|  |
| --- |
|  |

|  |
| --- |
| **Mentor On Demand (Mid-Tier Phase 4) 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** |
| **1/10/2019** |
|  |

Table of Contents

[1. Business Requirement(Mentor On Demand) 2](#_Toc12976564)

[1.1. Retail Use cases 2](#_Toc12976565)

[1.1.1. User(Retail) Use Cases. 2](#_Toc12976566)

[1.1.2. Mentor Use Cases 3](#_Toc12976567)

[1.1.3. Admin Use Cases 3](#_Toc12976568)

[2. Development of individual Microservices 4](#_Toc12976569)

[3. Architecture Diagram 4](#_Toc12976570)

[4. Full Stack Technologies 5](#_Toc12976571)

[5. Database Tables 6](#_Toc12976572)

[6. Technical Spec – Solution Development Environment 6](#_Toc12976573)

[6.1. Front End Layer 6](#_Toc12976574)

[6.2. Middle Tier Layer 6](#_Toc12976575)

[6.3. Database & Integration Layer 6](#_Toc12976576)

[6.4. Ancillary Layer 7](#_Toc12976577)

[Controllers can be tested using Postman Tool 7](#_Toc12976578)

[6.5. Security 7](#_Toc12976579)

[6.6. Deployment & Infrastructure 7](#_Toc12976580)

[6.7. Editors 7](#_Toc12976581)

[7. Other Design constraints 7](#_Toc12976582)

[8. Assessment Deliverables 7](#_Toc12976583)

[9. Important Instructions 8](#_Toc12976584)

# 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.

# Development of individual Microservices

This specific Phase is to design/develop individual Microservices. Analyze the requirement and divide back end functionality into multiple Microservices. Based on the Mentor on Demand requirements, below can be possible Microservices

1. User Microservice
2. Training Microservice
3. Technology Microservice
4. Search Microservice
5. Security Microservice
6. Payment Microservice

Each of above Microservice need to comprise below functionality, which need to be developed

1. REST Controllers
2. Services
3. 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)
4. 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 Microservice.
5. Microservice interaction with corresponding DB tables or Databases it owns.
6. It is possible that one Microservice need to interact with other Microservice(using RestTemplate or FeignClient)
7. Repository class which implements JPA or CrudRepository, if RDBMS is used
8. Usage of Custom Queries using @Query where ever custom functionality required
9. Feign Client can be used to invoke one Microservice, from another Microservice
10. Send Email Notifications to Users/Mentors in Use Cases wherever appropriate.
11. Use Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice
12. Unit Testing code can be developed using JUnit, Mockito, and perform Unit Testing

# Architecture Diagram

Multiple Microservices interacting with Database each one owns.



Architecture of a Single Microservice with REST Controller, Service, Model & Entity Classes and Repository classes



# 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>

# Database Tables

Below are list of Database Tables.

|  |  |
| --- | --- |
| Table Name | Purpose |
| Users | Stores list of User, Mentor, Admin – usernames and login credentials |
| Technologies | Store list of all Technologies, whose Trainings are supported by Portal |
| Trainings | Stores details of each Training in progress or completed |
| MentorSkills | Stores skills of Mentor |
| MentorCalendar | Stores Calendar slots of a Trainer |
| Payments | Stores payments released to Trainer, based on Training Progress |

# 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

# 

# 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. Checked in(to internal GIT) Source code of all Microservices
2. Screen shots of Usage of Post Man tool to test each End Point of all Microservices
3. Few Steps on how to run the solution.
4. Test code need to be included

# Important Instructions

1. Sample Design provided is just for reference, Associates can make changes over it or follow their own Design.
2. Based on your current work, alternate Technologies can be used, for example ReactJS instead of Angular, etc…, however prior approval from the Mentor is required.
3. Please make sure that your code does not have any compilation errors while submitting your case study solution.
4. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
5. Implement the code using best design standards/family Design Patterns.
6. Use Internationalization for all the labels and messages in Rest API Development.
7. 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.
8. 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.
9. Write web service which takes input and return required details from database.
10. Use JSON format to transfer the results.