|  |
| --- |
|  |

|  |
| --- |
| **Mentor On Demand (Deployment Phase7) 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](#_Toc12977935)

[1.1. Retail Use cases 2](#_Toc12977936)

[1.1.1. User(Retail) Use Cases. 2](#_Toc12977937)

[1.1.2. Mentor Use Cases 3](#_Toc12977938)

[1.1.3. Admin Use Cases 3](#_Toc12977939)

[2. Full Stack Technologies 3](#_Toc12977940)

[3. Spring Microservices Tools to be used 4](#_Toc12977941)

[4. Technical Spec – Solution Development Environment 4](#_Toc12977942)

[4.1. Front End Layer 4](#_Toc12977943)

[4.2. Middle Tier Layer 4](#_Toc12977944)

[4.3. Database & Integration Layer 4](#_Toc12977945)

[4.4. Ancillary Layer 5](#_Toc12977946)

[4.5. Security 5](#_Toc12977947)

[4.6. Deployment & Infrastructure 5](#_Toc12977948)

[4.7. Editors 5](#_Toc12977949)

[5. Jenkins CI/CD 5](#_Toc12977950)

[6. Configure Jenkins and Docker for the Project 6](#_Toc12977951)

[6. Perform CI/CD 6](#_Toc12977952)

[7. Diagram 7](#_Toc12977953)

[8. Other Design constraints 7](#_Toc12977954)

[9. Assessment Deliverables 7](#_Toc12977955)

[10. Important Instructions 7](#_Toc12977956)

# 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.

# Full Stack Technologies

The technologies included in Full Stack are not limited to following but may consist of:

* UI Layer (HTML5, CSS3, Bootstrap 4, JavaScript, Jquery, Angular 4/6)
* Middleware Restful API (Spring Boot Restful & MicroServices, JAX-RS, Spring MVC)
* Database Persistence ( Hibernate)
* Database layer (MySQL, MongoDB)
* Ancillary skills (GIT, Jenkins(CI/CD), Docker, Maven) etc.

To complete this case study, you should be comfortable with basic single page web application concepts including REST and CRUD. You may use angular-cli to create your template project. All web pages need to be responsive.

Ref1: https://cli.angular.io/

Ref2: https://github.com/angular/angular-cli

# 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
* Feign Client
* Hystrix Circuit Breaker & Fault Tolerant Tool(optional)

# 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 |  |

Agile/Scrum Software development Model can be used

# Jenkins CI/CD

**Jenkins CI/CD:** As already known Jenkins is popular tool to perform CI/CD. When the code is pushed to GIT, build need to be automatically created and deployed. If possible create a Docker image and run the Container on Docker Host

This Phase also includes completion of Integration of Front end with Mid Tier.

**Deployment on Cloud(optional):** Any of the Microservices or Front End can be deployed on any Cloud(AWS, Azure, etc…) of your choice.

# Configure Jenkins and Docker for the Project

* Import the project (as discussed above) in Spring Tool Suite and configure it locally to run it as Spring Boot App.
* You may need to configure MySQL credentials and database name.
* Execute the project locally and access the app at http://localhost:portnumber
* Once, it is working fine in local development environment; Configure CI/CD in Jenkins, along with Dockerization
* Push the app source in internal GIT server. Pl. ask your mentor for the Internal GIT server URL.
* Configure Jenkins locally to pull the source from internal GIT repository
* Jenkins should build the project and create the deployable (war/jar). It should run the unit tests created in ”Maven, GIT, Junit, Tomcat Micro Layer for the Project”
* From Jenkins, invoke Docker commands to perform, below
* Creation of Docker Image(docker build . )
* Create and run Docker Container(docker run <image\_id>)

# 6. Perform CI/CD

1. Make few changes in the project (source code)

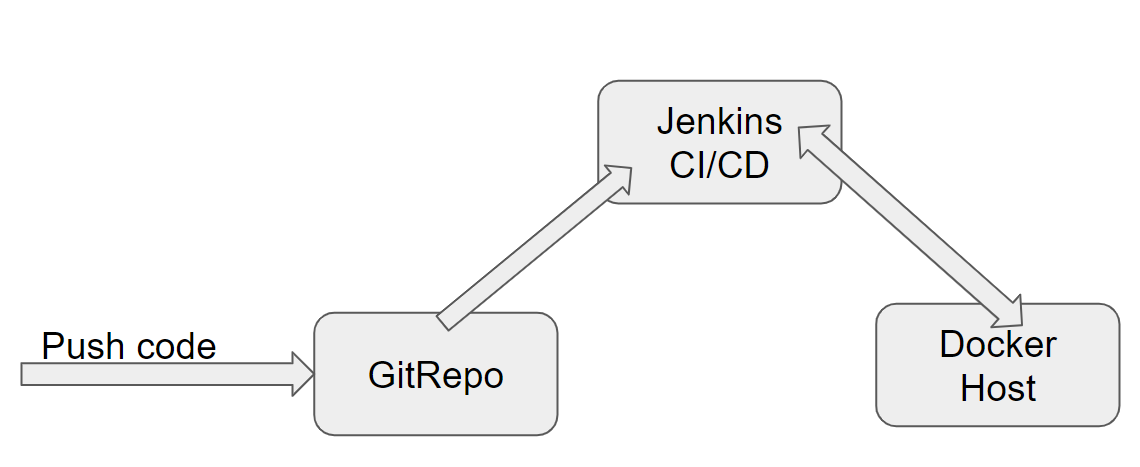
2. Make it sure that project is running locally in development environment without errors.

3. If it running locally without errors, push the changes to the internal GIT repository which was connected

4. If Project was Setup properly, Jenkins will automatically pull the code updates from internal GIT repo and build and deploy the project with updated code.

5. Now, when you visit http://localhost; you should see the changes in the browser window

# 7. Diagram



# 8. 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

# 9. Assessment Deliverables

1. Jenkinsfile, Jenkins screenshots
2. Dockerfile and Docker commands used and Screen shots of usage of Docker

# 10. Important Instructions

1. Adhere to the design specifications mentioned in the case study.
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.