|  |
| --- |
|  |

|  |
| --- |
| **Mentor On Demand (Mid Tier) Phase5 v2.0** |
| Case Study |
|  |
| This document covers Software Requirements of Mentor On Demand, along with list of Technologies to be used to develop this Software System, and also includes some details on the Architecture |
|  |
| **IIHT** |
| **2/7/2019** |
|  |

Table of Contents

[1. Business Requirement(Mentor On Demand) 2](#_Toc12976810)

[1.1. Retail Use cases 2](#_Toc12976811)

[1.1.1. User(Retail) Use Cases. 2](#_Toc12976812)

[1.1.2. Mentor Use Cases 3](#_Toc12976813)

[1.1.3. Admin Use Cases 3](#_Toc12976814)

[2. Microservices Integration and Security 3](#_Toc12976815)

[3. Spring Microservices Tools to be used 4](#_Toc12976816)

[4. JWT Authentication 5](#_Toc12976817)

[5. Architecture/Design 5](#_Toc12976818)

[6. Technical Spec – Solution Development Environment 5](#_Toc12976819)

[6.1. Front End Layer 5](#_Toc12976820)

[6.2. Middle Tier Layer 5](#_Toc12976821)

[6.3. Database & Integration Layer 6](#_Toc12976822)

[6.4. Ancillary Layer 6](#_Toc12976823)

[Controllers can be tested using Postman Tool 6](#_Toc12976824)

[6.5. Security 6](#_Toc12976825)

[6.6. Deployment & Infrastructure 6](#_Toc12976826)

[6.7. Editors 6](#_Toc12976827)

[7. Other Design constraints 6](#_Toc12976828)

[8. Assessment Deliverables 7](#_Toc12976829)

[9. Important Instructions 7](#_Toc12976830)

# Business Requirement(Mentor On Demand)

Build a software system which lets users search for a Mentor for a specific Technical Course, User can search, select, propose & finalize a Mentor of his Choice, post which Training gets started. Along with B2C, B2B functionality also need to be supported, for which Users can be Corporate Organizations seeking Technical Courses, which are offered by various Vendor Organizations(which may have pool of Trainers). Payment processing, E-mail interaction is within scope of the Project.

Project can be divided into two Phases 1 and 2. Phase 1 comprises Retail Use Cases(B2C) and Corporate related Use Cases(B2B).

## Retail Use cases

Below are the different roles, which need to be supported by above Software System.

#1. User(Retail)

#2. Mentor(Retail)

#3. Admin

Below are the Use Cases which need to be supported by each of above Roles

### User(Retail) Use Cases.

User should be able to Signup. Email ID can be used as username, which need to be confirmed by User to complete Signup process.

User should be able to Signin.

User can search for Mentors for a specific Time and Technology(mandatory), even without signing in.

Search results need to list Mentors having expertise in specific Technology, who are available in that specific time(Mentor's Calendar can be checked to display search results)

A search result has to display Name of Mentor, years of experience, no. of trainings delivered in total and in specific technology, Fee charged (including commission).

After viewing Mentor's Profile/History/Rating, User can select a specific Trainer from above Search Results and should be possible to send Proposal, to the Mentor.

Once the User Proposes, Propose request is received by a specific Mentor. A Mentor can Confirm or Reject a Proposal. Proposal Confirmation or Rejection need to be sent to User.

User can Finalize a Confirmed Proposal response, after which user need to make payment, for Training Course.

Payment is collected in advance from User. But Mentor will be paid in 4 slots, as the Training Progress reaches 25%, 50%, 75% and 100%

Actual progress need to be updated by User.

Should be possible to Rate any Mentor, of the Trainings which are in Progress

User should be able to view list of Current Trainings in Progress.

Trainings in any of below states are considered to be in Progress.

1. Proposed.
2. Confirmed Proposals
3. Trainings started, and not completed

User should be able to view List of Trainings already availed and Completed.

### Mentor Use Cases

Mentor need to SignUp to provide Mentoring service through the Portal

During SignUp process Mentor need to provide timezone/working timings, list of technologies, facilities provided - material, examples/cloud labs, email/mobnum verification, linkedin URL, mentor profile, number of years of experience. Email id can be used as Mentor’s username.

View History of Mentor - Trainings delivered, ratings, etc...

Payment will be fixed based on Technology/Complexity, etc... and based on Trainers experience, proposal

Trainer will be paid in 4 slots, based on progress

Payment will get accumulated based on Progress, which can be withdrawn by Trainer.

E-Mail Notifications need to be automatically sent to User and Mentor, during appropriate Use Cases.

### Admin Use Cases

Admin can add/remove List of Technologies

Block or Unblock User or Mentor

Admin should be able to search the Payments made, and display reports

Admin should be able to edit parameters such as Payment Commission percentage(which will be deducted from the payments done by User).

Contact details should be confidential, and should not be shared with each other.

# Microservices Integration and Security

Assuming that you are done with developing individual Microservices in previous Phase, current Phase includes creating and integrating Zuul gateway, Eureka Server and Eureka client in each Microservice. This is shown in architecture Diagram, in next section.

Zuul Gateway(create a Zuul based Project using Spring Initilaizer or STS IDE), add required annotation. Authentication and JWT Token validation can be performed in Zuul’s Pre Filter.

Add below details to yml or property file

1. add route configurations
2. port number & url of eureka Server

Eureka Server(create a Eureka Discovery Server using Spring Initializer or STS IDE), add required annotation & port number in yaml configuration file

Add Eureka Discovery Client to all the Microservice

Now open Eureka Server Dashboard by opening and crosscheck if all Microservices are registered in the dashboard

Now start sending the requests to Zuul Gateway which further routes to a specific Microservice based on the url pattern

Develop code for Unit Testing

PostMan, to test REST end points

# Spring Microservices Tools to be used

As already specified under Full Stack Technologies Microservice Architecture need to be followed. Ensure that the Application is divided into multiple Microservices, along with database/tables each Microservice Manages. Below Spring Microservices Tools need to be used

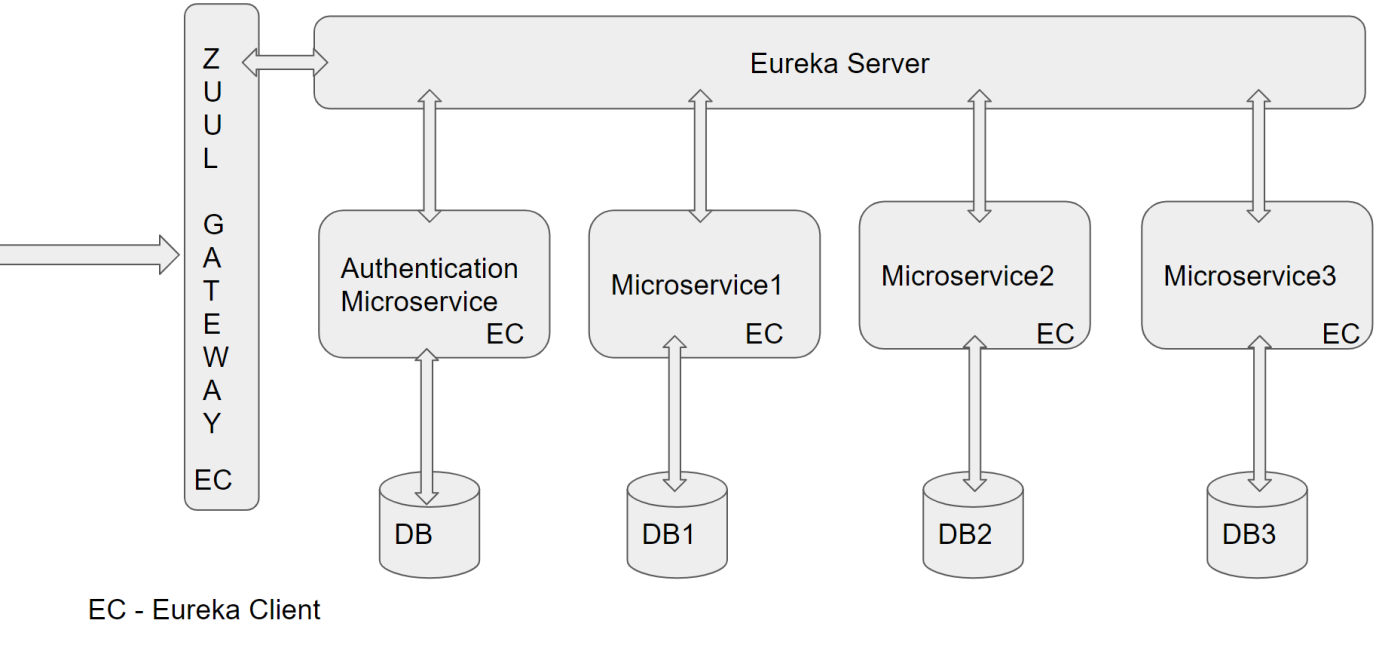
* Zuul API Gateway
* Eureka Service Registry & Discovery
* Ribbon Client side Load Balancer(optional)
* Feign Client
* Hystrix Circuit Breaker & Fault Tolerant Tool(optional)

# JWT Authentication

Create additional Microservice which takes care of authentication and role activities, and JWT Token validation. Spring Security need to be used for Authentication. On successful authentication or token validation the actual request need to be forwarded to the corresponding Microservice. Invoke authentication REST endpoints from Zuul Gateway. Use PreFilter to perform JWT Token validation by invoking REST endpoint of this Microservice.

Instead of JWT, any other security protocol such as OAuth2 can be used. Authentication data can be stored in MySQL DB or LDAP or any other data source.

# Architecture/Design



# Technical Spec – Solution Development Environment

## Front End Layer

|  |  |
| --- | --- |
| **Framework(s)/SDK/Libraries** | **Version** |
| Angular with TypeScript | 4/6 |
| Bootstrap | 3.0 or above |
| CSS | 3 |
| HTML | 5 |
| JavaScript | 1.8 or above |
| JQuery | 1.3 |

## Middle Tier Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Java Stack | Spring Boot | 1.5 or above |
| Spring MVC | 4.0 or above |
| JDK | 1.7 or above |
| Maven | 3.x or above |

## Database & Integration Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Java Stack | Hibernate | 4.0 or above |
| JAX-RS Jersey/ Spring Restful |  |
| MySQL | 5.7.19 |
| MongoDB | MongoDB | 3.4 |
| NoSQL |  |

## Ancillary Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Source Code Management Tool | GIT | 2.14.2 |
| Build Tool/JAVA Stack | Maven | 3.x |
| Testing Tool/JAVA Stack | JUnit/Mockito | 4.x |
| Testing Tool/JAVA Stack | Spring Test | 4.x |

## Controllers can be tested using Postman Tool

## Security

|  |  |
| --- | --- |
| **Name** | **Version** |
| Spring Boot Security |  |
| JWT |  |

## Deployment & Infrastructure

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Docker | - |  |
| Apache Tomcat | - |  |
| Jenkins(CI/CD) | - |  |
| Node | - |  |

## Editors

|  |  |
| --- | --- |
| **Name** | **Version** |
| STS(Spring Tool Suite) |  |
| Visual Studio Code |  |

# 

# Other Design constraints

Below are other Design constraints to be considered

* Integrate with any SMTP Server, to send Emails in appropriate Use cases
* Integrate with any Payment Gateway to process Payments

# Assessment Deliverables

1. Updated Design document by adding Zuul, Eureka into the Architecture
2. Check in below project Zuul Gateway Project, Eureka Server Project, all Microservices with Eureka Client added
3. Screen shots showing sample invocation of Zuul Gateway end points from Post Man Tool.
4. Few Steps on how to run the solution.
5. Test code need to be included

# Important Instructions

1. Above sample Design provided is just for reference, Associates can make changes over it or use their own Design.
2. Please make sure that your code does not have any compilation errors while submitting your case study solution.
3. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
4. Implement the code using best design standards/family Design Patterns.
5. Use Internationalization for all the labels and messages in Rest API Development.
6. Do not use System out statements or console.log for logging in Rest API and FrontEnd respectively. Use appropriate logging methods for logging statements/variable/return values.
7. If you are using Spring Restful or Jersey JAX-RS to develop Rest API, then use Maven to build the project and create WAR file.
8. Write web service which takes input and return required details from database.
9. Use JSON format to transfer the results.