System Requirements Specification

Index

For

Learning Management System

Version 1.0

TABLE OF CONTENTS

LEARNING MANAGEMENT SYSTEM

System Requirements Specification

You need to consume APIs exposed by Backend application in Angular to make application work as FULLSTACK

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

The **Learning Management System (LMS)** is a comprehensive web application that facilitates the management and delivery of educational courses and training programs. It consists of a backend implemented using a technology stack such as Spring Boot, a MySQL database for data storage, and a frontend developed using Angular.

Following is the requirement specifications:

	Learning Management System
Modules	
1	LMS
LMS Module Functionalities	
1	Add a LMS
2	Update the existing LMS details
3	Get the LMS by Id
4	Get all LMS
5	Remove an existing LMS
6	Search LMS by title

2 Assumptions, Dependencies, Risks / Constraints

2.1 LMS Constraints:

- When fetching LMS details by ID, if the LMS ID does not exist, the operation should throw a custom exception.
- When updating a LMS, if the LMS ID does not exist, the operation should throw a custom exception.
- When removing a LMS, if the LMS ID does not exist, the operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in dto classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 Business Validations

- Title is not null, min 1 and max 100 characters.
- Description is not null.
- Instructor is not null.
- Duration is not null.
- Start Date is not null and should be in valid date time format (“yyyy-mm-dd”).
- End Date is optional and should be in valid date time format (“yyyy-mm-dd”).
- Syllabus is optional.

4 REST ENDPOINTS

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 LMSCONTROLLER

URL Exposed		Purpose
1. /lms		Fetches all the LMS
Http Method	GET	
Parameter	-	
Return	List<LMS>	
2. /lms		Add a new LMS
Http Method	POST	
Parameter 1	LMS	
Return	LMS	
3. /lms/{id}		Delete LMS with given id
Http Method	DELETE	
Parameter 1	Long (id)	
Return	Boolean	
4. /lms/{id}		Fetches the LMS with the given id
Http Method	GET	
Parameter 1	Long (id)	
Return	LMS	
5. /lms/{id}		Updates existing LMS
Http Method	PUT	
Parameter 1	Long (id)	
Parameter 2	LMS	
Return	LMS	
6. /lms/search?title={title}		Search the LMS with the given title
Http Method	GET	
Parameter 1	Sring (title)	
Return	List<LMS>	

5 TEMPLATE CODE STRUCTURE

5.1 PACKAGE: COM.LMS

Resources

LMSApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.LMS.REPOSITORY

Resources

Class/Interface	Description	Status
LMSDAO (interface)	o Repository interface exposing CRUD functionality for LMS Entity. o You can go ahead and add any custom methods as per requirements.	Partially implemented.

5.3 PACKAGE: COM.LMS.SERVICE

Resources

Class/Interface	Description	Status
LMSService (interface)	• Interface to expose method signatures for LMS related functionality. • Do not modify, add or delete any method.	Already implemented.

5.4 Package: com.lms.service.impl

Resources

Class/Interface	Description	Status
LMSServiceImpl (class)	• Implements LMSService. • Contains template method implementation. • Need to provide implementation for LMS related functionalities. • Do not modify, add or delete any method signature	To be implemented.

5.5 Package: com.lms.controller

Resources

Class/Interface	Description	Status
LMSController (Class)	1. Controller class to expose all rest-endpoints for LMS related activities. 2. May also contain local exception handler methods	To be implemented

5.6 PACKAGE: COM.LMS.DTO

Resources

Class/Interface	Description	Status
LMSDTO (Class)	Use appropriate annotations from the Java Bean Validation API for validating attributes of this class.	Partially implemented.

5.7 PACKAGE: COM.LMS.ENTITY

Resources

Class/Interface	Description	Status
LMS (Class)	• This class is partially implemented. • Annotate this class with proper annotation to declare it as an entity class with **lmsId** as primary key. • Map this class with a **lms table**. • Generate the **lmstId** using the IDENTITY strategy	Partially implemented.

5.8 PACKAGE: COM.LMS.EXCEPTION

Resources

Class/Interface	Description	Status
CustomException (Class)	• Custom Exception to be thrown when trying to fetch or delete the LMS info which does not exist.	Already implemented.

	• Need to create Exception Handler for same wherever needed (local or global)	
ResourceNotFoundException (Class)	• Custom Exception to be thrown when trying to fetch or delete the LMS info which does not exist. • Need to create Exception Handler for same wherever needed (local or global)	Already implemented.
RestExceptionHandler (Class)	• RestControllerAdvice Class for defining global exception handlers. • Contains Exception Handler for **InvalidDataException** class. • Use this as a reference for creating exception handler for other custom exception classes.	Already implemented.

6 CONSIDERATIONS

A. There is no roles in this application
B. You can perform the following possible action

LMS

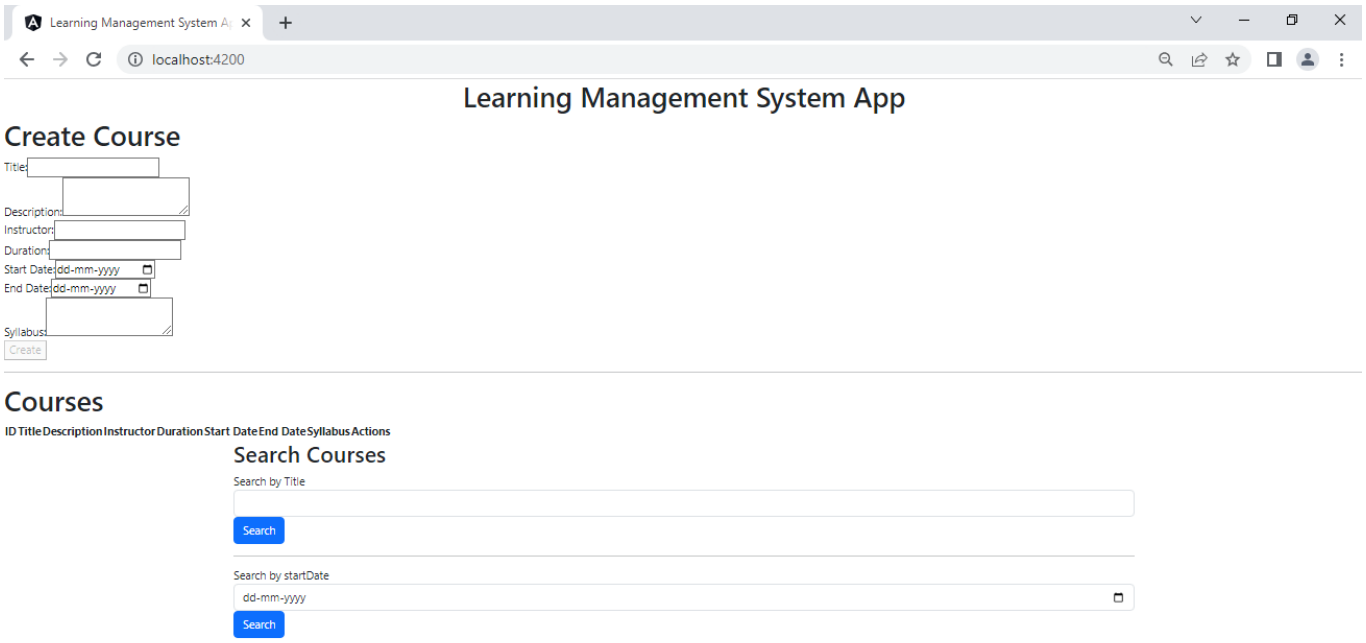
FRONTEND-ANGULAR SPA

1 PROBLEM STATEMENT

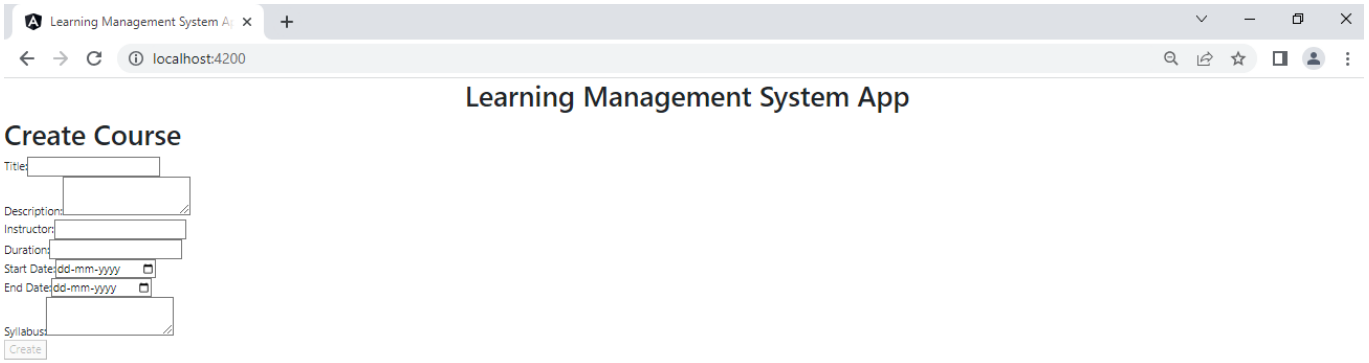
Learning Management System is SPA (Single Page Application), it allows to manage the courses with functionalities to create a new course, update an existing course, get detailed information, and search any particular course.

2. PROPOSED Learning Management System MANAGEMENT WIREFRAME

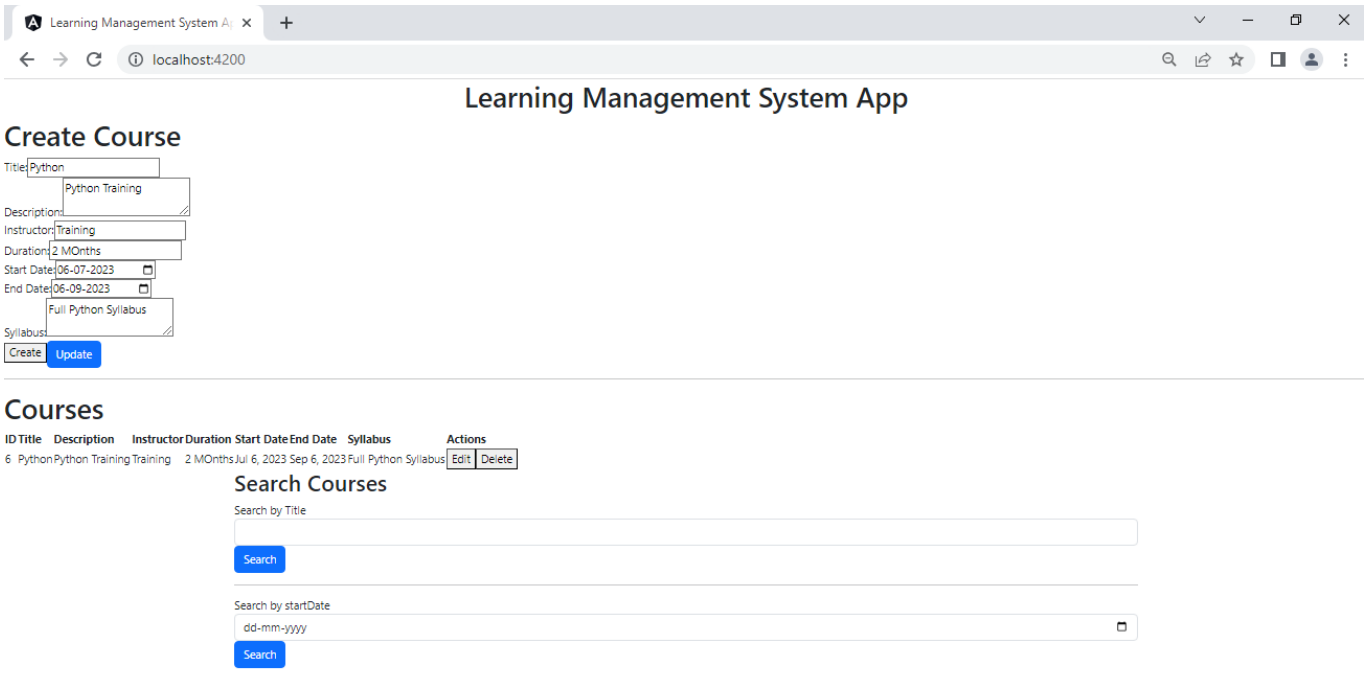
2. 1 HOME PAGE



AFTER ADDING THE COURSES:



ON CLICK UPDATE BUTTON:



3. BUSINESS-REQUIREMENT:

As an application developer, develop the Learning Management System App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Home Page	As a user I should be able to visit the home page as default page, where I can see a form to create or update the course, list of all courses with options to edit and delete any course and in the last two should be a form to search any course on its title or start date criteria.
US_01	Home Page	As a user I should be able to see the homepage and perform all operations: Acceptance criteria: 1. As a user I should be able to furnish the following details at the time of creating the course. 1.1 Title 1.2 Description 1.3 Instructor 1.4 Duration 1.5 Start date 1.6 End date 1.7 Syllabus 2. Create button should be disabled until all fields are validated. 3. Update button should be displayed when you click on the Edit button. 4. Title, Description, Instructor, Duration, Start date fields are mandatory. If any of these field is missing or if any constraint is not satisfied then must show a message. 5. Form control names should be case sensitive and they should be like as follows: title description instructor duration startDate

		endDate syllabus

.

1 EXECUTION STEPS TO FOLLOW FOR BACKEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **cd into your backend project folder**
4. **To build your project use command:**
   mvn clean package -Dmaven.test.skip
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**
   java -jar <your application jar file name>
6. **This editor Auto Saves the code.**
7. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
8. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
9. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**
10. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
11. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**

11. To login to mysql instance: Open new terminal and use following command:
    a. sudo systemctl enable mysql
    b. sudo systemctl start mysql
    c. mysql -u root -p
       The last command will ask for password which is ‘pass@word1’
12. Mandatory: Before final submission run the following command:
       mvn test

13. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

7 EXECUTION STEPS TO FOLLOW FOR FRONTEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to**

   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. **This is a web-based application, to run the application on a browser, use the internal browser in the environment.**
4. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min**
   b. **npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min**
   c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
5. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**