1. Create a Rest API for Student using noun plurals for endpoint and http verbs for different operations.(1 Mark)

Ans.

```java
package com.ttn.rest.controllers;
import com.ttn.rest.entities.Student;
import com.ttn.rest.services.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
public class StudentController {
    @Autowired
    StudentService studentService;
    @GetMapping("/students")
    public List<Student> getStudents(){
        return studentService.getAllStudents();
    }
    @GetMapping("students/{id}")
    public Student getStudentById(@PathVariable Integer id){
        Student student = studentService.getStudentById(id);
        return student;
    }
    @PostMapping("/students")
    public Student saveStudent(@RequestBody Student student){
        studentService.saveStudent(student);
        return student;
    }

}
```

2. Create a Customise Exception Handler.(1 Mark)

Ans.

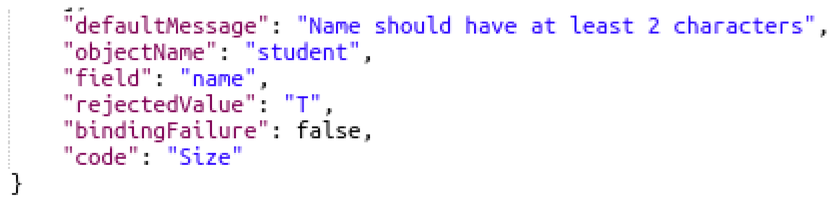
```java
package com.ttn.rest.exceptions;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeException {
    public UserNotFoundException(String message){
        super(message);
    }
}
```

3. Print the Validation Messages of student Entity in response.(1 Mark)

Ans.

```java
@Size(min = 2, message = "Name should have at least 2 characters")
private String name;
private Integer age;

@PostMapping("/students")
public Student saveStudent(@Valid @RequestBody Student student){
    studentService.saveStudent(student);
    return student;
}
```

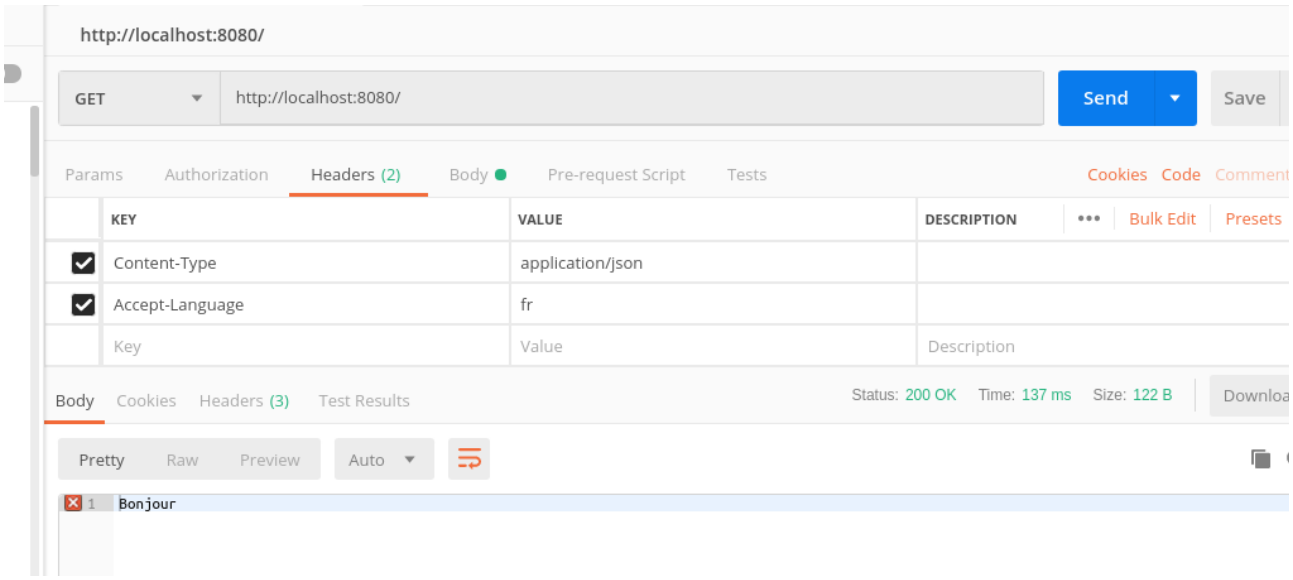


```
    "defaultMessage": "Name should have at least 2 characters",
    "objectName": "student",
    "field": "name",
    "rejectedValue": "T",
    "bindingFailure": false,
    "code": "Size"
}
```

4. Perform Internationalization for a greeting message in your app.(1 Mark)

Ans.

```
@GetMapping("/")
String helloWorld(@RequestHeader(name = "Accept-Language",required = false) Locale
locale){
    System.out.println("hello world");
    System.out.println(locale.getLanguage());
    return messageSource.getMessage("good.morning.message",null,locale);
}
```



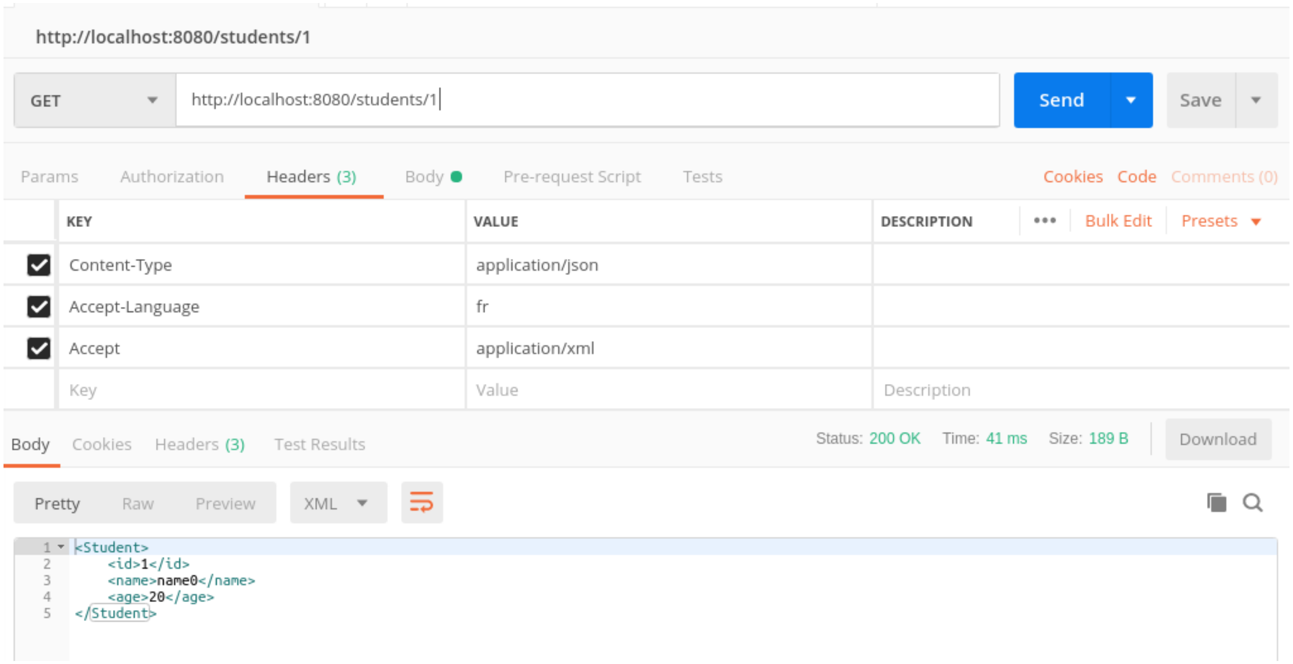
	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	Accept-Language	fr	
	Key	Value	Description

Status: 200 OK Time: 137 ms Size: 122 B

```
Bonjour
```

5. Return XML Response when new Student is created.(1 Mark)

Ans.



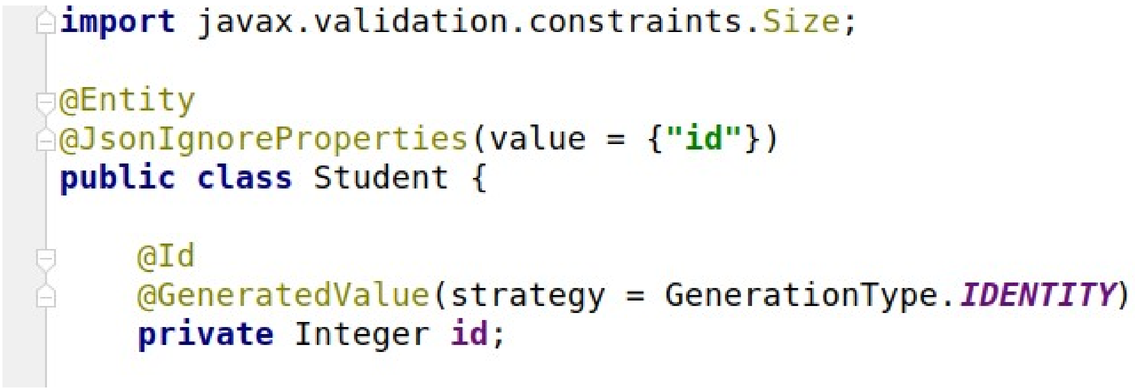
	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	Accept-Language	fr	
✓	Accept	application/xml	
	Key	Value	Description

Status: 200 OK Time: 41 ms Size: 189 B

```
<Student>
    <id>1</id>
    <name>name0</name>
    <age>20</age>
</Student>
```

6. Ignore ID field in the Response.(1 Mark)

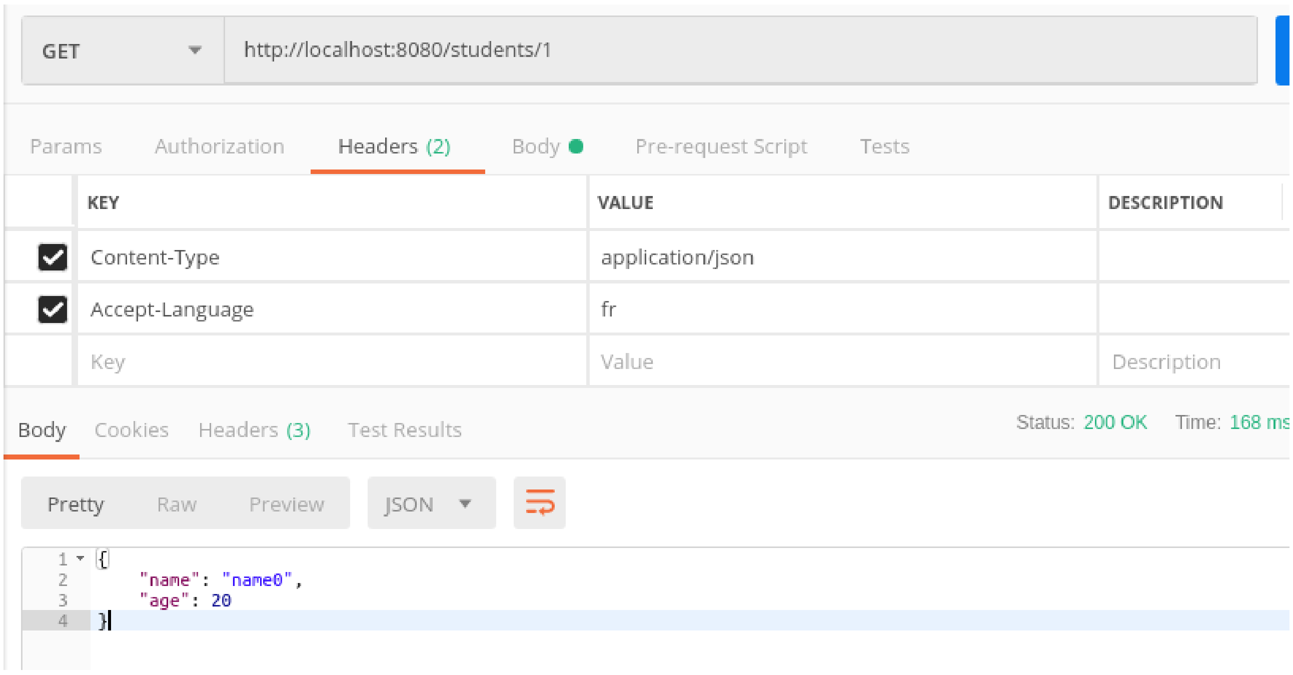
Ans.



```
import javax.validation.constraints.Size;

@Entity
@JsonIgnoreProperties(value = {"id"})
public class Student {

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



GET http://localhost:8080/students/1

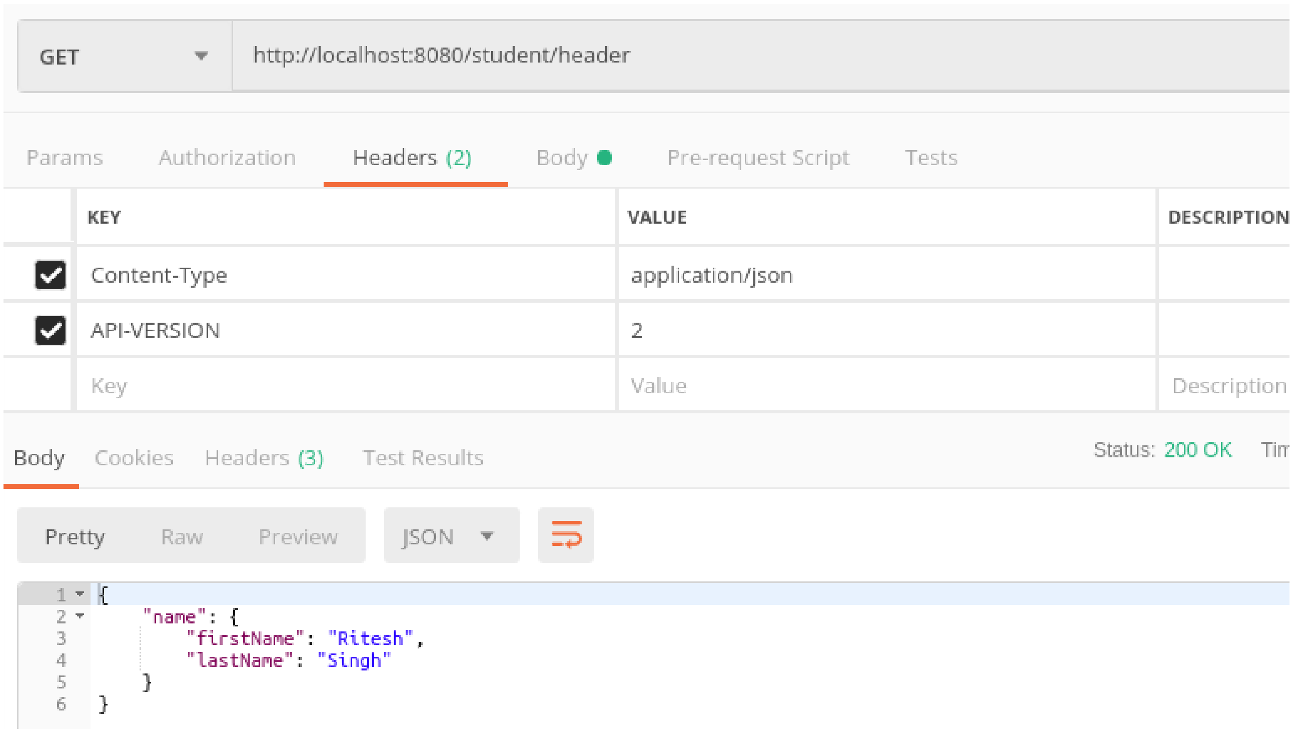
	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	Accept-Language	fr	
	Key	Value	Description

Status: 200 OK Time: 168 ms

```
{
    "name": "name0",
    "age": 20
}
```

7. Create 2 versions of your API one take reprsent name of the Student as single string and other showing firstname and lastname seperately. (Create the Versions of the API using URI, parameter and header versioning) (2 Marks)

Ans.



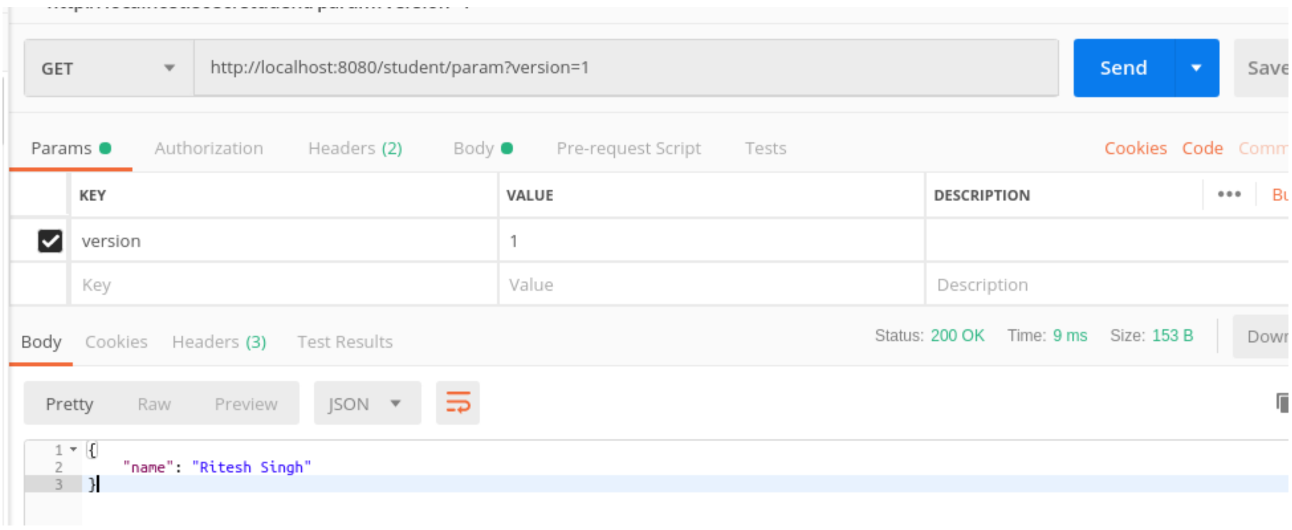
http://localhost:8080/student/header

KEY	VALUE	DESCRIPTION
✓ Content-Type	application/json	
✓ API-VERSION	2	
Key	Value	Description

Body · Cookies · Headers (3) · Test Results — Status: 200 OK

Pretty · Raw · Preview · JSON

```
{
    "name": {
        "firstName": "Ritesh",
        "lastName": "Singh"
    }
}
```



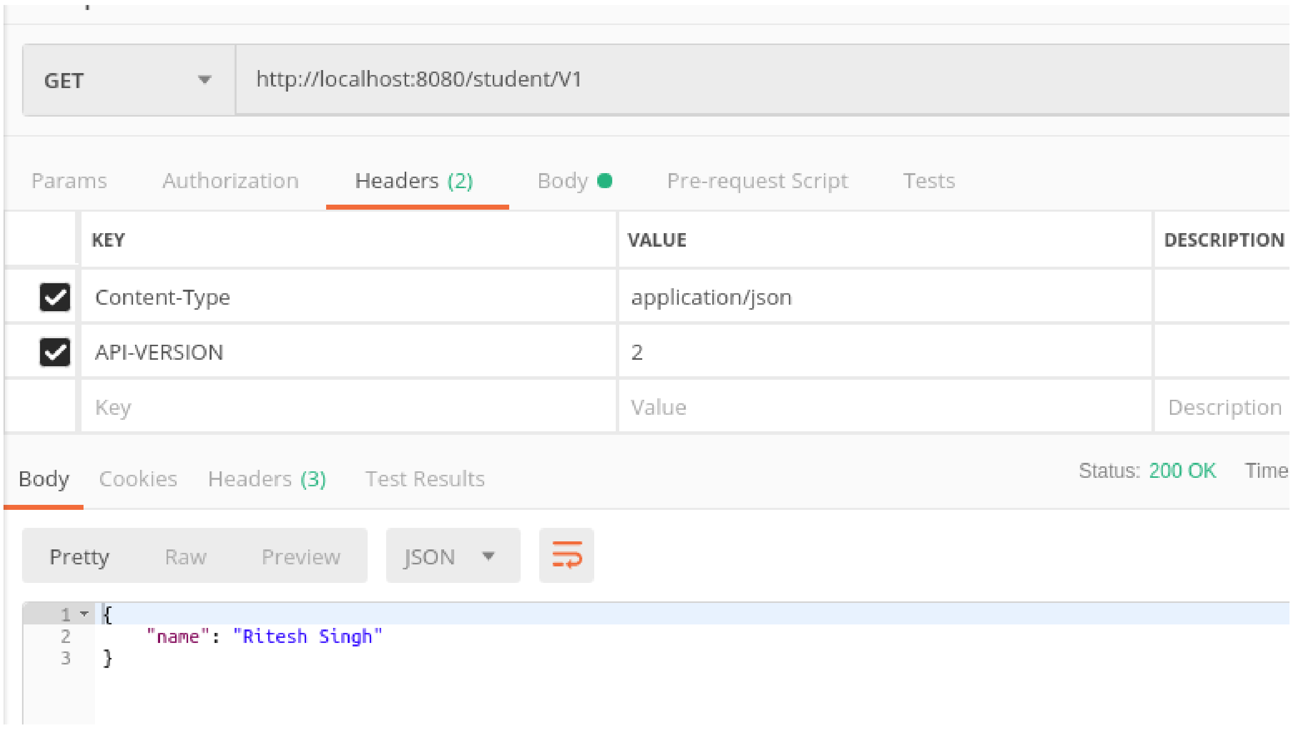
http://localhost:8080/student/param?version=1

KEY	VALUE	DESCRIPTION
✓ version	1	
Key	Value	Description

Body · Cookies · Headers (3) · Test Results — Status: 200 OK Time: 9 ms Size: 153 B

Pretty · Raw · Preview · JSON

```
{
    "name": "Ritesh Singh"
}
```



http://localhost:8080/student/V1

KEY	VALUE	DESCRIPTION
Content-Type	application/json	
API-VERSION	2	
Key	Value	Description

Status: 200 OK Time

```
{
    "name": "Ritesh Singh"
}
```

8. Perform CRUD operations on the resource below using RestTemplate

```
@PostMapping("/createPost")
public ResponseEntity<Post> createPost(){
    String url="https://jsonplaceholder.typicode.com/posts";
    RestTemplate restTemplate= new RestTemplate();
    HttpHeaders httpHeaders= new HttpHeaders();
    httpHeaders.add("Content-type","application/json; charset=UTF-8");
    HttpEntity<Post> request=new HttpEntity<>(new
Post(10000L,10000L,"title1","description1"),httpHeaders);
    Post post=restTemplate.postForObject(url,request,Post.class);
    System.out.println(post);
    return new ResponseEntity<Post>(post, HttpStatus.CREATED);
}
@GetMapping("/postList")
public List<Post> getPostList(){
    RestTemplate restTemplate = new RestTemplate();
    String url="https://jsonplaceholder.typicode.com/posts";
    ResponseEntity<List<Post>> response = restTemplate
            .exchange(url, HttpMethod.GET, null,
                    new ParameterizedTypeReference<List<Post>>(){});
    return response.getBody();
}
@PutMapping("/updatePost")
public ResponseEntity<Post> updatePost(){
    String url="https://jsonplaceholder.typicode.com/posts/1";
    RestTemplate restTemplate= new RestTemplate();
    HttpHeaders httpHeaders= new HttpHeaders();
    httpHeaders.add("Content-type","application/json; charset=UTF-8");
    HttpEntity<Post> request=new HttpEntity<>(new
Post(1L,100L,"title1","description1"),httpHeaders);
    return restTemplate.exchange(url,HttpMethod.PUT,request,Post.class);
}
@DeleteMapping("/deletePost")
public void deletePost(){
    String url="https://jsonplaceholder.typicode.com/posts/1";
    RestTemplate restTemplate= new RestTemplate();
    HttpHeaders httpHeaders= new HttpHeaders();
    httpHeaders.add("Content-type","application/json; charset=UTF-8");
    HttpEntity<Post> request=new HttpEntity<>(httpHeaders);
```

```java
restTemplate.exchange(url,HttpMethod.DELETE,request,Post.class);
```