IIHT

Time To Complete: 10 to 12 hr

Learning management system

Contents

[1 Problem Statement 2](#_Toc102742099)

[2 PROPOSED LMS WIREFRAME 3](#_Toc102742100)

[3 Application Architecture 4](#_Toc102742101)

[3.1 Microservice Architecture (Governance and Tooling/ Database and Storage): 4](#_Toc102742102)

[3.2 Microservice Architecture (Governance and Tooling/ Compute and Integration): 4](#_Toc102742103)

[3.3 Microservice Architecture (Compute and Integration/Presentation/Networking and Content Delivery): 5](#_Toc102742104)

[4 Cloud Architecture 6](#_Toc102742105)

[5 Tool Chain 7](#_Toc102742106)

[6 Development Flow 8](#_Toc102742107)

[7 Business Requirements: 8](#_Toc102742108)

[8 Proposed Rest Endpoints to be exposed 10](#_Toc102742109)

[8.1 Rest APIs: 10](#_Toc102742110)

[9 Rubrics/Expected Deliverables 11](#_Toc102742111)

[9.1 Engineering Concepts (Compute & Integration): 11](#_Toc102742112)

[9.2 Engineering Concepts (Security & Identity): 11](#_Toc102742113)

[9.3 Products & Framework (Database & Storage): 11](#_Toc102742114)

[9.4 Products & Framework (Governance & tooling): 11](#_Toc102742115)

[9.5 Debugging & Troubleshooting 12](#_Toc102742116)

[9.6 Code Quality/Optimizations 12](#_Toc102742117)

[10 Frontend 12](#_Toc102742118)

[11 Platform 12](#_Toc102742119)

[11.1 Compute 12](#_Toc102742120)

[11.2 Compute, Identity & Compliance, Security& Content Delivery 12](#_Toc102742121)

[11.3 FrontEnd Deployment 13](#_Toc102742122)

[12 Methodology 13](#_Toc102742123)

[12.1 Agile 13](#_Toc102742124)

# Problem Statement

**Learning Management System(LMS)** Application is a Restful Microservice application, where it allows users to search/add/delete the course from system for all the technology.

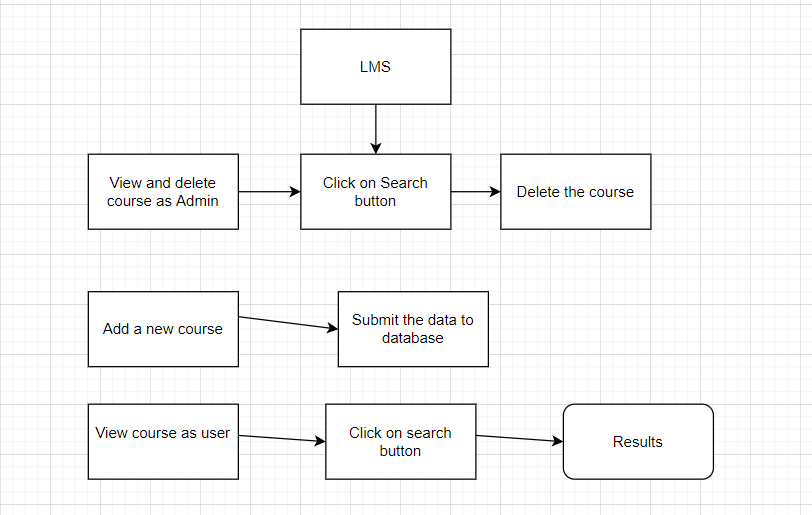
The core modules of LMS app are:

* Allows to add a new course to the system
* Allows to delete an existing course from the system
* Allows to search the courses available for a particular technology
* Allows to display all courses available

The scope includes developing the application using tool chain mentioned below.

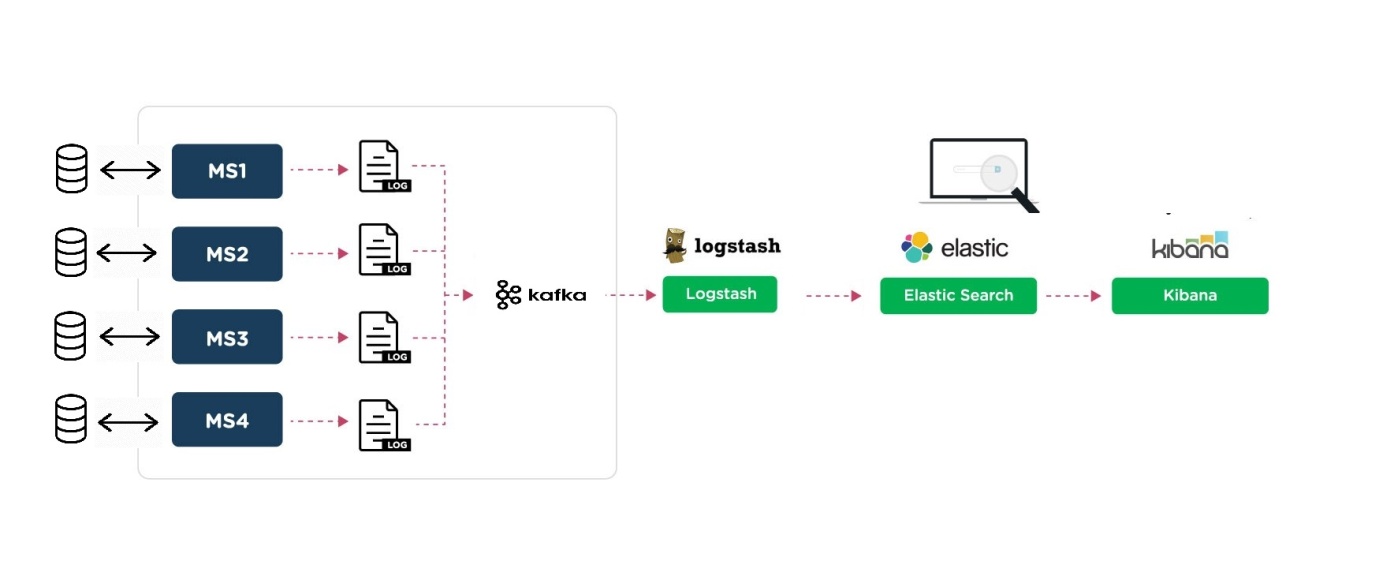
# PROPOSED LMS WIREFRAME

UI needs improvisation and modification as per given use case.

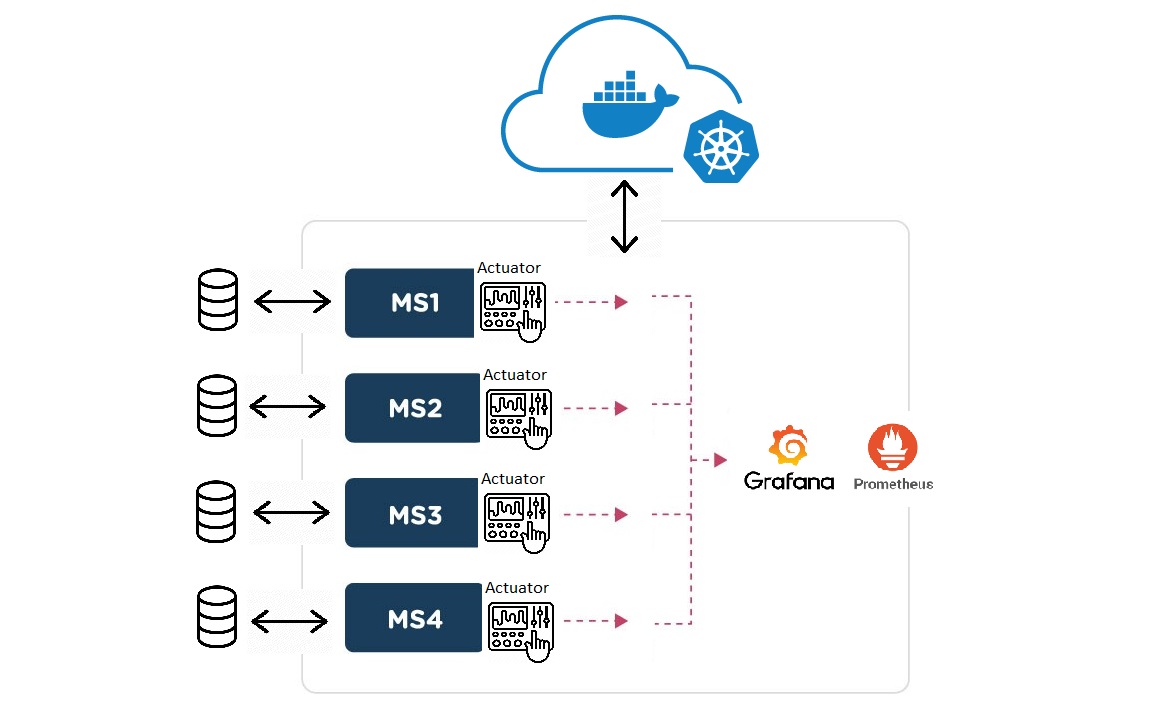


# Application Architecture

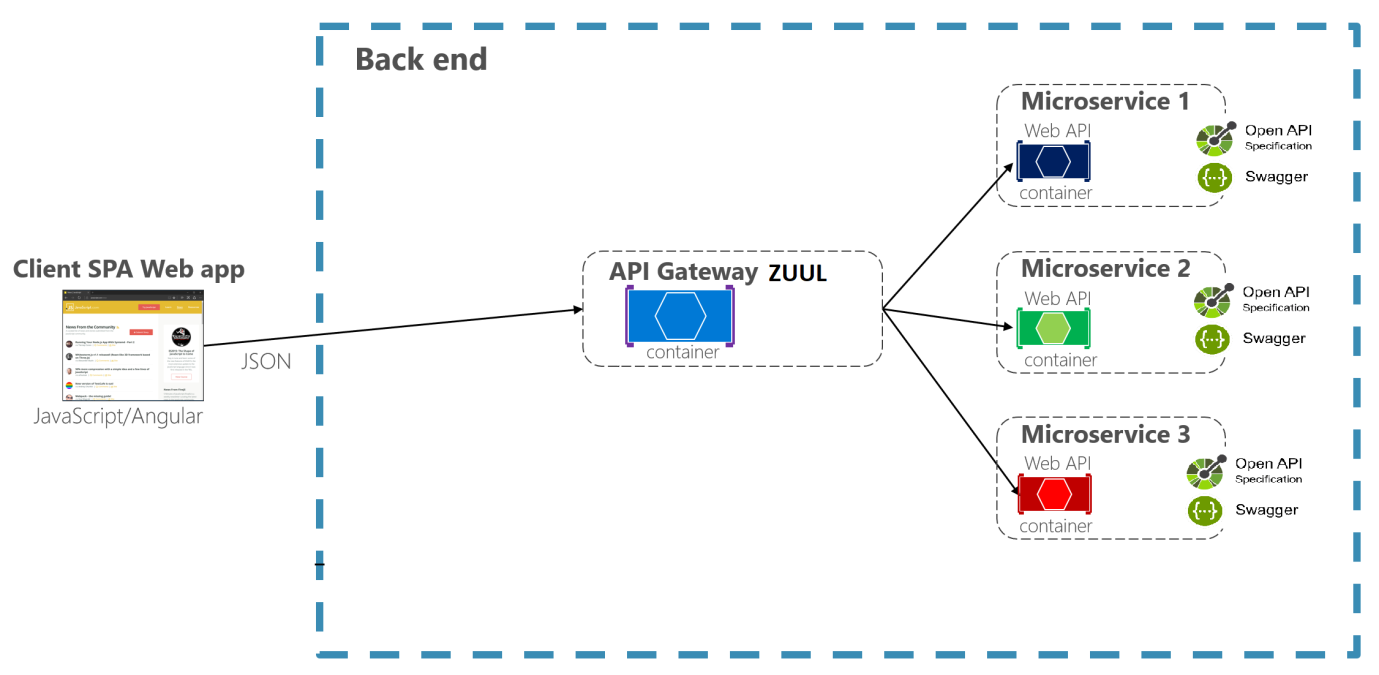
## Microservice Architecture (Governance and Tooling/ Database and Storage):



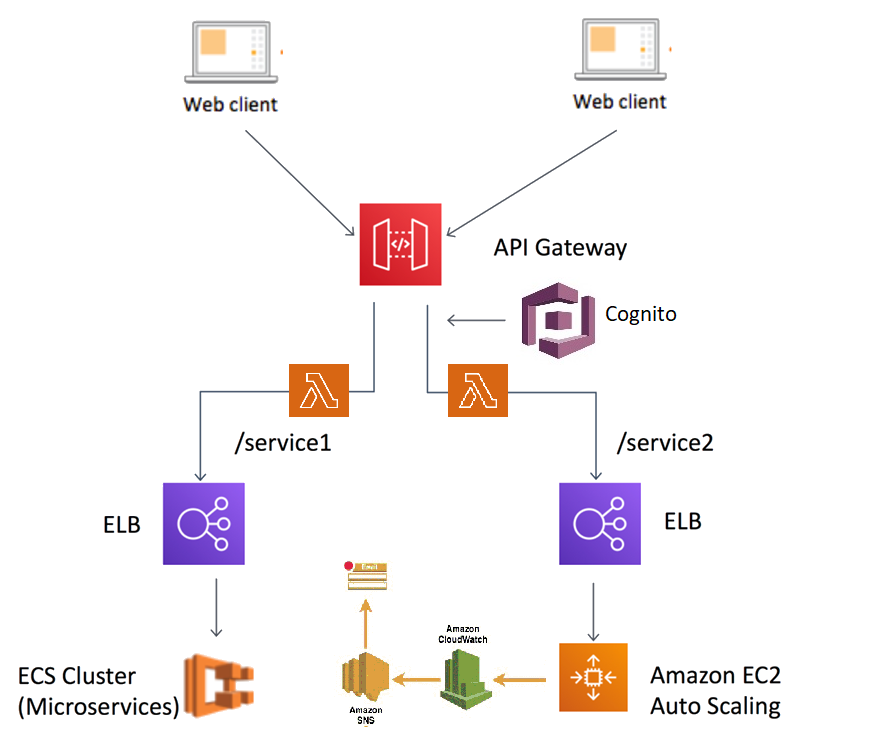
## Microservice Architecture (Governance and Tooling/ Compute and Integration):



## Microservice Architecture (Compute and Integration/Presentation/Networking and Content Delivery):



# Cloud Architecture



# Tool Chain

|  |  |  |
| --- | --- | --- |
| Competency | Skill | Skill Detail |
| Engineering Mindset | Networking and Content Delivery |  |
|  | Ways of Working |  |
|  | Consulting Mindset |  |
|  | DevOps |  |
| Programming Languages | Application Language | Java |
| Products & Frameworks | Presentation | Angular |
|  |  | Javascript/Typescript |
|  | Networking and Content Delivery | Zuul |
|  | Security and Identity | OpenIAM |
|  | Compute & Integration | Spring Boot |
|  |  | ELK Stack |
|  |  | Docker |
|  |  | Kubernetes |
|  | Database & Storage | MongoDB |
|  |  | MySQL |
|  | Governance & Tooling | Git |
|  |  | Junit |
|  |  | Mockito |
| Engineering Quality | Secure Coding/ Code Quality | Veracode |
|  |  | PMD/SonarQube |
| Platform | Cloud Tools | AWS ECS |
|  |  | AWS DynamoDB/Aurora |
|  |  | AWS Lambda |
|  |  | AWS ElasticCache |
|  |  | AWS CodeDeploy |
|  |  | AWS API Gateway |
|  |  | AWS ELB(Elastic Load Balancer) |
|  |  | AWS SNS |
|  |  | AWS SSO |
|  |  | AWS Cognito |
|  |  | AWS CloudWatch |

# Development Flow

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **MC** | **Competency** | **Section** | **Indicative**  **Mechanism for Evaluation ( Passing score of 60% in each MC)** | **Points to Note** |
| [**Business Requirement**](#_Business-Requirement:) | | | | |  |
| **1** | **Backend** | Rest API, Database, Messaging System, Log/Monitoring, Identity and Access Management system, container orchestration, TDD, Debugging and Troubleshooting, automate repetitive workflows, Static and Dynamic Analysis | [Click here](#_Rubrics/Expected_Deliverables) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete this MC with 60% to proceed with the next Frontend MC |
| **2** | **Front End** | Angular/React, AJAX, Build & Deploy | [Click here](#_Frontend) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete each MC with 60% to proceed with the next Cloud MC |
| **3** | **Cloud** | Compute, Identity, Compliance, Security and Content Delivery | [Click here](#_Platform) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete this Cloud MC with 60% |

# Business Requirements:

As an application developer, develop microservices with below guidelines:

|  |  |  |  |
| --- | --- | --- | --- |
| User  Story # | User Story Name | User Story | Development |
| US\_01 | User Registration | As a user I am be able to register to the LMS system and access courses   1. As a user I am be able to furnish following details at the time of registration    1. User Name    2. User Email id    3. Create Password   Acceptance Criteria:   * All Fields are mandatory * Email id should contain “@” and “.com” * Password should be Alphanumeric and at least 8 characters | Only API to be developed |
| US\_02 | Add new course | As a admin I should be able to add any new course to the LMS  Fields to be added:   * Course Name, Course duration, Course description, Technology, Launch URL   Acceptance criteria:   * Course Name should be of minimum 20 characters * Course description should be minimum 100 characters * All the fields are mandatory * Course duration should be numeric | Only API to be developed |
| US\_03 | View and delete Course | As admin I am able to view and delete details of courses from the system.  Acceptance criteria:   * Should be able to view any particular courses * Should be able to view all the courses available in the system * Should be able to delete any courses from the system | Only API to be developed |
| US\_04 | View Courses Details | As a user I am able to view course details for any technology  Acceptance criteria:  List the courses based on provided technology  Along with listing the courses, display course duration and course description as well  UI should provide a search button to enter the course/technology  UI should provide options to select the courses to be displayed based on technology and duration range between few hours  UI should then show the course details – course name, duration, description and technology | API and Frontend to be developed |

# Proposed Rest Endpoints to be exposed

## Rest APIs:

|  |  |  |
| --- | --- | --- |
| **POST** | **/api/v1.0/lms/company/register** | **Register a new company** |
| **GET** | **/api/v1.0/lms/courses/info/<technology>** | **Fetches the course Details based on given technology** |
| **GET** | **/api/v1.0/lms/courses/getall** | **Fetches all the course Details** |
| **DELETE** | **/api/v1.0/lms/courses/delete/<coursename>** | **Deletes a course** |
| **POST** | **/api/v1.0/lms/courses/add/<coursename>** | **Add new course to the system** |
| **GET** | **/api/v/1.0/lms/courses/get/<technology>/**  **<durationFromRange>/<durationToRange>** | **Fetches courses based on the given duration** |

Use Creational design pattern for composing the model object to be sent back as response on following end-point:

* 1. **/api/v/1.0/lms/courses/get/<technology>/<durationFromRange>/<durationToRange>**

(Fetches course details)

# Rubrics/Expected Deliverables

## Engineering Concepts (Compute & Integration):

1. As an application developer:
   1. Develop the application as a microservice architecture.
   2. Implementation as follows:
      1. Use Domain Driven Design to implement distributed architecture
      2. Follow the Single Data Store per microservice practice
      3. Document REST endpoints with OpenAPI or Swagger
      4. Add CQRS pattern for Event Sourcing for saving and retrieving course details
      5. Expose all rest Endpoints using a common API Gateway Zuul

## Engineering Concepts (Security & Identity):

* + - 1. As an Application Developer:
         1. Restrict the access over all write operation (secured operations) by adding authentication
         2. Secure all Rest EndPoints by configuring SSL Certificate for Cloud
         3. Use OpenIAM to add OTP based two factor authentication for secured operations

## Products & Framework (Database & Storage):

1. As an application developer:
   1. Implement ORM with Spring Data MongoRepository and MongoDB. For complex and custom queries, create custom methods and use @Query, Aggregations (AggregationOperation, MatchOperation, AggregationResults), implementation of MongoTemplateetc as necessary.
   2. Use MySQL for maintaining data for atleast one of the microservice
   3. Introduce a backup mechanism, such that when record count crosses 10,000 rows, a backup should trigger

## Products & Framework (Governance & tooling):

1. As an application developer
   1. Containerize the complete application, using docker and Dockerfile.
   2. Orchestrate container instances using kubernetes
   3. Use .dockerignore as necessary to avoid containerizing un-necessary packages.
   4. Integrate Spring Boot Actuator with Prometheus/Grafana to monitor middleware.
   5. Assemble logs from all microservice through kafka in Logstash.
   6. Pass on logstash data to ElasticSearch and organize them index. One index for each microservice.
   7. Perform unit and integration testing of your application and do proper CI/CD
   8. Pass this indexed data from ElasticSearch to Kibana and generate charts for each microservice logs

## Debugging & Troubleshooting

1. Generate bug report & error logs - Report must be linked with final deliverables which should also suggest the resolution for the encountered bugs and errors.

## Code Quality/Optimizations

1. Optimize MongoDB implementations using index search and then report the updated response time metrics.
2. Use Veracode to scan the backend application for security vulnerabilities, and submit the report of the same
3. Associates should have written clean code that is readable
4. Associate should have used the Code Analyzer (PMD/SonarQube) to ensure code quality and standard code style.

# Frontend

* Develop the front end for user story #US\_04.
* Implement using either Angular or React
* Implement all the Front-End validation rules
* Proper naming conventions and folder structures
* Implement using proper SOLID design principles
* Perform unit and integration testing for the front end application

# Platform

## Compute

Use User Story-4 from the Business Requirements to implement the below.

* + - 1. Use ECS CLI (as an alternative to AWS Management Console) for container management and deployment of microservices. You should be able to explain and demonstrate the same in interview.
      2. Use NoSQL instance of AWS DynamoDB/Aurora(SQL) as a database for the Market Application

## Compute, Identity & Compliance, Security& Content Delivery

1. Use AWS Lambda and AWS Aurora to build a backend process for handling requests for Market App.
2. Use Amazon API Gateway to expose the Lambda functions built in the previous step to be accessible on public internet.
3. Orchestrate containers with AWS ECS to run the Microservices
4. Use AWS ELB(Network Load Balancer) to configure the load balancing of the instances
5. Configure AWS SNS to issue messages whenever Auto Scaling is launching or terminating the EC2 instances in the Auto Scaling group
6. Use AWS SSO(Single Sign On) for authentication.
7. Use AWS Cognito to SignUp the user for secured operations.
8. Configure CloudWatch to receive, maintain and expose metrics of all the write operations (Adding new course details)

Note : Minimum 3 API’s to be hosted in cloud

## FrontEnd Deployment

Deploy the Frontend solution as follows:

Maintain the production build of Frontend application on S3 bucket

Configure an EC2 instance to access Frontend artifacts from S3 bucket and expose it as dynamic web application

Configure the S3 to cache the Javascript build files

Configure the Route 53 to register domain name to expose the Frontend solution.

Ensure that Privacy Protection feature is enabled for Amazon Route 53 domain.

# Methodology

## Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. As an application developer, the scope of discussion with mentor is limited to:
   1. Q/A
   2. New Ideas, New feature implementations and estimation.
   3. Any development related challenges
   4. Skill Gaps
   5. Any other pointers key to UI/UX and Middleware Development