**Faculty Management System**

**1.ABSTRACT:** Faculty Management System aims to create a web application using Spring Boot, Java, Spring MVC, Thyme leaf, Spring Data JPA, and MySQL database. In this project, one can add a faculty, view the list of faculties, update any faculty, delete a faculty and the courses they teach. Faculty information can be easily maintained with the help of this project.

**2.USE CASE DIAGRAM:**

Diagram

Description automatically generated

**3.IMPLEMENTATION DETAILS:**

I have implemented a CRUD operation for the faculty entity in our **Faculty Management System** web application.

**Tools and Technologies Used**

* Java 16
* Spring Boot
* Spring MVC
* Spring Data JPA (Hibernate)
* MySQL
* Thyme leaf
* Eclipse STS

**1. Create Spring Boot Project**

After Creating a Spring Boot Project in Spring Tool Suite and add the dependencies while creating spring boot project using spring initializer:

- Spring Web

- Thymeleaf

- Spring Data JPA

- MySQL Driver

- Spring Boot Devtools

**2. Create Spring Boot Project Structure**

Let's create the below packages in our Spring boot project:

**Controller**- When a user requests from the browser, it first goes to the controller. From the controller, the request is passed to the service layer. So, in this layer A FacultyController class with handler methods is created to handle the get and post requests.

**Service**- This service layer has the business logic and is responsible to send the further request to repository. In this layer, an interface FacultyService is created which has all the abstract methods.

**ServiceImpl-** In this folder, the interface of the service layer is implemented by the class FacultyServiceImpl which implements the functionality of all the methods.

**Repository**- In this folder, an interface FacultyRepository that extends the JPARepository is created which acts as a mediator between service and the database.

**Entity**- In this folder, the entity Faculty is created which has the attributes FirstName, Last Name, Course Name and email.

The annotation @Entity indicates that the class is an entity. This annotation can be used on classes and enums' interfaces.

The JPA annotation @Table provides the database table to which this entity is mapped.

@Id - The @Id JPA annotation gives the entity's primary key.

**Configure Mysql Database:** After configuring the database in the Spring Boot project, Create database fms in Mysql Workbench.

**View Layer:** Create faculties.html, create\_faculty.html, edit\_faculty.html

I have used *th:each* Thymeleaf attribute in my template to iterate the list of faculties.

**3.Run the SpringBoot application Main class and use the below URL to access the application:**

[**http://localhost:8080/faculties**](http://localhost:8080/faculties)

**Output Screenshots:**

A screenshot of a computer

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated