~~1 Aim:~~

The aim of this assignment is to develop a web application using **Spring** framework and JDBC/Hibernate. There will be a sub-component that must be built using **AngularJS**.

This assignment simulates a real-world application, *you can always present it as a part of your portfolio during job interviews*. Doing a good job of this assignment will require research, extra reading and following lectures and tutes closely.

2 Problem Description:

Your team has been hired by ACME Bank to design and develop their new banking system, which they expect to be operational within three months. Below, ACME Bank have provided you with a brief outline of what their day-to-day business consists of, in order to provide you with some background knowledge in the area of banking systems:

Banking systems can be considered to be one of the largest e-commerce & enterprise systems that are currently in operation. These systems are designed to handle millions of dollars per day, without losing track of any money that passes through their system.

The role of a bank is to provide credit (borrow out money) to customers who do not have enough money to make a purchase. In return, customers pay the bank interest on the money they have borrowed. Banks however, cannot just borrow out money to customers, they first need customers with an excess of money (savings) with the bank, which the bank can then borrow out to their other customers.

This is a simplified explanation of how a bank works for the purposes of your assignment. Banks and banking systems are much more complex than the one we have explained here and will be building in this course. This is to make your life simpler☺

With a brief idea of how a bank works, your team now has to design and implement the necessary software systems for the ACME Bank. The software systems that are required have been detailed in this document. Although ACME Bank is new, they already have a small team of software design experts, who have prepared this design specification.

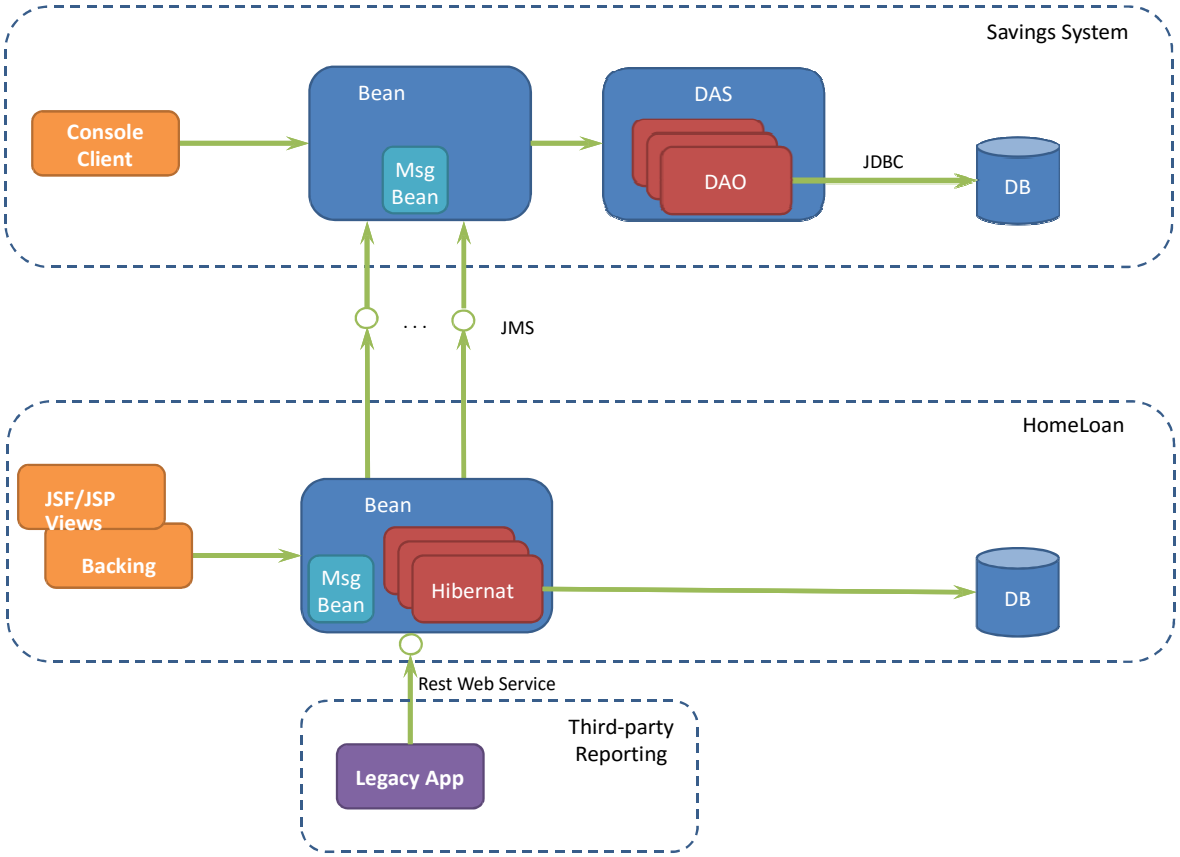
In addition, the software experts at ACME Bank know that developing such large complex systems is no easy task, so they have requested that your team complete the banking system in two parts- AngularJS part and Spring part.

For each part, we would like you to demonstrate the system working in week 12 in a face to face demo.

We have provided a diagram of the overall system architecture to illustrate the components within the system.

Please feel free to make changes in this architecture but you must justify the changes during face to face demo session.

Sub-system 1: Savings System and Sub-system 2: HomeLoan



Looking at the diagram above, the sub-system is designed using a three tier architecture. The DB and DAS (Data Access Service) form the database tier. The bean component forms the Composite Service (Business Logic) middle tier and the Console Client will be used for the presentation tier. We leave it for you to decide the types of beans that are needed.

To manage the data, we will use Beans and the Data Access Objects (DAOs) and Java Database Connectivity (JDBC).

The Client will be console-based and will be used to perform the high-level use-cases specified on page 4. That is, for each of the use-cases below, your console client must provide a way to execute each function. A simple menu driven system would work well. As an example, for the withdraw functionality, the console client could provide a menu option for withdraw that get's the account ID and amount from the user, and then makes a call to the necessary functionality in your session beans to make the withdrawal. Note that the main purpose of this client is to demonstrate your project to the demo markers, so keep it simple but functional. Do not implement a graphical user interface client here.

3 Database Layer:

The following is an *initial database schema* that you will need to use as a starting point. That is, you must implement the following schema as a minimum. **You are free however, to extend or make minor changes to the schema as you see fit**.

The schema only outlines the required fields, it does not state what the data-types for each field should be. This is for you to decide.

Table	Fields
Employee	E_ID, First Name, Last Name
Customer	C_ID, First Name, Last Name, DOB,
Savings	C_ID, AccNum, Balance
Transaction	T_ID, AccNum, Amount, Description

Table Specification Details:

ID Fields: E_ID, C_ID, AccNum and T_ID

The **E_ID**, **C_ID**, **AccNum** and **T_ID** fields are to uniquely identify employees, customers, savings accounts and transactions respectively. **E_ID**, **C_ID**, **AccNum** may have a similar format to student numbers (S12345678). For example, a typical **E_ID** might be *E112*.

Savings Table:

- **AccNum** account number.
- **Balance** represents the amount of money the customer owns in this account
- **Amount** represents the dollar amount deposited or withdrawn.
- **Description** represents a description of the transaction. For example, withdrawal and deposit.

4 Business Rules:

In this section, the business rules for the ACME banking system your team is required to implement are specified.

- A maximum of two savings accounts may be opened by each customer.

- For auditing purposes, all transactions (deposits and withdrawals) need to be recorded. All customer transactions are stored in the same table, however in the GUI, a customer should only be able to view transactions that belong to their account.

- At ACME bank, employees are only allowed to perform 5operations before they are required to take a break. To keep track of this, the employee must be able to see how many operations they have performed since login. After 5 operations, the system must invalidate the server session preventing the employee from continuing on without being logged out and logging back in. Upon login, the number of operations performed can be reset.

- For security reasons, after 1 minute of in-activity (no operations performed on the server), the session must be timed out and the employee must be forced to logout and login in again.

More information for Sub-system 2:

Table	Fields
HomeLoan	C_ID, AccNum, AmountBorrowed, AmountRepayed

Table Specification Details:

- **AmountBorrowed** represents the total amount of money the customer has borrowed for this home loan.
- **AmountRepayed** represents the total amount of money the customer has repayed for this home loan.

Business Rules:

- A Home Loan account may be opened only when the customer has opened a savings account. To open a Home Loan account, customers must prove that they can make regular deposits. ACME Bank will only accept three or more deposits into their savings account as the proof that they have some form of income.
- Creating a new home loan account must make use of a wizard style entry, where by, at each stage of the wizard, the state of the details entered are maintained in a session on the server. The details that each wizard screen will need to accept and therefore for the system to store can be found in the GUI section below.
- The URL of the rest web service should be /investors/stats.

5 Assignment tasks:

The assignment is divided into 4 parts – Pass, Credit, Distinction and High-Distinction. Attempt as much as you can, depending upon the time available and inclination/motivation.

PA part [35 marks]

1. (10 marks) Project manager has asked the team to develop an SPA **AngularJS** application to keep a track of the project development. This application must have the following features-

    a. (1 mark) Initial discussion plans including requirements analysis of the banking website
    b. (1 mark) Use cases, Class diagram
    c. (1 mark) Team details and skill summary of each of the members
    d. (1 mark) Issues encountered during software development and how they were resolved
    e. (2 marks) Timeline displayed in a graphical format (*ie how the project progressed*)
    f. (2 marks) Wireframes of the application
    g. (1 mark) Add search/filter options to search any of the above information. The project manager should be able to track the project by viewing and searching the relevant data for any of the above options.

    (1 mark) Look and appeal: In order to get full-marks on this part, make your AngularJS SPA as interactive as possible.

2. Savings System (25 marks): Using **Spring** framework and **JDBC/Hibernate** create a console-based application that has the following features-

    a. (3 marks) Employee login before any other functionality is available. (Employee accounts are hard-coded into the database)
    b. (4 marks) Create a customer
    c. (4 marks) Open a Savings account
    d. (4 marks) Make deposit into Savings account
    e. (4 marks) Make withdrawal from Savings account
    f. (3 marks) View balance of Savings account
    g. (3 marks) Employees should have the number of operations performed per session displayed above the menu. Operations include all of the above major use-cases except login.

→

CR part [07 marks]

3. Customer Home Loan Graphical application: Using **Spring** framework and **JDBC/Hibernate** create a GUI application that has the following features-

    a. (3 marks) Customer login
    b. (4 marks) Create a Home Loan account (Amount borrowed entered during account creation)

DI part [07 marks]

4. Extend the requirement 3 by adding the following features-

    a. **Changed specification:** (4 marks) Add a feature to the website that displays information about a different type of account known as **Investment account**.

    Each user may have more than one investment account, which they can use to make payment(s) towards Loan account.

    Make a repayment to a Home Loan account. Money must come from Investment account. Use JMS to complete this part.

    b. **Changed specification:** (3 marks) List all customer accounts in a professional tabulated manner. (Investment accounts & Home Loan).

HD part [21 marks]

Note: Some of the specifications under this part are challenging, you will have to do some self-exploration and research of your own to complete them!!

5. **Changed specification:** (10 marks) Expose a rest web service that a third party system can use to retrieve investor data. The web service should return the total amount of money the bank has from all of its Investment accounts. This will need to use JMS.

6. (5 marks) Thorough unit testing code in the project

7. (6 marks= 2 + 2 + 2) Code elegance

    a. Code commenting
    b. Code elegance, separation, use of Java patterns
    c. Professional interface for sub-systems 1 and 2