Collateral Loans – Risk Assessment

Table of Contents

1.0 Important Instructions

1. Associate must adhere to the Design Considerations specific to each Technolgy

Track

2. Associate must not submit project with compile-time or build-time errors
3. Being a Full-Stack Developer Project, you must focus on ALL layers of the application development
4. Unit Testing is Mandatory, and we expect a code coverage of 100%. Use Mocking Frameworks wherever applicable.
5. All the Microservices, Client Application, DB Scripts, have to be packaged together in a single ZIP file. Associate must submit the solution file in ZIP format only
6. If backend has to be set up manually, appropriate DB scripts have to be provided along with the solution ZIP file
7. A READ ME has to be provided with steps to execute the submitted solution, the Launch URLs of the Microservices in cloud must be specified.

   (Importantly, the READ ME should contain the steps to execute DB scripts, the LAUNCH URL of the application)
8. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable
9. You are supposed to use an In-memory database or sessions as specified, for the Microservices that will be deployed in cloud. No Physical database is suggested.

2.0 Introduction

2.1 Purpose of this document

The purpose of the software requirement document is to systematically capture requirements for the project and the system “Collateral Loans – Risk Assessment System” that has to be developed. Both functional and non-functional requirements are captured in this document. It also serves as the input for the project scoping.

The scope of this document is limited to addressing the requirements from a user, quality, and non-functional perspective.

High Level Design considerations are also specificed wherever applicable, however the detailed design considerations have to be strictly adhered to during implementation.

2.2 Project Overview

The term "collateral" refers to any asset or property that a consumer promises to a Bank as backup in exchange for a loan. Typically, collateral loan agreements let the Bank to take over the asset if the borrowers fail to repay the debt according to the loan contract.

This Collateral can be any item of value that is accepted as an alternate form of repayment in case of default. If loan payments are not made, assets can be seized and sold by banks. This ensures that a lender receives full or partial compensation for any outstanding balance on a defaulted debt.

The Bank wants to develop a Microservies based middleware layer, for the core components like:

1. Loan Management
2. Collateral Management
3. Risk Assessment

There will be an MVC based back office portal called “Collateral Loans” portal, that will allow the back office staff to continuously monitor the risks (for both consumer and as well as bank) for every loan issued by the bank.

As part of enhancement to the application, the bank wants to enable its customers with the following options

I. Apply for loan

II. Check loan status

It should also enable the bank to approve/reject loan application.

2.3 Scope

Below are the modules that needs to be developed part of the Project:

Req. No.	Req. Name	Req. Description
REQ_01	Loan Management	Loan Management Module is a Middleware Microservice that performs following operations: • Get the Loan Details • Save Collaterals for Sanctioned Loan • Apply Loan • View Loan application status • Approve/Reject Loan application
REQ_02	Collateral Management	Collateral Management Module is a Middleware Microservice that performs the following operations: • Verify and Save the Collaterals • Get Collaterals
REQ_03	Risk Assessment	Risk Assessment Module is a Middleware Microservice that performs the following operations: • Raise Risk based the Collaterals Value based on Market
REQ_04	Collaterals Loan Portal	A Web Portal that allows a Back Office Staff to Login and allows to do following operations: • Login • Get Loan Details based on customer / loan ID • Save Collaterals for a sanctioned Loan • View Risk Assessment for every Collateral Loans (for Bank)

Note: The project phase is for 2 weeks. The first week is to be developed on local machine and the second week deals with Cloud deployment.

The requirement details given below states in-memory database usage. **The first phase of the development which is done in the first week,**

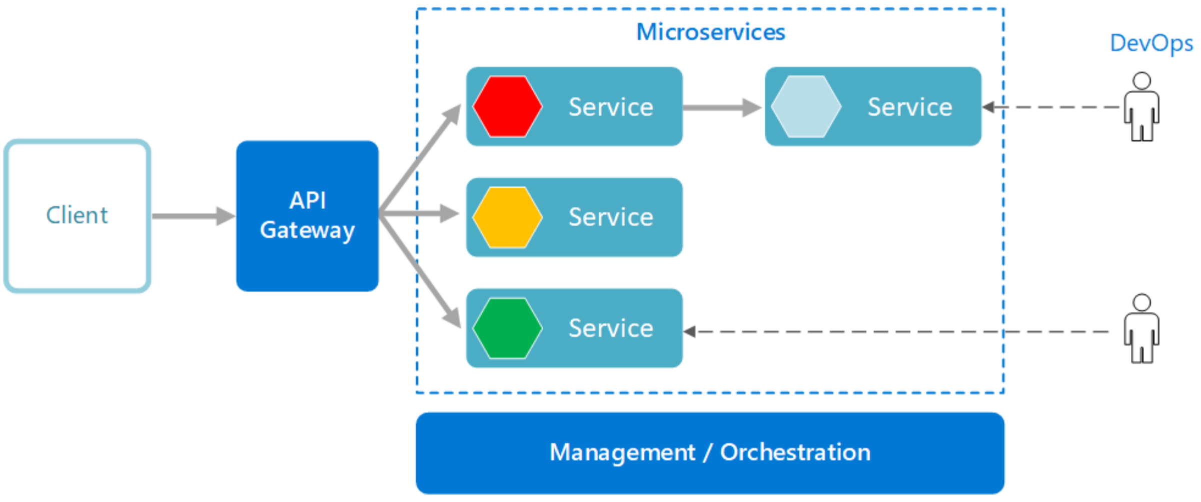
SHOULD use the Database for related activities and NOT the in-memory database.

The second phase of the development which is done in the second week, can use the in-memory database as mentioned in the requirement, with appropriate code modifications.

2.4 Hardware and Software Requirement

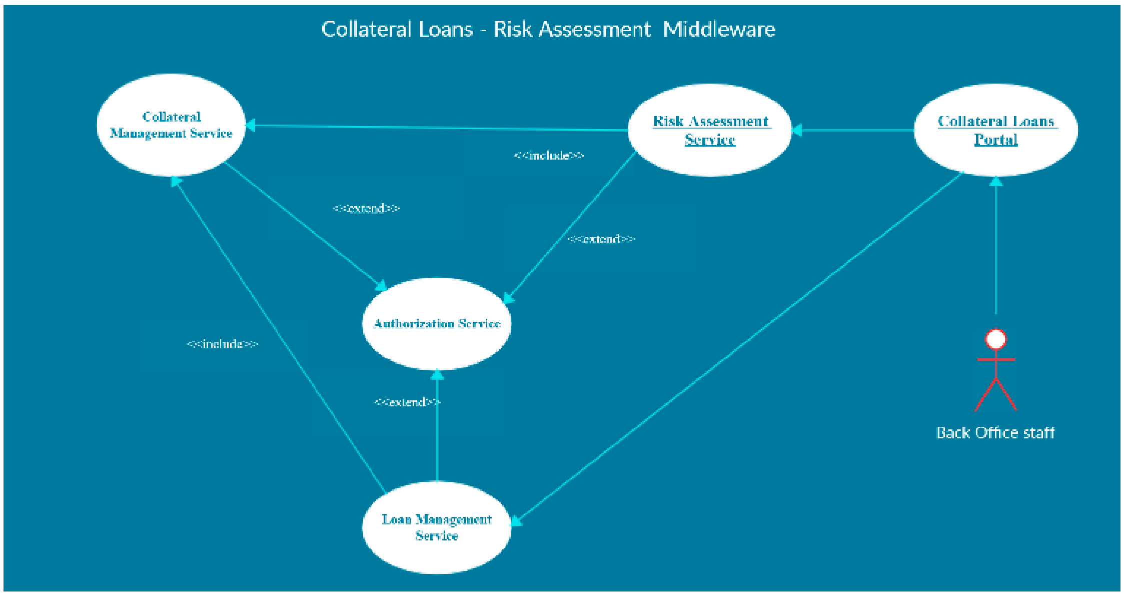
1. Hardware Requirement:
   a. Developer Desktop PC with 8GB RAM
2. Software Requirement (Java)
   a. Spring Tool Suite (STS) Or any Latest Eclipse
      i. Have PMD Plugin, EclEmma Code Coverage Plugin and AWS Code Commit Enabled
      ii. Configure Maven in Eclipse
   b. Maven
   c. Docker (Optional)
   d. Postman Client in Chrome
3. Software Requirement (Dotnet)
   a. Postman Client in Chrome

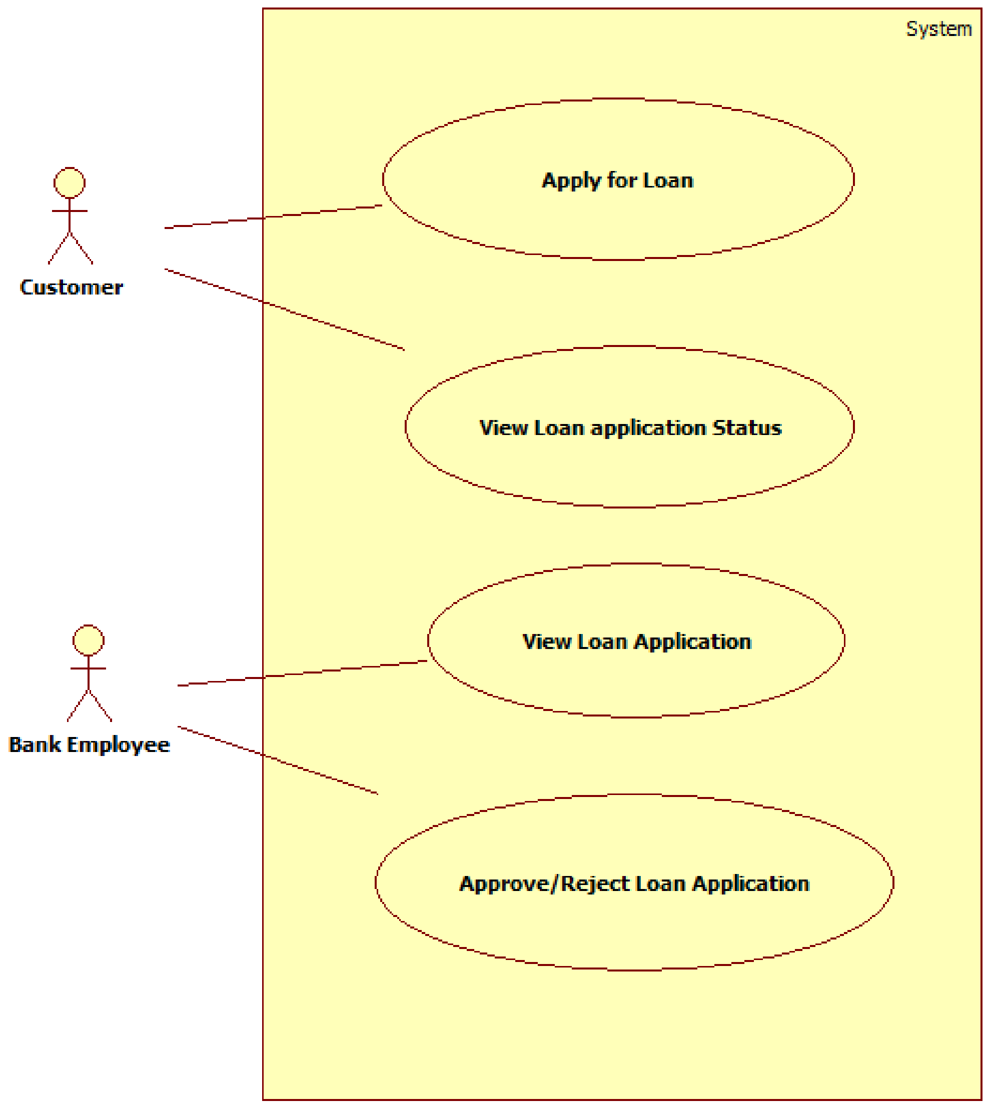
2.5 System Architecture Diagram



3.0 Functional Requirements and High Level Design

3.1 Use Case Diagram





3.2 Individual Components of the System

3.2.1 Collaterals Management Microservice

Collateral Loans – Risk Assessment	Collaterals Management Microservice
Functional Requirements This Microservice will be invoked by Loan Management Service and Risk Assessment Service. When invoked by Loan Management Microservice, • Authorization will be performed • For a sanctioned loan, save the collateral details for the loan ID. When invoked by Risk Assessment Microservice, • Authorization will be performed • While assessing a risk, the Risk Assssment Microservice wil invoke the Collateral Management Microservice to retrieve the latest collateral details	
Entities	

1. **Collateral_Loan**

   <Details of Collateral like Loan ID, Collateral ID, Collateral Value, Pledged Date etc.>

Few Entities based on type of collaterals can be created, given below are examples.

2. **Collateral_RealEstate**

   <Loan ID and Details of Collateral like Collateral Type, Owner Details, Address, Current Value, Rate Per Sq.Ft., Depreciation Rate etc.>

3. **Collateral_CashDeposits**

   < Loan ID and Details of Collateral like Collateral Type, Owner Details, Bank Name, Current Value, Interest Rate, Deposit Amount, Lock Period etc.>

REST End Points

Collaterals Management Microservice

- GET: /getCollaterals (Input: Loan_ID, Customer_ID | Output: Collateral ID, Type, Collateral Details based on each Type)
- POST: /saveCollaterals (Input: Loan_ID, Collateral_ID, Collateral Type, Details of the Collateral based on Type) | Output: Status)

Note: can also have different POST calls for different type of collaterals, or can be in single POST call.

Trigger – Can be invoked from Loan Management Service or from Risk Assessment Service

Steps and Actions

1. Once a request is received to Collaterals Service, Authorization will be performed
2. If /getCollaterals end point is invoked the corresponding collateral details of the loan, has to be returned.
3. If /saveCollateral end point is invoked, then the Microservice will save the collateral based on appropriate type.

Non-Functional Requirement:

- Only Authorized Back Office Staff can access these REST End Points
- If the getCollateral request is received multiple times for the same Loan ID, there must not be multiple hits leading to performance issues.

3.2.2 Loan Management Microservice

Collateral Loans Risk Assessment	Loan Management Microservice

Functional Requirements

Loan Management Microservice will be invoked from Collateral Loans Portal. This Microservice will be invoke Collateral Management Service for saving the Collaterals related to the loan.

To view the loan details:

- Retrieve the Loan Details based on Loan ID

To save the collaterals for the sanctioned loan:

- This will in turn reach the Collateral Management Microservice and saves the collateral details

To Apply for Loan.

- This will enable the customer to apply for loan

To View the loan application status

- This will enable the customer to view the loan application status

To Approve/Reject loan application

- This will enable the authorized bank employee to approve/reject the loan application by evaluating the merit of loan application like collateral details etc.
- If the application is approved, the status of the loan application to be updated with “approved” status, else the loan status to be updated with “Rejected” status.
- Once the loan is approved, the loan details to be updated in the database. Also, the microservice needs to communicate with the Collateral Management Microservice to update the collateral details in the database.
- Verifying the loan application is a manual process. Based on the verification, loan application can be approved or rejected.

Entities

1. **Loan**

   <Loan Products, Maximum Loan eligible, Interest, Tenure, Type of Collateral Accepted etc.>
2. **Customer_Loan**

   <Loan Product ID, Loan_ID, Customer_ID, Loan_Principal, Tenure, Interest, EMI, Collateral ID etc.>
3. **Customer**

   <Details of the Customer>
4. **Loan_Application**

   <application_id, customer _id, loan amount, tenure, collateral Details, status>

REST End Points

Loan Management Microservice

- GET: /getLoanDetails (Input: Loan_ID, Customer_ID | Output: Loan ID, Sanctioned Loan Amount, Tenure, Interest, Collateral ID pledged etc.)
- POST: /saveCollaterals (Input: Loan_ID, Collateral_ID, Collateral Type, Details of the Collateral based on Type) | Output: Status)
- POST: /applyLoan(Input: Loan_Application) | Output : Status
- GET:/ getLoanApplicationStatus (Input: application_id | Output: Loan_Application)
- PUT: /approveLoanApplication (Input: application_id | Output: Status)
- PUT: /rejectLoanApplication (Input :application_id | Output: Status)

Trigger – Can be invoked from Collaterals Loan Portal, however for saving collaterals, this will in turn invoke the Collateral management microservice.

Steps and Actions

1. Loans Management Microservice will have 2 End Points exposed to Collaterals Loan Portal used by Back Office.
2. If /getLoanDetails end point is invoked, the Microservice will return the appropriate loan details
3. If / saveCollaterals end point is invoked, then the Microservice will invoke the Collaterals Management Microservice for saving the collateral details of the sanctioned loan.

Non-Functional Requirement:

- Saving the Collaterals for every input request must happen in parallel

3.2.3 Risk Assessment Microservice

Collateral Loans Risk Assessment	**Risk Assessment Microservice**
Functional Requirements Risk Assessment Microservice interacts with Collaterals Management Microservice. Collaterals Loan Portal will invoke the services of Risk Assessment Microservice to portray the risks of the collaterls held by the bank ○ View Risks of Collaterals held by the bank	
Entities **1. Collateral_Risk** <Risk Details of each Collateral pledged, % Risk, Date Assessed, Market Value, Sanctioned Loan etc..> **2. Collateral_Market_Value** <market Value of each collateral ex: Rate / Square feet of land in dfferent areas, Intereset rate of a Fixed Deposit in a Bank.> - Assume that this will be refreshed in the system using a flat file, whenever there is a change. However there must ben an option to refresh the values using a file. **REST End Points** **Risk Assessment Microservice** ○ GET: /getCollateralRisk (Input: Loan_ID | Output (Risk Percent, Date of Risk Assessed)	
Trigger – Can be invoked from Collaterals Loan Application. Howeve this Microservice need to access Collaterals Management Microservice for getting the details of collaterals pledged for a given Loan ID.	
Steps and Actions 1. Risk Assessment Microservice will have only one end point exposed to Collateral Loans Portal ○ Based on Collateral Market Value for the pledged collateral, whenever the market value goes down beyond the loan sanctioned, then a risk alert must be raised.	

3.2.4 Authorization Microservice

Collateral Loans Risk Assessment	**Authorization Microservice**
Security Requirements ○ Service to Service communication has to happen using JWT ○ Pass End User Context across Microservices ○ Have the token expired after specific amount of time say 15 minutes. ○ Have this service configured in the cloud along with other services	

3.2.5 **Swagger**

Collateral Loans Risk Assessment	Swagger
Documentation Requirements ○ All the Microservices must be configured with Swagger for documentation **For Java based Implementation:** ○ Register the swagger resources in the Swagger Microservice and enable them as REST end points ○ Configure this service along with other services in the cloud	

3.2.6 Collateral Loans Portal

Collateral Loans Risk Assessment	Collateral Loans Portal
Client Portal Requirements There are 2 kinds of users for the portal, 1. Bank Back-office staff login 2. Customer login ○ Collateral Loans Portal must allow a Back Office Staff to Login. Once successfully logged in, the staff should be redirected to the staff home page where he/she can do the following operations: ○ View Loan Details ○ Save Collaterals of a Sanctioned Loan ○ View Risks associated with the collateral ○ View Loan application ○ Approve/Reject Loan application ○ Collateral Loan portal should also enable is customers login. Once successfully logged in, he/she should be redirected to the home page of the customer where he/she can do the following operations. ○ Apply for loan ○ View loan application status ○ Each of the above operations will reach out to the middleware Microservices that are hosted in cloud.	

4.0 Cloud Deployment requirements

- All the Microservices must be deployed in Cloud
- All the Microservices must be independently deployable. They have to use In-memory database or user sessions wherever applicable
- The Microservices has to be dockerized and these containers must be hosted in Cloud using CI/CD pipelines
- The containers have to be orchestrated using AWS/Azure Kubernetes Services.

- These services must be consumed from an MVC app running in a local environment.

5.0 Design Considerations

Java and Dotnet specific design considerations are attached here. These design specifications, technology features have to be strictly adhered to.

CDE-Project-Design Considerations.pptx

6.0 Reference learning

Please go through all of these k-point videos for Microservices deployment into AWS.

Microservices deployment into Azure Kubernetes Service.

AzureWithCICD-1
AzureWithCICD-2
AzureWithCICD-3
AzureWithCICD-4

Other References:

Java 8 Parallel Programming	https://dzone.com/articles/parallel-and-asynchronous-programming-in-java-8
Feign client	https://dzone.com/articles/Microservices-communication-feign-as-rest-client
Swagger (Optional)	https://dzone.com/articles/centralized-documentation-in-Microservice-spring-b
ECL Emma Code Coverage	https://www.eclipse.org/community/eclipse_newsletter/2015/august/article1.php
Lombok Logging	https://javabydeveloper.com/lombok-slf4j-examples/
Spring	https://dzone.com/articles/spring-boot-security-json-web-tokenjwt-hello-world

Security	
H2 In-memory Database	https://dzone.com/articles/spring-data-jpa-with-an-embedded-database-and-spring-boot https://www.baeldung.com/spring-boot-h2-database
AppInsights logging	https://www.codeproject.com/Tips/1044948/Logging-with-ApplicationInsights
Error response in WebApi	https://stackoverflow.com/questions/10732644/best-practice-to-return-errors-in-asp-net-web-api
Read content from CSV	https://stackoverflow.com/questions/26790477/read-csv-to-list-of-objects
Access app settings key from appSettings .json in .Netcore application	https://www.c-sharpcorner.com/article/reading-values-from-appsettings-json-in-asp-net-core/ https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-3.1

7.0 Change Log

	Changes Made			
V1.0.0	Initial baseline created on <24-Jul-2020> by <Srilakshmi Jayaraman>			
V2.0.0				
	Section No.	**Changed By**	**Effective Date**	**Changes Effected**
	2.2 2.3 3.1 3.2.2 3.2.6	Reni Varghese	25-Feb-2021	Enhancement suggested to include the following functionalities 1. View Loan Application 2. Approve/Reject Loan application 3. Apply for loan 4. View application status