Loan Management System

​​**Contents**

**​**[**1.0** **Problem statement** **1**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122916)

**​**[**2.0** **Skills to develop the project** **2**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122917)

**​**[**3.0** **Architecture Diagram for the Problem Statement** **2**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122918)

**​**[**4.0** **User Stories** **3**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122919)

**​**[**5.0** **Expected Deliverables** **4**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122920)

**​**[**6.0** **Milestone and duration** **4**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122921)

**​**[**7.0** **Implementation Notes** **5**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122922)

**​**[**8.0** **Evaluation rubrics** **6**](file:///C:\Users\427565\Downloads\CDE_Frontend-Angular+Backend-Java%20+%20AnthosKF-%20Project%20UseCase%20Template%2001%20-%2030032022%20(1).docx#_Toc43122923)

​​

1. **Problem statement**

Loan management system manages the process of originating Loan orders, modify/update and cancellation of loan related information. The Loan information accounts for the loan number, loan amount, loan term, borrower information, property information, status, loan management fees, origination date, origination account, lien information, legal documents and loan history.

The main components that make up the loan are as follows:

1. Borrower information: details of the person or party borrowing the loan amount
2. Property information: details of the property for which loan is originated
3. Lien information: information related to the taxes and bill amounts liable to be paid to various taxing authorities
4. Legal documents: legal documents related to the property and ownership related information

The scope of this project is to deploy an application using the latest state-of-the-art technology stack on Google Cloud platform Anthos KF.

1. **Skills to develop the project**

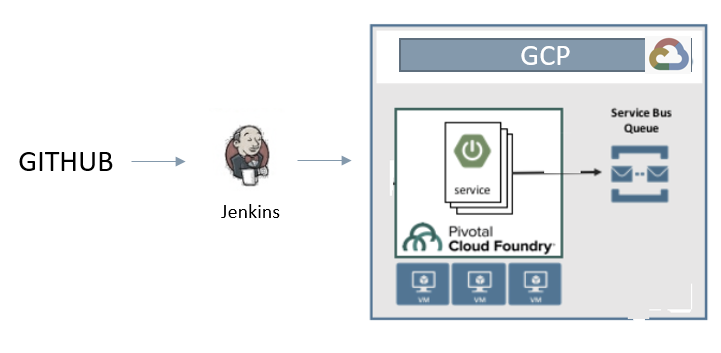
Associate will implement skills from Java to develop the application and Anthos KF to deploy the application.

Below are the skill details.

|  |  |
| --- | --- |
| **Tower Name** | **Topics** |
| Frontend | Angular, Jasmine |
| Backend – Java | Core Java 8  Spring Boot  Microservices  Junit, Mockito  JSON  Data Structures  Redis |
| Cloud -Paas – Anthos KF, Iaas-GCP | Anthos KF  Service Broker, Jenkins |

1. **Architecture Diagram for the Problem Statement**

Sample Architecture Diagram given below.



1. **User Stories**

Capture the user stories of the application.

|  |  |
| --- | --- |
| User Story # | User Story |
| US\_01 | As a user, I should be to login the Loan management application using User id and Password. I should be able to logout of the system safely.  Acceptance criteria:  I should be able to login with a User Id and Password that exists in database. On clicking logout the session should be invalidated and login page must be displayed |
| US\_02 | As a user, I should be able to search for loan information existing in the system. I should not have the ability modify or cancel the loan.  Acceptance criteria:  The search screen must have User’s First name, Last name, Loan Number. On entering any of the fields, I should get matching search results. If there are multiple search results indicate to the user to narrow down the search. Display only Loan detail screen only if matching results are found. No wild card search is required. |
| US\_03 | As an admin user I should be able to originate/add new loan  Acceptance criteria:  New Loan add screen must have User’s First Name, Last Name, Loan Number, and Property Address. All fields are required. If validation is successful, then add the record if not show validation message. |
| US\_04 | As an admin user I should be able to modify a loan (loan details – loan amount, loan type, loan term)  Acceptance criteria:  After a successful search from the Loan Search screen, display loan detail screen. Loan detail screen must have Update button to update any modification to the loan details. Fields will be displayed in text box. |
| US\_05 | As a developer I should be able to build and deploy the application using Jenkins pipeline to Anthos KF  Acceptance criteria:  Create Jenkins deployment job to push the application to Anthos KF. Application should be successfully built, deployed and running via the Jenkins pipeline. |

1. **Expected Deliverables**

Capture the deliverables to be submitted by associate after completing the development.

* Readme document on the complete application
* Setup of the application
* High level steps used to convert to server less architecture
* How to run the application
* Any inference
* Snapshot of any implementation
* Reports
* Code Assessment Report
* Functional Test Report
* Vulnerability Assessment Report
* Performance Test Report
* Code Profiling Report
* Automation Scripts
* Script file for Continuous Build, Continuous Integration & Continuous Deployment

1. **Milestone and duration**

As per project requirement, modification can be done in the below table.

|  |  |
| --- | --- |
| Milestone | Topic |
| Milestone -1 | Develop the UI using Angular. |
| Milestone -2 | Design and develop microservice application using Spring Boot and Core Java 8 features. |
| Milestone -3 | Deploy your application in Anthos KF using Jenkins Pipeline and perform the different types of testing |

1. **Implementation Notes**

As per the project requirement modification can be done in the below table.

|  |  |
| --- | --- |
| Frontend - Angular | Milestone – 1   Implementation of UI functionalities using Angular  Unit Test the UI using Jasmine |
| Backend -Java | Milestone-2   * Use Spring Boot- Rest APIs to develop the services * Use Microservice Architecture * Use Core Java 8 features * Use Domain Driven Design * Use Spring Data to work with database, under repository pattern * Use Oracle DB * Use browser / POST Man to invoke APIs * User access security microservice to allow/disallow CRUD operations * Any error message or exception should be logged (and help in refactor) * Unit test the application * All implementation should publish Code Quality Metrics using SonarCloud/SonarQube * Technical Debt – lower-the-better * Code Smell – lower-the-better * Cyclomatic Complexity - lower-the-better * Code Coverage – higher-the-better * Secure coding practices * Follow coding standards |
| Cloud - Anthos KF | Milestone-3   * Develop pipeline using Jenkins      * Deploy the application in Anthos KF |

1. **Evaluation rubrics**

As per the project requirement any addition can be done in the below table.

|  |  |
| --- | --- |
| Microservices | * Follow the below basic structure * API - Controllers * Domain - Model, Events, Business Services Integration * Services – API Implementation * Infrastructure Project      * Associate must have designed/developed Microservices as per the requirement * Each of the Microservices need to comprise below functionality, which need to be developed * Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc…) between Entity Classes. (Entity and Model classes have been developed in the Previous Phase) * In case specific Entity or Model classes are required across multiple Microservices, it is recommended to maintain separate copy of Entity or Model classes for each Microservices. * Microservices should interact with corresponding databases it owns. * Microservice need to interact with other Microservice * Usage of Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice * Circuit Breaker, Service Registry, Service Discovery should be implemented |
| Rest API | * Associate must have used REST API for exposing resources * Associate must have used HTTP GET/PUT/POST request method designators for the business methods which is to be exposed * Associate must have customized the request and response formats according to the requirement * Associates must have used appropriate RETURN CODES based on the service outcome * Associates must have extracted query/form/header parameters from the input * Associate must have built a custom response based on the input |
| Core Java 8 | * Associate should have used appropriate Base class Libraries, Control Statements and Operators, File Handling and Java 8 features for implementing the functionalities. |
| Unit Testing | * Test cases covers the functionality of API with custom inputs * Good test Coverage |
| Common | * Code Smell * Technical Debt * Secured Coding * Coding Standards |
| DevOps on Cloud | * DevOps pipeline for each microservices which uses cloud PaaS services to trigger a CI/CD pipeline when code is checked-in to GIT * The check-in process should trigger unit tests with mocked dependencies * Unit tests should not alter persistent data * DevOps dashboard should show status of CI/CD pipeline * DevOps pipeline should support manual approval for rollout, gradual traffic shifting and rollback to earlier version * Checked-in code should meet 75%+ code coverage in unit testing |
| Angular | * Usage of Angular Components-Module and Component, Databinding, data validation, * Routing, service, CLI commands usage |