Milestone SBA-2

Create a Student registration form where user can create student , fetch student and update an student data. Student has attributes like id, name, marks. Work on the below functionalities.

1. Create a Student data with spring boot RESTAPI with JPA. Create minimum 7 to 10 Student.

Spring boot, Mysql, Spring Jpa

Creating Student spring boot application

```
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class StudentAppApplication {

    public static void main(String[] args) {

      SpringApplication.run(StudentAppApplication.class, args);

   }


}
```

Application properities

```
spring.datasource.url=jdbc:mysql://localhost:3306/student_db?userSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false

spring.datasource.username=root

spring.datasource.password=root

server.port=8084

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto=update
```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.spring</groupId>
    <artifactId>student_app</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>student-app</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>



            <build>
                <plugins>
                    <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                    </plugin>
                </plugins>
            </build>

        </project>
```

Student jpa entity

```
package com.student_app.data.models;

import javax.persistence.*;

import java.util.Objects;


@Entity

public class Student {

   @Id
```

```
@GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  private int marks;
   public Student(){}
  public Integer getId() {
     return id;
  }
  public void setId(Integer id) {
     this.id = id;
  }
  public String getName() {
     return name;
  }
  public void setName(String Name) {
     this.name = name;
  }
  public void getMarks() {
     return marks;
  }
  public void setMarks(Integer Marks) {
     this.marks = marks;
  }


  @Override
  public String toString() {
     return "Student{" +
           "id=" + id +
           ", name='" + name + '\'' +
```

```
            ", marks='" + marks + '\'' }
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Student  student = (Student) o;
        return Objects.equals(id, student.id) && Objects.equals(name, student.name) &&
Objects.equals(marks, student.marks)
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, marks);
    }
}
```

Student jpa repository

```
package com.student_app.data.repository;

import com.student_app.data.models.student;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface StudentRepository extends JpaRepository<Student, Integer> {
}
```

Student Service

```
package com.student_app.service;

import com.student_app.data.models.Student;
import com.student_app.data.payloads.request.StudentRequest;
import com.student_app.data.payloads.response.StudentResponse;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.Optional;

@Component
public interface StudentService {
   MessageResponse createStudent(Request studentRequest);
   Student getASingleStudent(Integer StudentId);
   List<Employee> getAllStudent();
}
```

Student Controller

```
package com.student_app.web;

import com.student_app.data.models.Student;
import com.student_app.data.payloads.request.StudentRequest;
import com.student_app.data.payloads.response.MessageResponse;
import com.student_app.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
```

```
@RestController
@RequestMapping("/student")
public class StudentController {

   @Autowired
   StudentService StudentService;

   @GetMapping("/all")
   public ResponseEntity<List<Student>> getAllStudents () {
      List<Student> Students = studentService.getAllStudents();
      return new ResponseEntity<>(students, HttpStatus.OK);
   }
   @GetMapping("/find/{id}")
   public ResponseEntity<Student> getStudentById (@PathVariable("id") Integer id) {
      Student student= studentService.getASinglestudent(id);
      return new ResponseEntity<>(student, HttpStatus.OK);
   }
}
```

2. Get all students from database based on their name in ascending order.

```
package com..student_app.service;
import com..student_app.data.models.Student;
import com.student_app.data.payloads.request.StudetnRequest;
import com.student_app.data.payloads.response.MessageResponse;
import com._app.data.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;
```

```
@Service
public class StudentServiceImpl implements StudentService {
   @Autowired
   StudentRepository studentRepository;

   @Override
   public MessageResponse createStudent(StudentRequest studentRequest) {
      Student newStudent = new Student();
        newStudentsetId(studentRequest.getid());

      newStudentsetName(studentRequest.getname());
      newStudent.setMarks(employeeRequest.getmarks());

      return new MessageResponse("New student created successfully");

   }

   @Override
   public Optional<Student> updateStudent(Integer StudentId, StudentRequest StudentRequest)
throws ResourceNotFoundException{
      Optional<Student> student = studentRepository.findById(studentId);
      if (Student.isEmpty()){
      throw new ResourceNotFoundException("Student", "id", studentId);
      }
      Else
      student.get().setId(studentRequest.getid());
      student.get().setName(studentRequest.getname());
      studentget().setMarks(studentRequest.getmarks());

     studentRepository.save(student.get());
      return student;
```

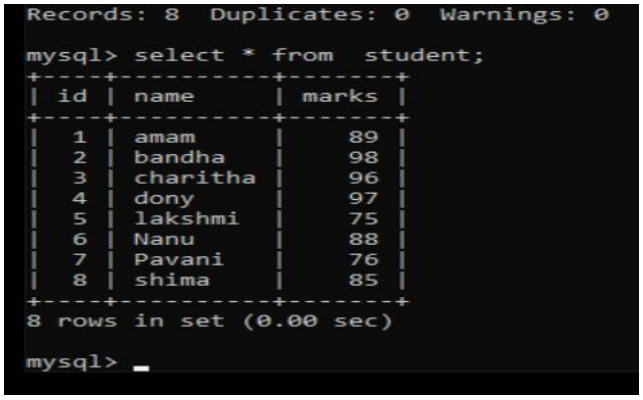
```
    }
   @Override
    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }
}
```

We have already entered all students into postman they must be saved in databse

Output:

Select*from employee:

Will display all the above students in ascending order of names from mysql database



```
Records: 8  Duplicates: 0  Warnings: 0

mysql> select * from  student;
+----+----------+-------+
| id | name     | marks |
+----+----------+-------+
|  1 | amam     |    89 |
|  2 | bandha   |    98 |
|  3 | charitha |    96 |
|  4 | dony     |    97 |
|  5 | lakshmi  |    75 |
|  6 | Nanu     |    88 |
|  7 | Pavani   |    76 |
|  8 | shima    |    85 |
+----+----------+-------+
8 rows in set (0.00 sec)

mysql>
```

3. Get a particular Student by giving the studentId as path variable.

StudentServiceImpl application

```
import com.student_app.data.payloads.response.MessageResponse;
import com.student_app.data.repository.StudentRepository;
import com.student_app.exception.ResourceNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.util.List;

import java.util.Optional;


@Service

public class StudentServiceImpl implements StudentService {

   @Autowired

   StudentRepository StudentRepository;


   @Override

   public MessageResponse createStudent(StudentRequest StudentRequest) {

      Student newStudent = new Student();

      newStudent.setFirstName(studentRequest.getName());

      newStudent.setLastname(studentRequest.geMarks());


      return new MessageResponse("New student created successfully");

}

@Override

   public Student getASingleStudent(Integer studentId) throws ResourceNotFoundException{

      return studentRepository.findById(studentId).orElseThrow(() -> new
ResourceNotFoundException(“id”,studentId);

   }
```

Output:

In post man place select Get:https//localhost:8084/api/students/2

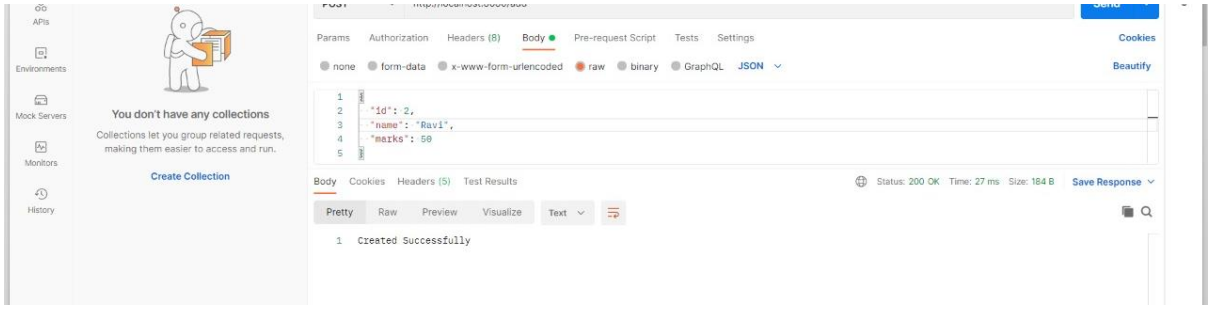

```
{
"id":2
"name":"bandha"
:marks":98
}
```

4. Update an Student mark by giving the Student id along with mark by path variable. After successfully updating show the updated value in the postman.

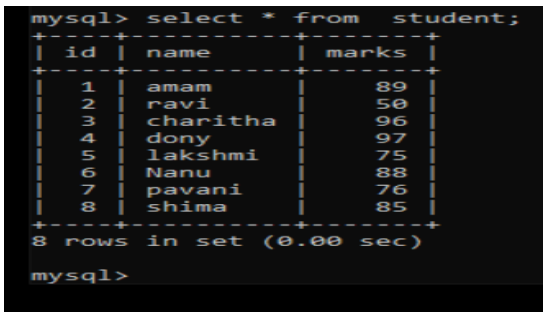
```
import com.student_app.data.payloads.response.MessageResponse;
import com.student_app.data.repository.StudentRepository;
import com.student_app.exception.ResourceNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class StudentServiceImpl implements StudentService {
   @Autowired
   StudentRepository StudentRepository;

   @Override
   public MessageResponse createStudent(StudentRequest StudentRequest) {
      Student newStudent = new Student();
      newStudent.setFirstName(studentRequest.getName());
      newStudent.setLastname(studentRequest.geMarks());

      return new MessageResponse("New student created successfully");

   }

   @Override
```

```
    public Optional<Student> updateStudent(Integer StudentMarks, StudentName StudentRequest
studentRequest)  throws ResourceNotFoundException{

        Optional<Student> student = studentRepository.findById(StudentId);

        if (student.isEmpty()){

        throw new ResourceNotFoundException("Student", "id", studentId);

        }

        else

        student.get().setName(studentRequest.getName());

        studentget().setMarks(studentRequest.getMarks());


        studentRepository.save(student.get());

        return student;

    }
```

Updated new student name and marks at id=2



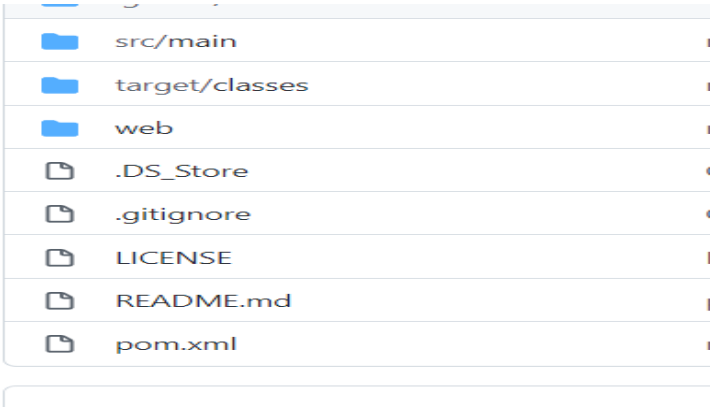
The updated table



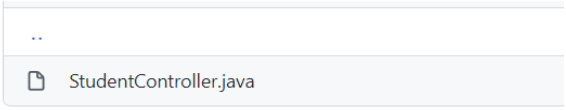
```
mysql> select * from  student;
+----+----------+-------+
| id | name     | marks |
+----+----------+-------+
|  1 | amam     |    89 |
|  2 | ravi     |    50 |
|  3 | charitha |    96 |
|  4 | dony     |    97 |
|  5 | lakshmi  |    75 |
|  6 | Nanu     |    88 |
|  7 | pavani   |    76 |
|  8 | shima    |    85 |
+----+----------+-------+
8 rows in set (0.00 sec)

mysql>
```

After making the above application on Springboot Restapi do the following task on GIT and Jenkins.

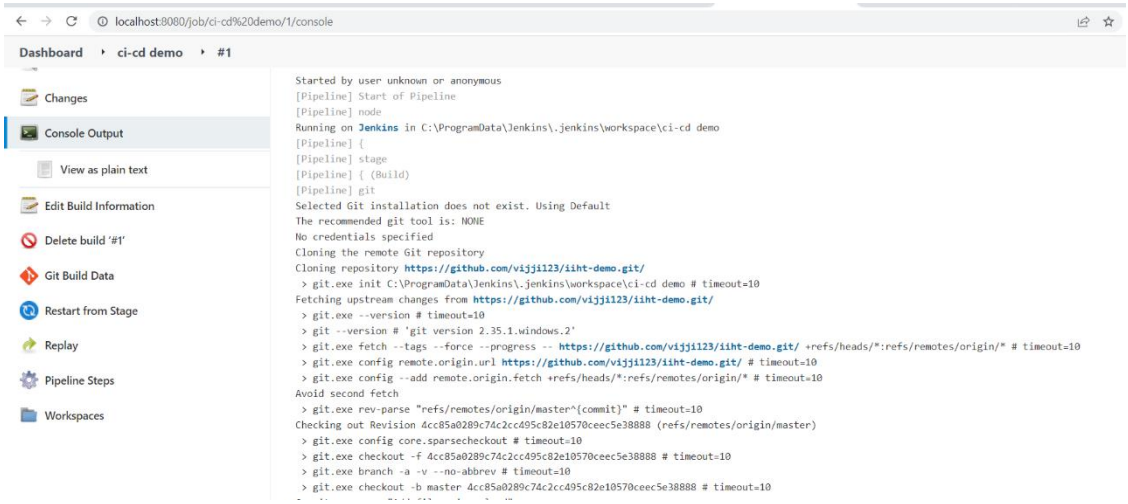
1. **Commit the code to GITHUB repository (iiht-demo)**



After commiting all the student files are inside the repository like this for every commit in src.



2. **Make a CI/CD pipeline and create a JOB in Jenkins so that when an commit is happening it will auto trigger to Jenkins and a build will happen.**



Commit build was success.