Finance Management System
Project Overview

By: K.A.L. Kavishika Kahanadawala

1. Implementation Overview

a. Login

Here login Happens with username and password. The virtual identity will be taken as "username". If user successfully logged in to the system, a session will be created. This session has a lifespan of a 10 minutes, considering system is about financial aspects. So as a security measure that was taken. Cookies won't be used, even secured ways like 'JWT cookies' which taken as industrial way, but considering this is on financial platform, client side cookies haven't taken.

b. Registration

Registration happens with email, username and password. Username will be used to identify user throughout the platform. When saving password in the database, special considerations must be taken. That's why the system uses 'password hashing'. Passwords will be hashed using an industrial method, 'bcrypt'. This happens using 'spring framework security' library. This hashed passwords will be taken for user login validation as well. Hashing makes even in a database leak, the thief can't get the original value back.

c. Account creation

This allows user to create accounts within the platform. User can have multiple accounts within platform on same category. Reasoning for this is,

Assumptions:

Normally within a bank environment, some allows to make multiple accounts. So even in saving accounts, I have made it that user can make multiple saving accounts. When an account made it will be made on status 0. This means account is not activated. Saving accounts will be activated as soon as made. But Loan and FD accounts won't activate till user selects how much time on next prompt screen. Reasoning is that if user somehow missed this 2nd page, which contains how much time, it would be a fatal thing in database part. As details user inputted in 1st screen are already saved. So after validation, accounts will be activated.

d. Transaction Operations

Transaction operations have for all 3 account types. Saving has a interface, mainly for withdraw and deposit.

Assumption:
Loans, I assumed that user will withdraw money as soon as they opened account, so this section doesn't have a specified interface for withdraw.

FD accounts have a withdraw option as well, But only can withdraw if user has passed amount of time. Otherwise can't.

e. Balance Inquiry
   Saving accounts and FD account both can see monthly interest earned portion as well.

f. Loan Repayment
   When loan repayment happens it will saved in a specific table called 'loan_repayment'. Reasoning is for this is, it can help count how much user paid. In the interface also user can see their monthly debt. Even if user has paid only portion or in several portions but still not his full debt, system will only record paid portion and will calculate show how much rest have to pay, when user revisit interface.

   Assumption:
   As in the real world too a user can pay more than his debt. Because of this, User can pay more than his debt within the system as well. When this happened user will see minus value on debt interface

g. Fixed Deposit Details
   Within this system user can see his monthly interest rate earned percentage. User can withdraw only when the times passes.

2. Other assumptions and overview

a. Project uses Apache tomcat 10. This was taken as it's the latest stable version (version 11 is still in alpha) and such updated approach would be beneficial for a finance field, security needed aspect. As a result javax.* is changed to jakarta.* .

b. This project is mainly made capturing client side perceptive. Because of this, some parts of the systems may not go with real world scenarios. Like this only calculate how much monthly loan user have to pay. But not if he has a debt on past month. Because these charges change according to time. And if these need to procedure correctly, a server side validations should happen manually or whatsoever time period, Because these operation in this system only executes when user interact with the system. Otherwise it won't. For example if user only paid for past month hand they never interact again within this system, it won't calculate, Such operations needs to be done by automatic scheduled services on server side. Mainly from developer side.
   The calculations like interest rate on FD for these difference months whatsoever will be calculated, but when it comes to loan side specially, system lacks such aspects because of this assignment is asked to for user side, so those lacks.

c. All rates are saved in 1 database table. Reasoning is this system is small. But when it comes to real-world, those should have separate tables. Tables have information like how much time and interest for these period likewise. But in real world, when all those made in same table, it's may not efficient. Specially when to change and and delete data. But if developer uses SQL queries like 'select within this account type' likewise, it won't be a problem that much.

d. System have validations when user interacts. Like when a user do some operation, like for example,

   Let's say a user select his saving accounts. Then on next page he can check how much he have. So some of these data are passe by url attributes. For example even the amount. As this is a web Application user can edit those. And these url attributes will be used in some methods as well. Like even for form submissions as well. For example user details takes from url, then they will added as hidden info in form, then those will be passed to servlet. When But whenever that happens, system checks whether these attributes like this username and password, are true. Like if there's a user that's actually has that username and that account number. Only when that happens system will be forwarded. Otherwise user will get a error page and no database executions will be happen in the meantime.

e. For the user friendliness environment, system will be check if that user exists in the platform as well. Then user will be informed accordingly.

f. When an FD withdraw happens, amount from DB will makes '0' as the balance. I assumes that withdrawal on that situation is whole portion

g. When a user creates a loan or FD account, interest of that time will be saved accordingly in the table as well. Reasonings is I assumed if user made those account, interest rate for those will be same throughout the account period. For example if user, creates a FD account with 12% interest rate, I assumed that user should get that value as the interest rate. Even if interest rate changes later on, it won't be changed. That's the reason to insert them as well in the project not only, taking rate from 'rates' table, when executions happens. That way, even when the interest rate changed later, the interest rate bank and user agreed at the 1st place will be same for the rest of the account time of that particular loan or FD account.

h. If user session expired or if user not logged in an user trying to access a location which only logged in users can view, User will redirect to login page asking to login. So user can't access those

i. If 1 of login credentials are wrong, error page will appear. But it won't say which is wrong whether username or the password. This has taken as a safety measure for others. If another user trying another one's credentials , by doing this they can't even find

whether that username is in the platform or not. This enhances the user security. I've seen this even when I try to login to my internet baking account, they won't provide which is wrong one. So I implement the system accordingly as well.

j. Some of the database tables may seems like it's normalized. There's a reason for that. This is a huge database platform. Thos can have so many data inside. So if it's normalized that way, for some operations it might make it slow. This comes in 'database tuning' part. So some tables may not seems like not normalized but the reason is it may result slower in database and impact the database tuning process.

k. The account tables has id and account_id. This id is the primary key. Account_id is the bank account number. As this system allows user to make multiple accounts on same account type, account_id can't be taken as the primary key.

l. For this platform I generate random number as a account id. These doesn't have pattern. The reasoning I went through that approach is that I only meant to demonstrates the system. In real world, we can use some pattern like, "loan pattern + sequence number+ branch info". Likewise pattern.

m. Database has separate table for loan repayment. The reason is to make it easier calculating loan paid amount. This helps even when a user paid only some portions, it would easier to calculate. Reason not putting to transactions table is because of differentiates and mainly for database tuning. And for making data easy to handle as well this table can be taken as a way of log data on loan repay.

n. There's an entity called 'rate' in the ER. This table is to store rates of each account and for what time period. That's why it taken as a separate entity.

o. System uses database pool way. This is much more safe and better for proposed system. Credentials for the db are taken as env passing.

p. In the uml class diagram jsp files are not included. As those are not directly classes even though they are views, those are neglected in the diagram.

q. System has server side validation on registration. As client side can be manipulated by user and when such data entering to server side it may generate issues, thus such approach taken. As server side validation is happening, client side validation in those covered areas are not taken and only happens in server side.

r. The SQL file has some data inside it. These are for representation only. Which some mainly used while testing. Credentials for the most used account is as following. (As password can't see due to hashed)

Login: Kavishika
Password: 1