```
Add this dep in pom.xml

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

1.Student.java

```java
package com.example.demoaiml.entity;

import jakarta.persistence.*;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;


@Entity
@Table(name = "students")
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank(message = "Name cannot be empty")
    private String name;

    @NotBlank(message = "Email cannot be empty")
    @Email(message = "Invalid email format")
    private String email;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
    public Student(Long id, @NotBlank(message = "Name cannot be
empty") String name,
            @NotBlank(message = "Email cannot be empty")
@Email(message = "Invalid email format") String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    public Student() {
    }



}
```

2.StudentRepository.java

```java
package com.example.demoaiml.repository;


import org.springframework.data.jpa.repository.JpaRepository;

import com.example.demoaiml.entity.Student;

public interface StudentRepository extends JpaRepository<Student,
Long> {
}
```

3.StudentService.java

```java
package com.example.demoaiml.service;


import com.example.demoaiml.entity.Student;
import java.util.List;

public interface StudentService {

    Student saveStudent(Student student);
    Student getStudentById(Long id);
    List<Student> getAllStudents();
    Student updateStudent(Long id, Student student);
    void deleteStudent(Long id);
}
```

4.StudentServiceImple.java

```java
package com.example.demoaiml.serviceimpl;


import org.springframework.stereotype.Service;

import com.example.demoaiml.repository.StudentRepository;
import com.example.demoaiml.entity.Student;
import com.example.demoaiml.exception.ResourceNotFoundException;
import com.example.demoaiml.service.StudentService;
import java.util.List;

@Service
public class StudentServiceImpl implements StudentService {

    private final StudentRepository repo;

    public StudentServiceImpl(StudentRepository repo) {
        this.repo = repo;
    }

    @Override
    public Student saveStudent(Student student) {
        return repo.save(student);
    }

    @Override
    public Student getStudentById(Long id) {
        return repo.findById(id)
                .orElseThrow(() -> new
ResourceNotFoundException("Student not found"));
    }

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

    @Override
    public Student updateStudent(Long id, Student student) {
        Student existing = getStudentById(id);
        existing.setName(student.getName());
        existing.setEmail(student.getEmail());
        return repo.save(existing);
    }

    @Override
    public void deleteStudent(Long id) {
        Student student = getStudentById(id);
        repo.delete(student);
    }
}
```

4.StudentController.java

```java
package com.example.demoaiml.controller;


import jakarta.validation.Valid;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.example.demoaiml.entity.Student;
import com.example.demoaiml.service.StudentService;
import java.util.List;

@RestController
@RequestMapping("/students")
public class StudentController {

    private final StudentService service;

    public StudentController(StudentService service) {
        this.service = service;
    }

    // CREATE
    @PostMapping
    public ResponseEntity<Student> createStudent(@Valid @RequestBody
Student student) {
        return new ResponseEntity<>(service.saveStudent(student),
HttpStatus.CREATED);
    }

    // READ ALL
    @GetMapping
    public List<Student> getAllStudents() {
        return service.getAllStudents();
    }

    // READ BY ID
    @GetMapping("/{id}")
    public Student getStudent(@PathVariable Long id) {
        return service.getStudentById(id);
    }

    // UPDATE
    @PutMapping("/{id}")
    public Student updateStudent(@PathVariable Long id,
                                 @Valid @RequestBody Student student)
{
        return service.updateStudent(id, student);
    }

    // DELETE
    @DeleteMapping("/{id}")
    public ResponseEntity<String> deleteStudent(@PathVariable Long
id) {
        service.deleteStudent(id);
        return ResponseEntity.ok("Student deleted successfully");
    }
}
```

6.ResourceNotFoundException.jaav

```java
package com.example.demoaiml.exception;

public class ResourceNotFoundException extends RuntimeException {
    public ResourceNotFoundException(String message) {
        super(message);
    }
}
```

7.GlobalExceptionHandler.java

```java
package com.example.demoaiml.exception;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;
import java.util.Map;

@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
    public ResponseEntity<String>
handleNotFound(ResourceNotFoundException ex) {
        return new ResponseEntity<>(ex.getMessage(),
HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<Map<String, String>>
handleValidationErrors(
            MethodArgumentNotValidException ex) {

        Map<String, String> errors = new HashMap<>();

        ex.getBindingResult().getFieldErrors().forEach(error ->
                errors.put(error.getField(),
error.getDefaultMessage())
        );

        return new ResponseEntity<>(errors, HttpStatus.BAD_REQUEST);
    }
}
```