|  |  |
| --- | --- |
|  | **Cognizant Academy**  **Corporate Classifieds**  **FSE – Business Aligned Project**  **Case Study Specification**  **Version 1.0** |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** | Srilakshmi Jayaraman |  |  | | **Role** | Solution Designer |  |  | | **Signature** |  |  |  | | **Date** |  |  |  | |
|  |

Table of Contents

[1.0 Important Instructions 3](#_Toc47003316)

[2.0 Introduction 4](#_Toc47003317)

[2.1 Purpose of this document 4](#_Toc47003318)

[2.2 Project Overview 4](#_Toc47003319)

[2.3 Scope 4](#_Toc47003320)

[2.4 Hardware and Software Requirement 5](#_Toc47003321)

[2.5 System Architecture Diagram 6](#_Toc47003322)

[3.0 Functional Requirements and High evel Design 7](#_Toc47003323)

[3.1 Use Case Diagram 7](#_Toc47003324)

[3.2 Individual Components of the System 8](#_Toc47003325)

[**3.2.1** **Offers Microservice** 8](#_Toc47003326)

[**3.2.2** **Employee Microservice** 9](#_Toc47003327)

[**3.2.3** **Points Microservice** 10](#_Toc47003328)

[**3.2.4** **Authorization Microservice** 10](#_Toc47003329)

[3.2.5 **Swagger** 11](#_Toc47003330)

[**3.2.6** **Corporate Classifieds Portal (MVC)** 11](#_Toc47003331)

[4.0 Cloud Deployment requirements 11](#_Toc47003332)

[5.0 Design Considerations 11](#_Toc47003333)

[6.0 Reference learning 12](#_Toc47003334)

[7.0 Change Log 13](#_Toc47003335)

# 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)

1. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable
2. 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.

# Introduction

## Purpose of this document

The purpose of the software requirement document is to systematically capture requirements for the project and the system “Corporate Classifieds” 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.

## Project Overview

An organization has to create an Classifieds App for its employees to allow posting of ads on a particular product that the employees can buy and sell within the employee community. The employee who posts an offer in Classifieds Application is a seller and an employee who shows interest is a consumer. Though the real selling doesn’t happen over the app, an engagement between the seller and the consumer is created, by liking/commenting to the offer.

## Scope

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

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Req. Name** | **Req. Description** |
| REQ\_01 | Offer Module | Offer Module is a Microservice that performs following operations:   * Post a new Offer * Edit an existing Offer * View Offer Status by Offer ID (Offer Status, Likes, Offer Opened Date, Engaged Date, Closed Date) * View Offers by Category * View Offers by Posted Date * View Offers by Top 3 likes in a Category |
| REQ\_02 | Employee Module | Employee Module is a Microservice that performs the following operations:   * View Offers Posted By Employee * View Recent Liked Offers * View Employee Profile |
| REQ\_03 | Points Module | Points Module is a Microservice that performs the following operations:   * Refresh and Get Points for a Seller (If an offer is engaged and closed within 2 days after posting, the seller gets some points, or if there are more likes points are added) |
| REQ\_04 | Classifieds UI Portal | The MVC Portal will interact with other Microservices and allows an employee to do the following:  Post an Offer  Edit a posted offer  Engage an offer (as a buyer)  View offers posted by self  View Recently Liked Offers  View offers based on category, Posted Date, Top Likes  View Profile (Employee Details and Points Gained) |

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.

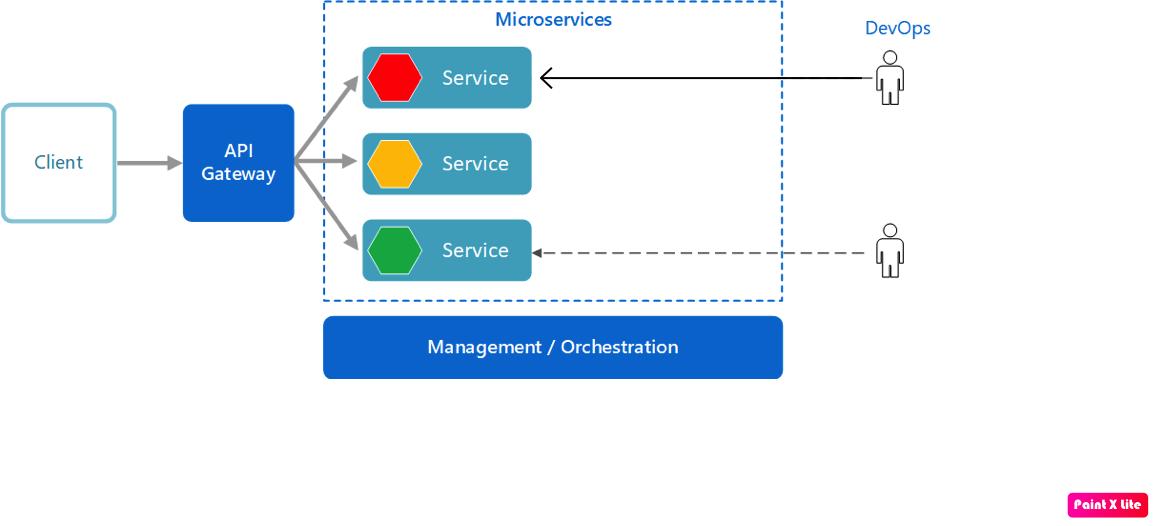
The requirement details given below states in-memory database usage. **The first phase of the development which is done in the first week, SHOULD use the Database for related activities and NOT the in-memory database.**

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.

## Hardware and Software Requirement

1. Hardware Requirement:
   1. Developer Desktop PC with 8GB RAM
2. Software Requirement (Java)
   1. Spring Tool Suite (STS) Or any Latest Eclipse
      1. Have PMD Plugin, EclEmma Code Coverage Plugin and AWS Code Commit Enabled
      2. Configure Maven in Eclipse
   2. Maven
   3. Docker (Optional)
   4. Postman Client in Chrome
   5. AWS Cloud Account
3. Software Requirement (Dotnet)
   1. Visual studio 2017 enterprise edition
   2. SQL Server 2014
   3. Postman Client in Chrome
   4. Azure cloud access

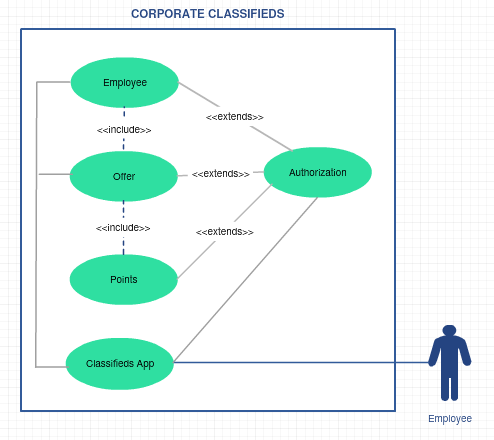
## System Architecture Diagram



Auth Service

# Functional Requirements and High evel Design

## Use Case Diagram



## Individual Components of the System

### **Offers Microservice**

|  |  |
| --- | --- |
| Corporate Classifieds | Offers Microservice |
| **Functional Requirements**  Offers Microservice holds the repository of offers posted by the corporate employees. An employee posts a new offer, edits the post and views the posts done by their peers.    Post Authorization, Offers Microservice will provide the following functionalities:   * Post / Edit an offer, or Engage an offer (means buyer likes and want to go with it) * View Offer status by offer ID (Likes, Opened Date, Closed Date etc.) * View Offers by category, top 3 likes in a category, by posted date   Anything related to offer will be managed by this Microservice only. However when an offer is edited or newly posted, meta data of an offer related to employee is pushed to Employee Service so that the Employee service will take care of static read only data for every request specific to an employee | |
| **Entities**   1. **Offer**   <Details of Offer, Offer categories>   1. **Offer\_Category**   <Details of Offer Categories (Apartment Rental, Resale of Electronics etc. >  **REST End Points**  **Offer Microservice**   * + GET: /getOfferDetails (Input: Offer ID | Output: Offer Details, Likes, Opened Date etc.)   + GET: /getOfferByCategory (Input: Offer ID | Output: Offer Details, Likes, Opened Date etc.)   + GET: /getOfferByTopLikes (Input: Offer ID | Output: Offer Details, Likes, Opened Date etc.)   + GET: /getOfferByPostedDate (Input: Offer ID | Output: Offer Details, Likes, Opened Date etc.)   + POST: /engageOffer (Input: offer ID, Employee ID| Output:Status of Update)   + POST: /editOffer (Input: offer ID, Employee ID| Output:Status of Update)   + POST: /addOffer (Input: Offer details, Employee Details) | Output: Status of Post) | |
| **Trigger** – This Microservice will get invoked by Employee Microservice, Points Microservice and by Classifieds Portal | |
| **Steps and Actions**   1. Offer details have to be retrieved from entties based in input conditions for all GET requests 2. When add / edit offer request is received, perform basic validations before actual adding / editing. 3. Post validations, the Offer entites are updated. The employee specific Offer Meta data is also pushed to Employee Service. The data can include the details of offer posted by respective employee. | |
| **Non-Functional Requirement:**   * Only Authorized Member can access these REST End Points | |

### **Employee Microservice**

|  |  |
| --- | --- |
| Corporate Classifieds | Employee Microservice |
| **Functional Requirements**  Employee Microservice allows to view the offers posted by an employee and view the most liked offers by an Employee. This allows the portal to display the details whenever they have logged in.  To view the following:   * Offers posted by an employee * Offers most liked by an employee * Own Profile | |
| **Entities**   1. **Employee**   <details of an employee, including points gained>   1. **Employee\_Offers**   <Details of Offers posted by every employee>  **REST End Points**   1. **Employee Microservice**    * GET: /viewEmployeeOffers (Input: employeeID | Output: Employee Offers details including Points gained so far)    * GET: /viewMostLikedOFfers (Input: employeeID | Output: Top 3 Offers most liked by the employee)    * GET: /viewProfile (Input: employeeID | Output: employee details with Points gained) | |
| **Trigger** – When Profile page is opened up in the Portal, the Employee service gets invoked. This will interact with Offer and Points service to get the necessary details | |
| **Steps and Actions**   1. Employee Microservice interacts with Offer Service. To retrieve the offers posted by an employee and the count of likes for all offers posted. | |

### **Points Microservice**

|  |  |
| --- | --- |
| Corporate Classifieds | Points Microservice |
| **Functional Requirements**  Points Microservice interacts Offer Service. Post authorization of request, Points Microservice allows the following operations:   * Refresh the points gained by the employee, for the posts.   + If the employee posts has more than 50 likes within 2 days, add 10 points   + If the employee posts has more than 100 likes within 2 days, add 50 points   + If the employee posts has been engaged withn 2 days, add 100 points * View the points gained by an employee | |
| **Entities**   1. **Employee\_Points**   <Points details of an employee>  **REST End Points**  **Policy Microservice**   * + GET: /getPointsOfEmployee (Input: Employee ID | Output: number of points accumulated   + POST: /refreshPointsOfEmployee (Input: Employee ID | Output: Status) | |
| **Trigger** – Points microservice will be invoked by Classifieds portal and Employee Service and interacts with Offer service to understand the offer details of an employee | |
| **Steps and Actions**   1. When Refresh Points end point is invoked it communicates with Offer service to understand the number of likes, engaged date, posted date etc., and calculate points. 2. Classifieds and Employee Service will invoke the Points service basically to understand the points gained | |

### **Authorization Microservice**

|  |  |
| --- | --- |
| Corporate Classifieds | 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 | |

### **Swagger**

|  |  |
| --- | --- |
| Corporate Classifieds | Swagger |
| **Documentation Requirements**   * All the Microservices must be configured with Swagger for documentation * 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 | |

### **Corporate Classifieds Portal (MVC)**

|  |  |
| --- | --- |
| Corporate Classifieds | Classifieds Portal |
| **Client Portal Requirements**   * Portal must allow an Employee to Login. Once successfully logged in, the member do the following operations: * Post an Offer – reach out to Offer service * Edit a posted offer– reach out to Offer service * Engage an offer (as a buyer) – reach out to Offer service * View offers posted by self -– reach out to Emplpoyee service * View Recently Liked Offers – reach out to Employee service * View offers based on category, Posted Date, Top Likes – reach out to Offer service * View Profile (Employee Details and Points Gained) – reach out to Employee service * Each of the above operations will reach out to the middleware Microservices that are hosted in cloud. | |

# 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.

# Design Considerations

These design specifications, technology features have to be strictly adhered to.



# Reference learning

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

|  |
| --- |
| <https://cognizant.kpoint.com/app/video/gcc-6e36500f-c1af-42c1-a6c7-ed8aac53ab22> |
| [https://cognizant.kpoint.com/app/video/gcc-92f246c9-024a-40b7-8bfc-96b3ce7c1a39](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fcognizant.kpoint.com%2Fapp%2Fvideo%2Fgcc-92f246c9-024a-40b7-8bfc-96b3ce7c1a39&data=02%7C01%7Ckumar.mahadevan%40cognizant.com%7C1278a7e184c6454d69c108d7fbe06c69%7Cde08c40719b9427d9fe8edf254300ca7%7C0%7C0%7C637254813626816518&sdata=9A4V%2F2ippq99uff4iyxYxHAr1qyLptaQgjcAJjvw5Kw%3D&reserved=0) |
| <https://cognizant.kpoint.com/app/video/gcc-cfedd9c1-e29e-4e3e-b3e2-1960277f72a3> |
| <https://cognizant.kpoint.com/app/video/gcc-900a7172-43b7-42f3-a6cc-e301bd9cc9b3> |

Microservices deployment into Azure Kubernetes Service.

|  |
| --- |
| [AzureWithCICD-1](https://cognizant.kpoint.com/app/video/gcc-19532393-d4e0-4fd9-8a0c-80ecbdb349d3) |
| [AzureWithCICD-2](https://cognizant.kpoint.com/app/video/gcc-6633a958-ab72-4c69-b926-fe832e4b56a1) |
| [AzureWithCICD-3](https://cognizant.kpoint.com/app/video/gcc-553eb186-c1cf-448e-96fc-a96fe37b2e6a) |
| [AzureWithCICD-4](https://cognizant.kpoint.com/app/video/gcc-fad7d4af-d651-4501-99c6-2785190670c2) |

**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](https://dzone.com/articles/microservices-communication-feign-as-rest-client) |
| Swagger (Optional) | [https://dzone.com/articles/centralized-documentation-in-Microservice-spring-b](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 Security | <https://dzone.com/articles/spring-boot-security-json-web-tokenjwt-hello-world> |
| 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> |

# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Changes Made | | | |
| V1.0.0 | Initial baseline created on <24-Jul-2020> by <Srilakshmi Jayaraman> | | | |
|  |  | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
|  |  |  |  |