(100),

EMAIL VARCHAR2(100),

PHONE VARCHAR2(15),

ADDRESS VARCHAR2(255),

DOB DATE

);

-- Create INTERN\_ACCOUNT table

CREATE TABLE INTERN\_ACCOUNT (

ID NUMBER PRIMARY KEY,

CUSTOMER\_ID NUMBER,

ACCOUNT\_TYPE VARCHAR2(20),

BALANCE NUMBER,

FOREIGN KEY (CUSTOMER\_ID) REFERENCES INTERN\_CUSTOMER(ID)

);

-- Create INTERN\_BANK\_TRANSACTION table

CREATE TABLE INTERN\_BANK\_TRANSACTION (

ID NUMBER PRIMARY KEY,

ACCOUNT\_ID NUMBER,

TRANSACTION\_TYPE VARCHAR2(20),

AMOUNT NUMBER,

TRANSACTION\_DATE DATE,

FOREIGN KEY (ACCOUNT\_ID) REFERENCES INTERN\_ACCOUNT(ID)

);

-- Create INTERN\_USER\_LOGIN table

CREATE TABLE INTERN\_USER\_LOGIN (

ID NUMBER PRIMARY KEY,

USERNAME VARCHAR2(50),

PASSWORD VARCHAR2(255)

);

**2. Create the Spring Boot Project**

1. **Generate the Project** using [Spring Initializr](https://start.spring.io/) with the following dependencies:
   * **Spring Web**
   * **Spring Data JPA**
   * **Spring Boot DevTools**
   * **Oracle JDBC**
2. Alternatively, **manual creation** of the pom.xml as shown below.

**3. Add pom.xml**

xml

CopyEdit

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.bank</groupId>

<artifactId>bank-management-system</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc8</artifactId>

<version>19.8.0.0</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**4. Configure application.properties**

In **src/main/resources/application.properties**, add your Oracle DB connection settings:

properties

CopyEdit

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=YOUR\_DB\_USERNAME

spring.datasource.password=YOUR\_DB\_PASSWORD

spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.Oracle12cDialect

**5. Source Code**

Here’s the **complete source code** for the application:

**BankManagementApplication.java (Main Class)**

java

CopyEdit

package com.bank;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class BankManagementApplication {

public static void main(String[] args) {

SpringApplication.run(BankManagementApplication.class, args);

}

}

**Model Classes**

* **Customer.java**

java

CopyEdit

package com.bank.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import java.util.Date;

@Entity

public class Customer {

@Id

private Long id;

private String name;

private String email;

private String phone;

private String address;

private Date dob;

// Getters and setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public Date getDob() {

return dob;

}

public void setDob(Date dob) {

this.dob = dob;

}

}

* **Account.java**

java

CopyEdit

package com.bank.model;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Account {

@Id

private Long id;

private Long customerId;

private String accountType;

private Double balance;

// Getters and setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Long getCustomerId() {

return customerId;

}

public void setCustomerId(Long customerId) {

this.customerId = customerId;

}

public String getAccountType() {

return accountType;

}

public void setAccountType(String accountType) {

this.accountType = accountType;

}

public Double getBalance() {

return balance;

}

public void setBalance(Double balance) {

this.balance = balance;

}

}

* **BankTransaction.java**

java

CopyEdit

package com.bank.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import java.util.Date;

@Entity

public class BankTransaction {

@Id

private Long id;

private Long accountId;

private String transactionType;

private Double amount;

private Date transactionDate;

// Getters and setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Long getAccountId() {

return accountId;

}

public void setAccountId(Long accountId) {

this.accountId = accountId;

}

public String getTransactionType() {

return transactionType;

}

public void setTransactionType(String transactionType) {

this.transactionType = transactionType;

}

public Double getAmount() {

return amount;

}

public void setAmount(Double amount) {

this.amount = amount;

}

public Date getTransactionDate() {

return transactionDate;

}

public void setTransactionDate(Date transactionDate) {

this.transactionDate = transactionDate;

}

}

* **UserLogin.java**

java

CopyEdit

package com.bank.model;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class UserLogin {

@Id

private Long id;

private String username;

private String password;

// Getters and setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**Controller Classes**

* **AccountController.java**

java

CopyEdit

package com.bank.controller;

import com.bank.service.AccountService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/accounts")

public class AccountController {

@Autowired

private AccountService accountService;

@GetMapping("/{accountId}/balance")

public Double getBalance(@PathVariable Long accountId) {

return accountService.viewBalance(accountId);

}

@PostMapping("/{accountId}/deposit")

public String deposit(@PathVariable Long accountId, @RequestParam Double amount) {

return accountService.deposit(accountId, amount);

}

@PostMapping("/{accountId}/withdraw")

public String withdraw(@PathVariable Long accountId, @RequestParam Double amount) {

return accountService.withdraw(accountId, amount);

}

}

* **AuthController.java**

java

CopyEdit

package com.bank.controller;

import com.bank.model.UserLogin;

import com.bank.repository.UserLoginRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/auth")

public class AuthController {

@Autowired

private UserLoginRepository userLoginRepository;

@PostMapping("/login")

public String login(@RequestParam String username, @RequestParam String password) {

UserLogin userLogin = userLoginRepository.findByUsername(username)

.orElseThrow(() -> new RuntimeException("Invalid credentials"));

if (userLogin.getPassword().equals(password)) {

return "Login successful!";

}

throw new RuntimeException("Invalid credentials");

}

}

**Service Classes**

* **AccountService.java**

java

CopyEdit

package com.bank.service;

import com.bank.model.Account;

import com.bank.model.BankTransaction;

import com.bank.repository.AccountRepository;

import com.bank.repository.BankTransactionRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.Date;

@Service

public class AccountService {

@Autowired

private AccountRepository accountRepository;

@Autowired

private BankTransactionRepository bankTransactionRepository;

public Double viewBalance(Long accountId) {

Account account = accountRepository.findById(accountId)

.orElseThrow(() -> new RuntimeException("Account not found"));

return account.getBalance();

}

public String deposit(Long accountId, Double amount) {

Account account = accountRepository.findById(accountId)

.orElseThrow(() -> new RuntimeException("Account not found"));

account.setBalance(account.getBalance() + amount);

accountRepository.save(account);

BankTransaction transaction = new BankTransaction();

transaction.setAccountId(accountId);

transaction.setTransactionType("DEPOSIT");

transaction.setAmount(amount);

transaction.setTransactionDate(new Date());

bankTransactionRepository.save(transaction);

return "Deposit successful. New balance: " + account.getBalance();

}

public String withdraw(Long accountId, Double amount) {

Account account = accountRepository.findById(accountId)

.orElseThrow(() -> new RuntimeException("Account not found"));

if (account.getBalance() < amount) {

throw new RuntimeException("Insufficient balance");

}

account.setBalance(account.getBalance() - amount);

accountRepository.save(account);

BankTransaction transaction = new BankTransaction();

transaction.setAccountId(accountId);

transaction.setTransactionType("WITHDRAW");

transaction.setAmount(amount);

transaction.setTransactionDate(new Date());

bankTransactionRepository.save(transaction);

return "Withdrawal successful. New balance: " + account.getBalance();

}

}

**Run the Application**

1. **Build and Run** the application using Maven:

bash

CopyEdit

mvn spring-boot:run

1. **Test API Endpoints**:
   * **Login**: POST /api/auth/login with username and password.
   * **Deposit**: POST /api/accounts/{accountId}/deposit with amount.
   * **Withdraw**: POST /api/accounts/{accountId}/withdraw with amount.
   * **View Balance**: GET /api/accounts/{accountId}/balance.