Payment Processing – Wire transfer

Contents

[1.0 Problem statement 2](#_Toc47362483)

[2.0 Skill Tower develop the project 2](#_Toc47362484)

[3.0 Use Case/Architecture Diagram 2](#_Toc47362485)

[4.0 Use case description 3](#_Toc47362486)

[5.0 Expected Deliverables 5](#_Toc47362487)

[6.0 Milestone and duration 5](#_Toc47362488)

[7.0 Implementation Notes 5](#_Toc47362489)

[8.0 Evaluation rubrics 6](#_Toc47362490)

# Problem statement

Need to implement Wire transfers for electronic interbank payments for a new ABC-Bank, we can transfer funds directly from one entity’s bank account to another’s.

* Wire instructions include the bank account number and ABA bank routing number for the payee receiving the money.
* Each financial institution sets its cutoff time policy for sending same business day bank wires.
* Once funds have been wired, because of the immediacy, recalling erroneous transactions is much more difficult

# Skill Tower develop the project

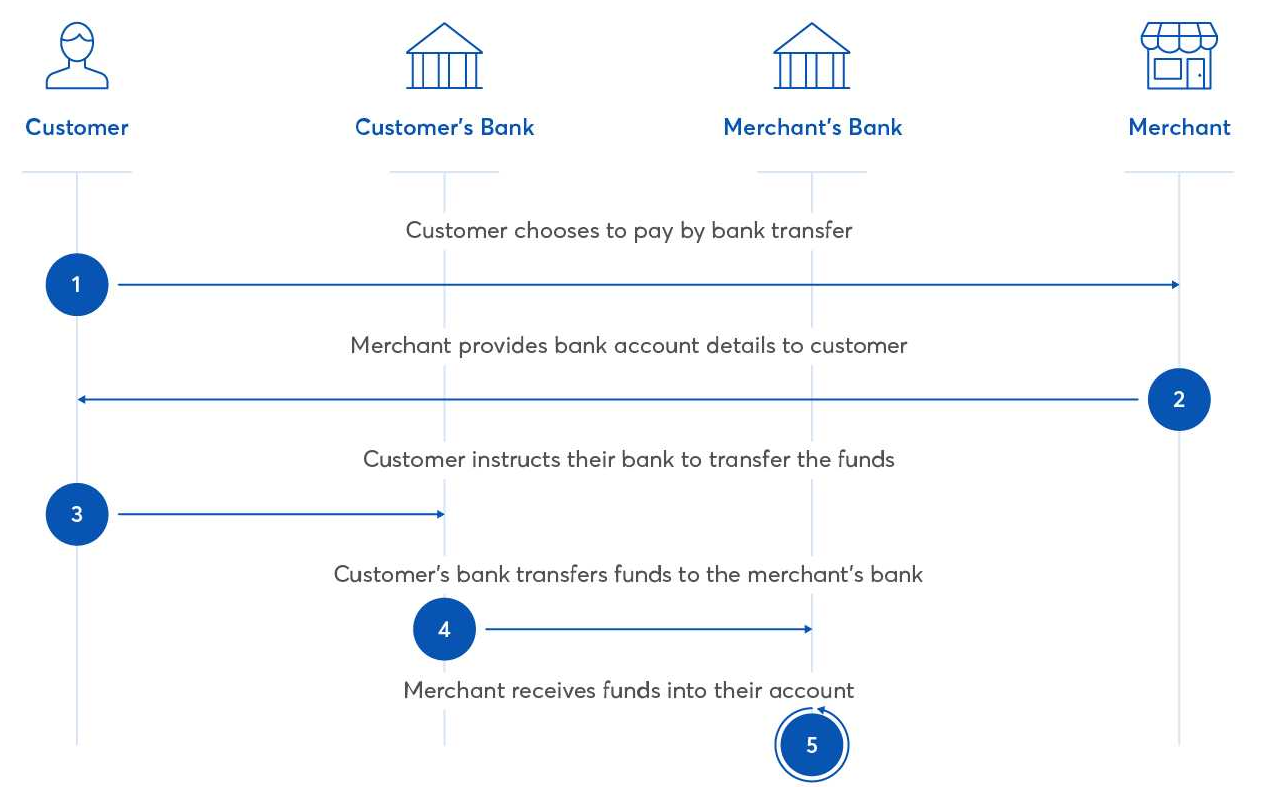
List the Technology based on your respective technology stack, that will be used to development the project.

Associate will choose any one of the technology stack and develop the application.

|  |  |
| --- | --- |
| Tower Name | Skill Names |
| Backend - Java | Core Java 8, Junit  Spring Boot  Spring Cloud, JSON  Microservice  Data Structures |

# Use Case/Architecture Diagram

**Use Case Diagram**



# Use case description

Develop a real time processing service, which has four micro services -

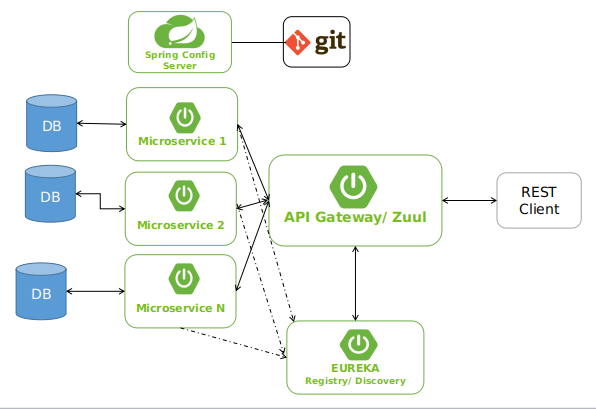
Either the interaction can queue based or REST based

* Initiation Service - Depending on Payment Instruments, the Payer, the Payee or other party (e.g., bank) initiates processing.
* Verification of Available Funds. Payee may need proof of funds or proof of hold before finalizing payment - Have a sufficiency service - This can be mocked using mockito
* Authorization of Transfer. Payee receives proof that the transfer of funds has been authorized by the Authorization Service of the Bank(like fraud check) – Mock this one also
* Completion of Transfer. When a customer submits a bank wire transfer request the money must be sent by the bank on the same day and received (as available funds) in the payee’s bank account.

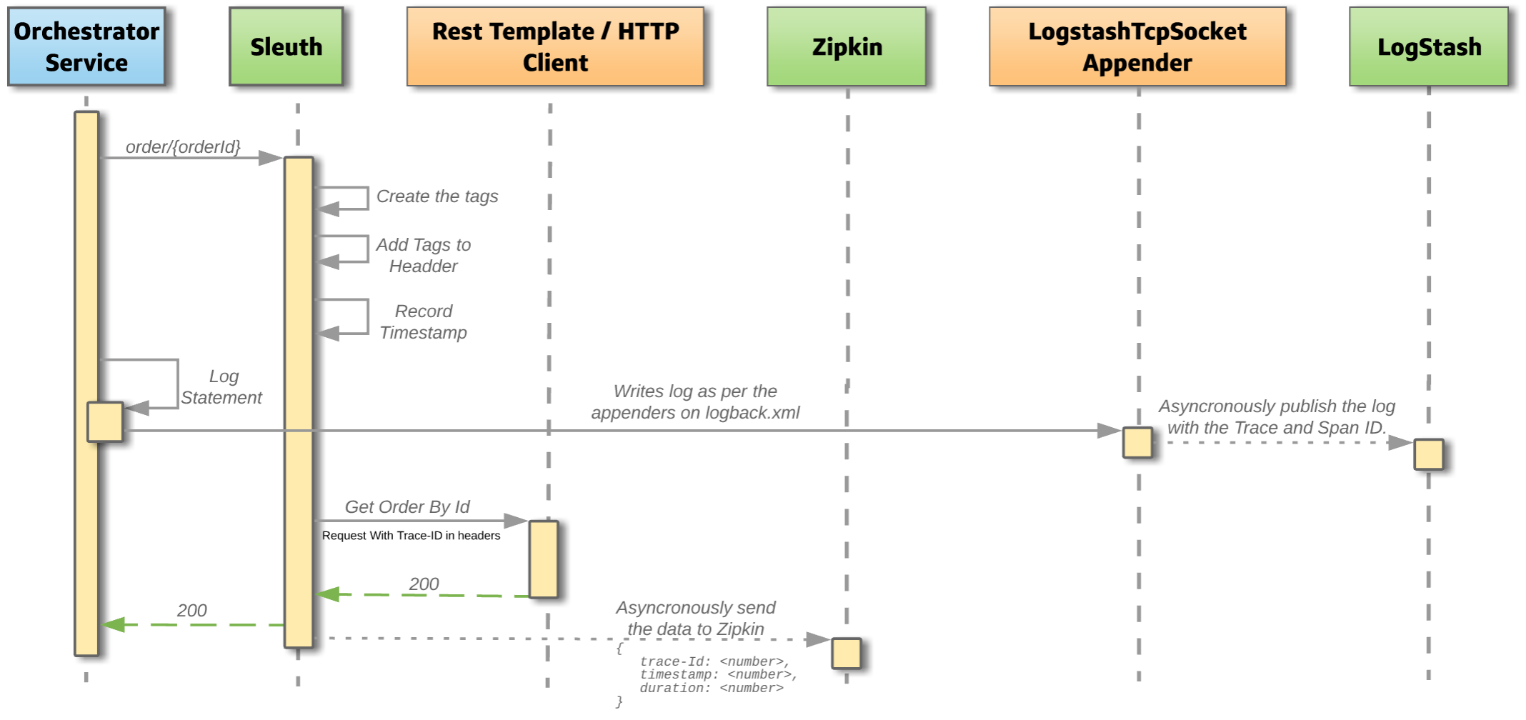
These micro services should interact sequentially to process the four steps

**Initiation -> Verification -> Authorization -> Completion**

The microservices should be developed to adhere the below reference architecture



Use Spring Cloud Sleuth and Zipking for tracing the microservices call end to end.



# Expected Deliverables

The following deliverables are expected as outcomes

* Application Code base
* Readme document on the complete application
* Testing Report/Screenshots
* Micro services cross cutting concerns used

# Milestone and duration

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

|  |  |
| --- | --- |
| Milestone | Topic |
| Milestone -1 | Developing the API/Microservice, Coding Standings, 12 Factor Principles  Developing RESTful services for initiation, verification, authorization of funds and final confirmation |
| Milestone -2 | Service Discovery, Registry, Circuit Breaker, API gateway, Testing the API, Code Quality Metrics |

# Implementation Notes

|  |  |
| --- | --- |
| Milestone -1 | * Use Rest APIs to develop the services * Use Microservice Architecture * Use Domain Driven Design * Implement repository pattern * Use Swagger definitions * Use browser / POST Man to invoke APIs * Use ORM to work with database, under repository pattern * Use browser / POST Man to invoke APIs * Implement API Versioning * Implement API Gateway * User access security microservice to allow/disallow CRUD operations * Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant * Any error message or exception should be logged to the user should be user-readable (and not technical) * Database connections and web service URLs should be configurable. * Web service URLs should be configurable. * Implement External Configurable Solution * Implement 12 Factor principles |
| Milestone -2 | * Implement Unit Test Project for testing the API * Implement Service Discovery, Registry, Circuit Breaker   All implementation should publish Code Quality Metrics   * Technical Debt –lower-the-better * Code Smell –lower-the-better * Code Coverage –higher-the-better * Secure coding practices * Follow coding standards |

# Evaluation rubrics

|  |  |
| --- | --- |
| Microservices | * Follow the below basic structure   + API - Controllers   + Domain - Model, Events, Business Services Integration   + Services – API Implementation   + Infrastructure Project * Associate must have designed/developed individual Microservices for each functionality. * Each of the Microservices need to comprise below functionality, which need to be developed   a. REST Controllers  b. Services  c. 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)  d. 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.  e. Microservices should interact with corresponding DB tables or Databases it owns.  f. Microservice need to interact with other Microservice  h. Usage of Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice  i. Unit Testing code should be developed using NUnit/Xunit and perform Unit Testing  j. Circuit Breaker, Service Registry, Service Discovery should be implemented   * Use Swagger UI and test each public method in the service * Implementation of Repository pattern |
| 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 |
| Java/C#/Node/Python | * Associate should have used appropriate Base class Libraries, Control Statements and Operators, File Handling and I/O Operations 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 |