Basic Setup

Student.java

```java
package com.demo.restWithSpring.student;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Student {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private String name;

  private Integer age;

   public Student() {
   }

   public Student(String name, Integer age) {
       this.name = name;
       this.age = age;
   }

   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 Integer getAge() {
       return age;
   }

   public void setAge(Integer age) {
       this.age = age;
   }
}
```

StudentRepository.java

```java
package com.demo.restWithSpring.student.repositories;

import com.demo.restWithSpring.student.Student;
import org.springframework.data.repository.CrudRepository;

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

StudentService.java

```java
package com.demo.restWithSpring.student.service;

import com.demo.restWithSpring.student.Student;
import com.demo.restWithSpring.student.repositories.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

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

@Service
public class StudentService {
  @Autowired
  StudentRepository studentRepository;

  public List<Student> getStudentsList()
  {
     return (List<Student>) studentRepository.findAll();
  }

  public Student addStudent(Student student)
  {
     Student savedStudent=studentRepository.save(student);
     return savedStudent;
  }
  public Student getStudentById(Long id)
  {
     Optional<Student>studentOptional=studentRepository.findById(id);
     return studentOptional.orElse(null);
  }
  public void deleteStudent(Student student)
  {
     studentRepository.delete(student);
  }
  public void saveAll(Iterable<Student>studentIterable)
  {
     studentRepository.saveAll(studentIterable);
  }
}
```

StudentNotFoundException.java

```
package com.demo.restWithSpring.student.exception;

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

DuplicateStudentFoundException.java

```
package com.demo.restWithSpring.student.exception;

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

Bootstrap.java

```
package com.demo.restWithSpring.commons.events;

import com.demo.restWithSpring.student.Student;
import com.demo.restWithSpring.student.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.event.ApplicationStartedEvent;
import org.springframework.context.event.ContextRefreshedEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;

import java.util.Arrays;
import java.util.Iterator;
import java.util.List;

@Component
public class Bootstrap {

  @Autowired
  StudentService studentService;

  @EventListener(ContextRefreshedEvent.class)
  public void insertStudentData()
  {
      List<Student> studentList=studentService.getStudentsList();
      if(studentList.isEmpty())
      {
          System.out.println("In event");
          List<Student> studentListToBeSaved = Arrays.asList(new Student("Surbhi", 23), new
Student("Vagish", 24), new Student("Reshma", 25));
```

```
        studentService.saveAll(studentListToBeSaved);
      }

   }
}
```

Q1. Create a Rest API for Student using noun plurals for endpoint and http verbs for different operations.(1 Mark)

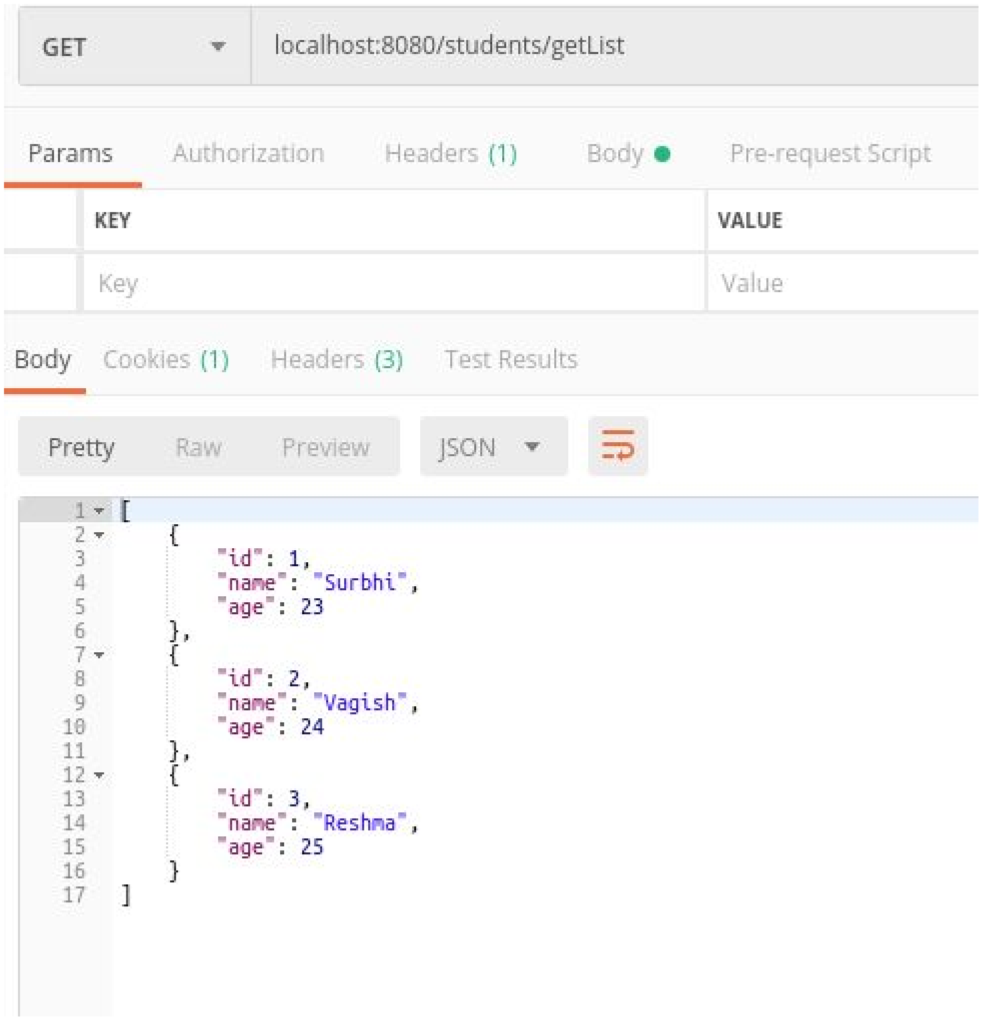
Solution

StudentController.java

```
package com.demo.restWithSpring.student.controller;

import com.demo.restWithSpring.student.Student;
import com.demo.restWithSpring.student.exception.StudentNotFoundException;
import com.demo.restWithSpring.student.service.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/getList")
   public List<Student>getStudentsList()
   {
       return  studentService.getStudentsList();
   }
   @GetMapping("/students/getStudent/{id}")
   public Student getStudent(@PathVariable Long id)
   {
      Student student=studentService.getStudentById(id);
            if(student==null)
                        throw new StudentNotFoundException("Student with id: "+id+" doesn't exist");
            return student;
   }
   @PostMapping("/students/postStudent")
   public Student addStudent(@RequestBody Student student)
   {
       Student studentToBeSaved=studentService.getStudentByName(student.getName());
            if(!(studentToBeSaved==null))
                       throw new DuplicateStudentFoundException("Student with this name already exists");

             return studentService.addStudent(student);
```

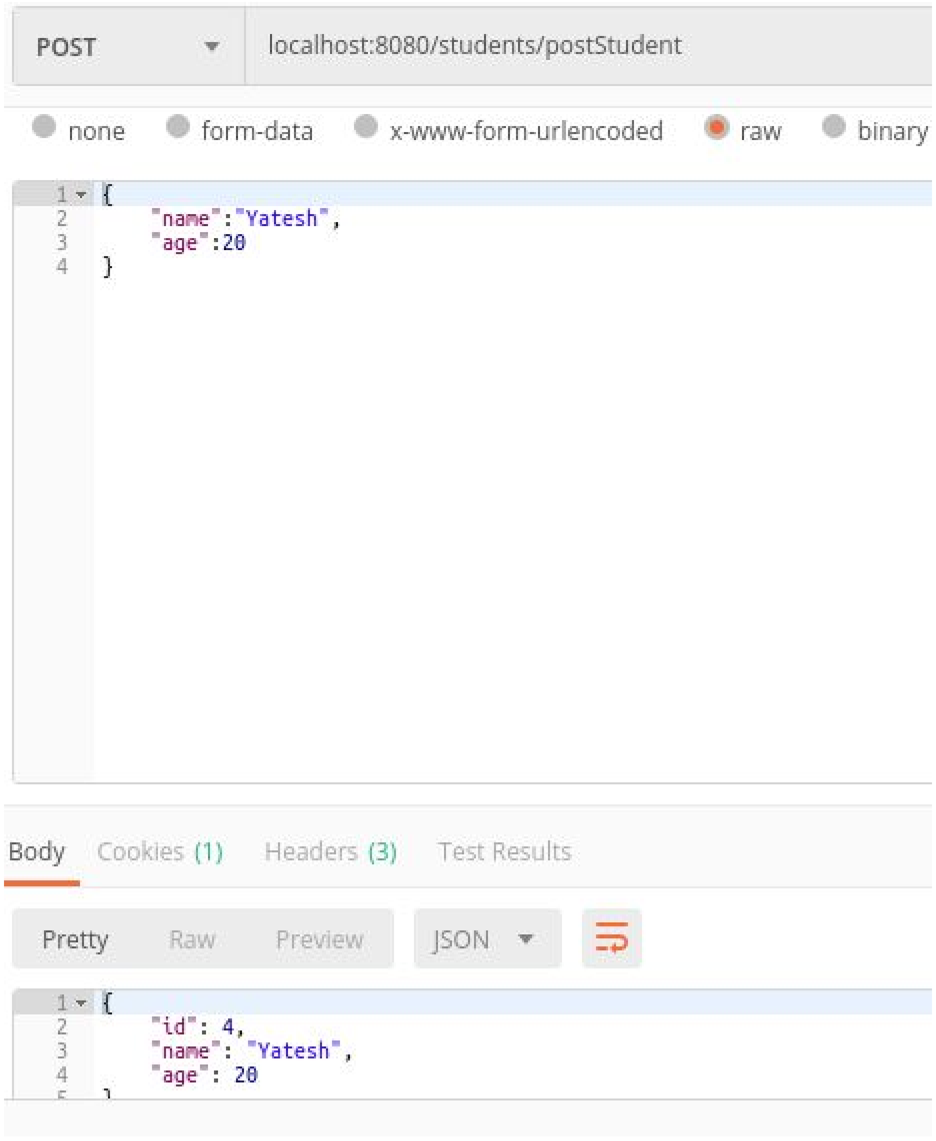
```
  }

  @PutMapping("/students/putStudent/{id}")
  public Student updateStudent(@PathVariable Long id)
  {
      Student studentToBeUpdated=studentService.getStudentById(id);
      if(studentToBeUpdated==null)
          throw new StudentNotFoundException("User with id "+id+" not found ");
       studentToBeUpdated.setName("Yukti");
       return studentService.addStudent(studentToBeUpdated);
  }

  @DeleteMapping("/students/deleteStudent/{id}")
  public void deleteStudent(@PathVariable Long id)
  {
      Student studentToBeDeleted=studentService.getStudentById(id);
      if(studentToBeDeleted==null)
          throw new StudentNotFoundException("User with id "+id+" not found ");
      studentService.deleteStudent(studentToBeDeleted);

  }

}
```

Output



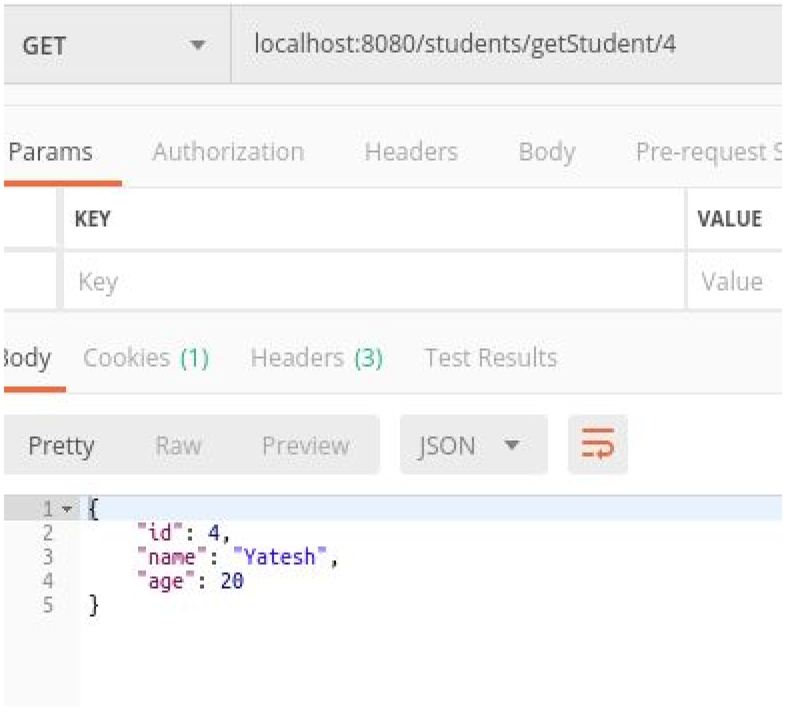
```
localhost:8080/students/getList
```

```
[
    {
        "id": 1,
        "name": "Surbhi",
        "age": 23
    },
    {
        "id": 2,
        "name": "Vagish",
        "age": 24
    },
    {
        "id": 3,
        "name": "Reshma",
        "age": 25
    }
]
```

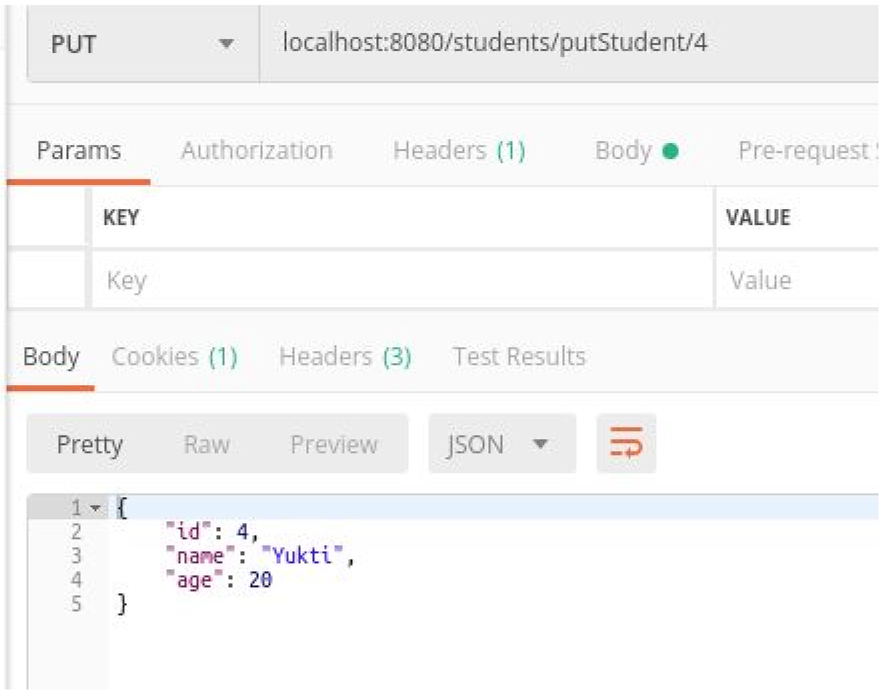


```
{
    "name":"Yatesh",
    "age":20
}
```

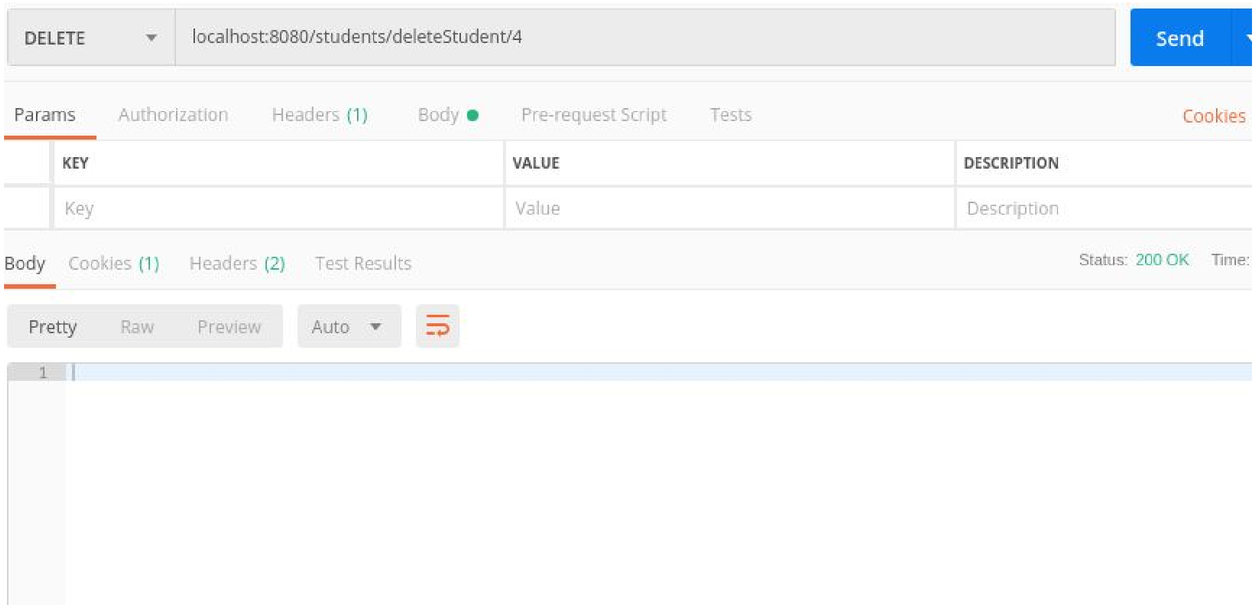
```
{
    "id": 4,
    "name": "Yatesh",
    "age": 20
}
```



```
{
    "id": 4,
    "name": "Yatesh",
    "age": 20
}
```



```
{
    "id": 4,
    "name": "Yukti",
    "age": 20
}
```



DELETE localhost:8080/students/deleteStudent/4

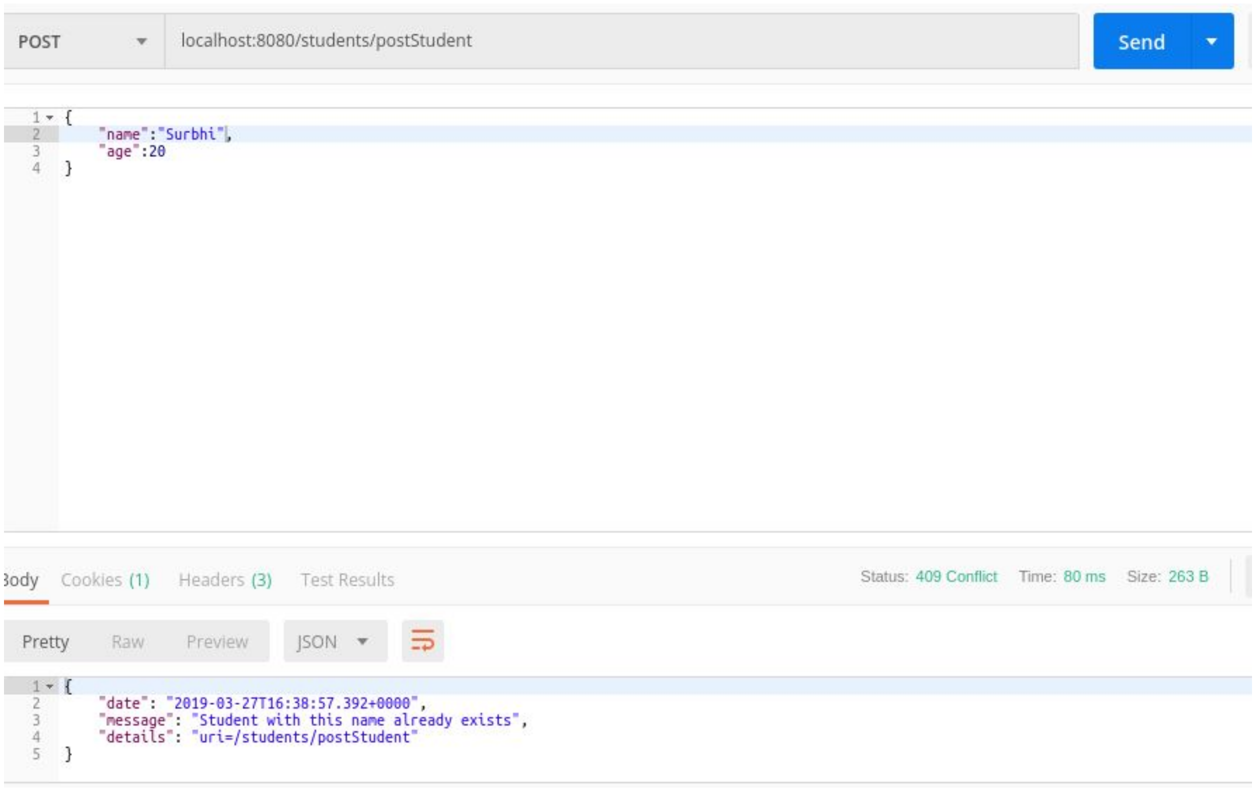
KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time:

Q2. Create a Customise Exception Handler.
Solution
ExceptionResponse.java
```
package com.demo.restWithSpring.commons.exceptionhandling;

import java.util.Date;

public class ExceptionResponse {
 private Date date;
 private String message;
 private String details;

  public ExceptionResponse(Date date, String message, String details) {
      this.date = date;
      this.message = message;
      this.details = details;
  }

  public Date getDate() {
      return date;
  }

  public void setDate(Date date) {
      this.date = date;
  }

  public String getMessage() {
      return message;
  }

  public void setMessage(String message) {
      this.message = message;
  }

  public String getDetails() {
      return details;
  }

  public void setDetails(String details) {
      this.details = details;
  }
}
```
CustomResponseEntityExceptionHandler.java
```
package com.demo.restWithSpring.commons.exceptionhandling;

import com.demo.restWithSpring.student.exception.DuplicateStudentFoundException;
import com.demo.restWithSpring.student.exception.StudentNotFoundException;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
```

```java
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.util.Date;

@RestController
@ControllerAdvice
public class CustomResponseEntityExceptionHandler extends ResponseEntityExceptionHandler {
  @ExceptionHandler(Exception.class)
  public  final ResponseEntity<Object>handleAllExceptions(Exception exception,WebRequest webRequest)
  {
     ExceptionResponse exceptionResponse=new ExceptionResponse(new
Date(),exception.getMessage(),webRequest.getDescription(false));
     return new ResponseEntity<>(exceptionResponse,HttpStatus.INTERNAL_SERVER_ERROR);
  }

  @ExceptionHandler(StudentNotFoundException.class)
  public final ResponseEntity<Object> handleStudentNotFoundException(StudentNotFoundException
studentNotFoundException, WebRequest webRequest)
  {
     ExceptionResponse exceptionResponse=new ExceptionResponse(new
Date(),studentNotFoundException.getMessage(),webRequest.getDescription(false));
     return new ResponseEntity<>(exceptionResponse, HttpStatus.NOT_FOUND);
  }

  @ExceptionHandler(DuplicateStudentFoundException.class)
  public final ResponseEntity<Object>
handleDuplicateStudentFoundException(DuplicateStudentFoundException
duplicateStudentFoundException, WebRequest webRequest)
  {
     ExceptionResponse exceptionResponse=new ExceptionResponse(new
Date(),duplicateStudentFoundException.getMessage(),webRequest.getDescription(false));
     return new ResponseEntity<>(exceptionResponse, HttpStatus.CONFLICT);
  }

}
```

Output

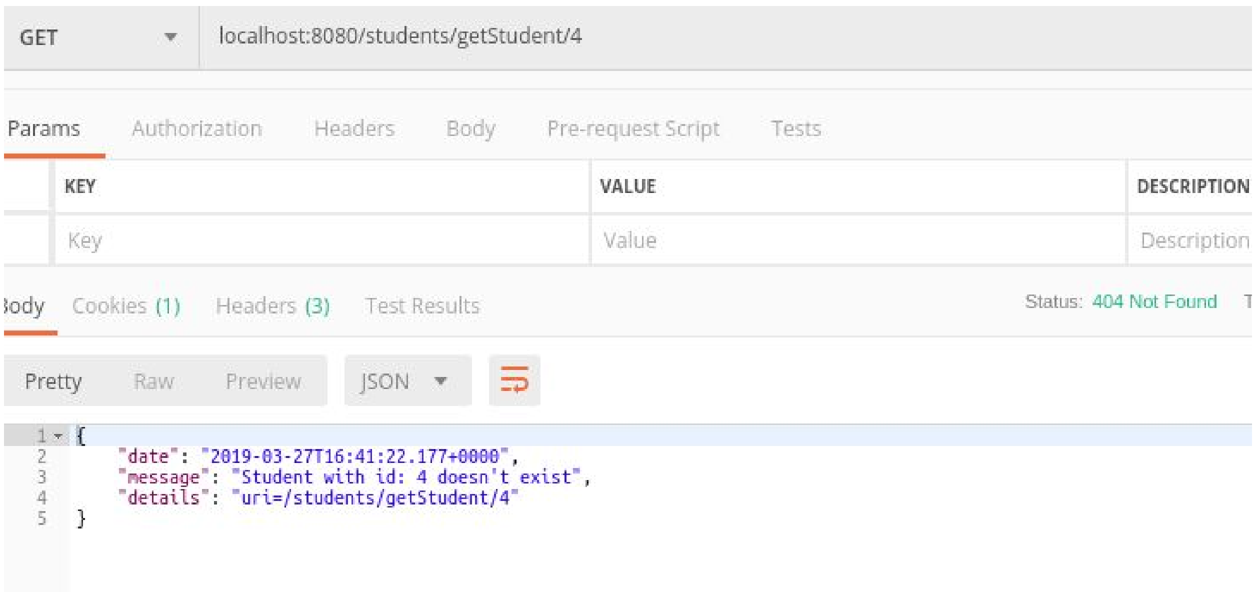


localhost:8080/students/postStudent

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

Status: 409 Conflict Time: 80 ms Size: 263 B

```
{
    "date": "2019-03-27T16:38:57.392+0000",
    "message": "Student with this name already exists",
    "details": "uri=/students/postStudent"
}
```



localhost:8080/students/getStudent/4

Status: 404 Not Found

```
{
    "date": "2019-03-27T16:41:22.177+0000",
    "message": "Student with id: 4 doesn't exist",
    "details": "uri=/students/getStudent/4"
}
```

Q3. Print the Validation Messages of student Entity in response.(1 Mark)
Solution
Change in Student.java

```java
package com.demo.restWithSpring.student;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.Valid;
import javax.validation.constraints.Size;

@Entity
public class Student {
//    Added validation for q3
   @Size(min = 4,message = "Name should contain atleast 4 characters")
   private String name;
}
```

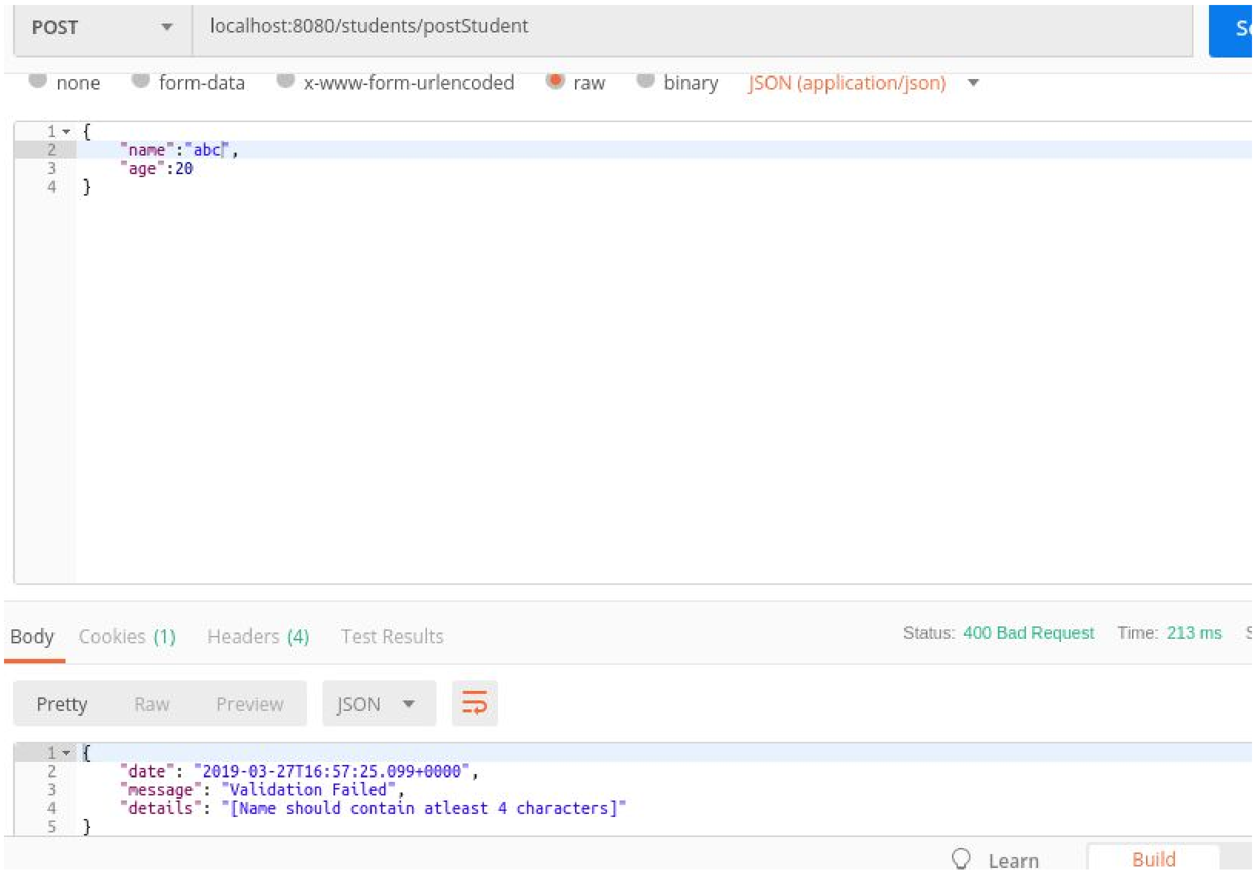
Change in StudentController.java

```java
package com.demo.restWithSpring.student.controller;

import com.demo.restWithSpring.student.Student;
import com.demo.restWithSpring.student.exception.DuplicateStudentFoundException;
import com.demo.restWithSpring.student.exception.StudentNotFoundException;
import com.demo.restWithSpring.student.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
public class StudentController {

//    Added @Valid for question3
   @PostMapping("/students/postStudent")
   public Student addStudent(@Valid @RequestBody Student student) {
       Student studentToBeSaved=studentService.getStudentByName(student.getName());
       if(!(studentToBeSaved==null))
           throw new DuplicateStudentFoundException("Student with this name already exists");
       return studentService.addStudent(student);
   }
}
```

CustomReponseEntityHandler.java

```java
package com.demo.restWithSpring.commons.exceptionhandling;

import com.demo.restWithSpring.student.exception.DuplicateStudentFoundException;
import com.demo.restWithSpring.student.exception.StudentNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.util.Date;
import java.util.List;
import java.util.stream.Collectors;

@RestController
@ControllerAdvice
public class CustomResponseEntityExceptionHandler extends ResponseEntityExceptionHandler {
  //Question3
  @Override
  protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException
ex, HttpHeaders headers, HttpStatus status, WebRequest request) {
      List<String> errorMessages = ex.getBindingResult()
              .getAllErrors()
              .stream()
              .filter(e -> e instanceof FieldError)
              .map(e -> {
                  FieldError fieldError = (FieldError) e;
                  return messageSource.getMessage(fieldError, null);
              }).collect(Collectors.toList());
      ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(), "Validation Failed",
errorMessages.toString());
      return new ResponseEntity<>(exceptionResponse, HttpStatus.BAD_REQUEST);
  }
}
```

Output



localhost:8080/students/postStudent

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

Status: 400 Bad Request Time: 213 ms

```
{
    "date": "2019-03-27T16:57:25.099+0000",
    "message": "Validation Failed",
    "details": "[Name should contain atleast 4 characters]"
}
```

Q4. Perform Internationalization for a greeting message in your app.(1 Mark)

Solution

Application.properties

```
spring.mvc.locale-resolver=accept_header
```

Message.propeties

```
greeting.message=Good Morning
```

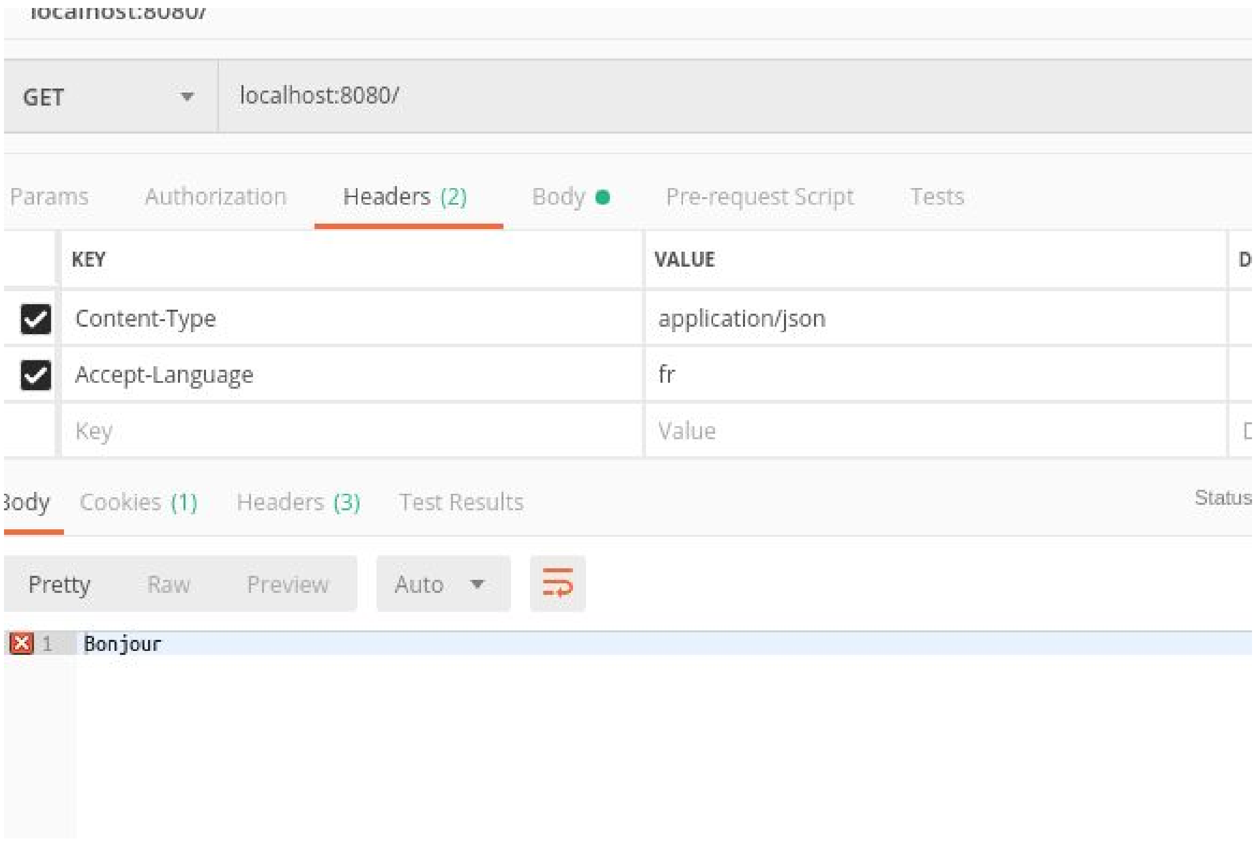
Message_fr.properties

```
greeting.message=Bonjour
```

StudentController.java

```
package com.demo.restWithSpring.student.controller;

import com.demo.restWithSpring.student.Student;
import com.demo.restWithSpring.student.exception.DuplicateStudentFoundException;
import com.demo.restWithSpring.student.exception.StudentNotFoundException;
import com.demo.restWithSpring.student.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;
import java.util.Locale;

@RestController
public class StudentController {

  @Autowired
  StudentService studentService;

  @Autowired
  MessageSource messageSource;

//forQuestion4
@GetMapping("/")
public String greet()
{
  return messageSource.getMessage("greeting.message",null, LocaleContextHolder.getLocale());
}

}
```

Output



localhost:8080/

GET localhost:8080/

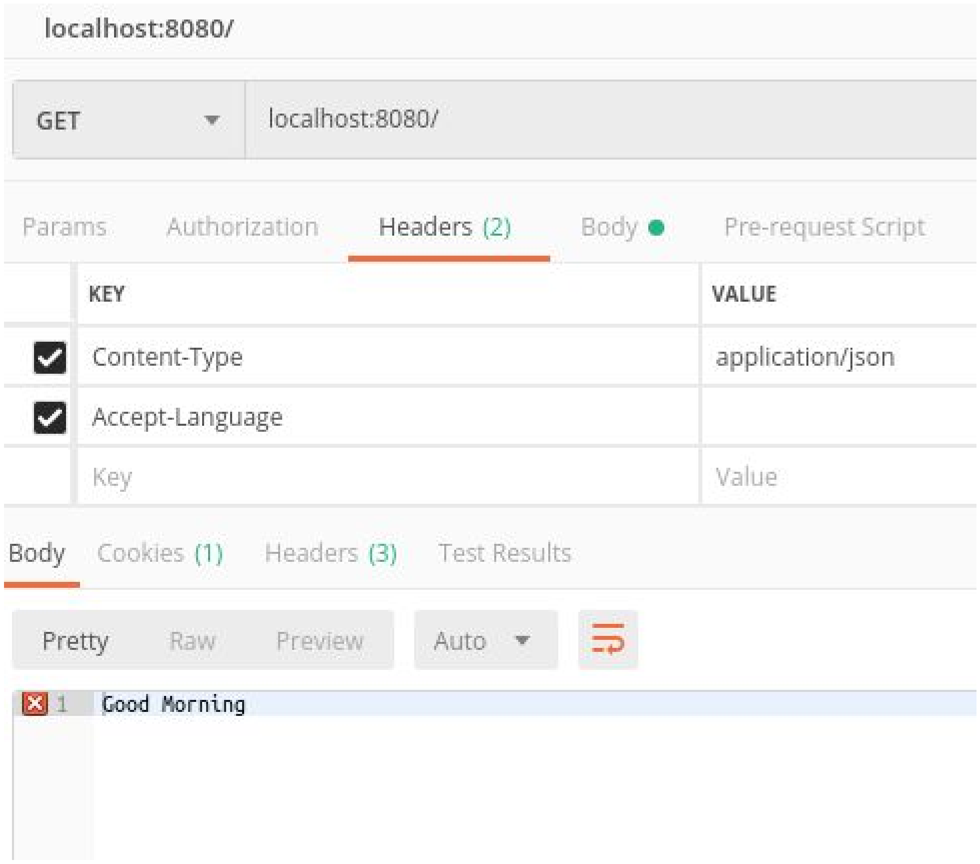
Params Authorization Headers (2) Body Pre-request Script Tests

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

Body Cookies (1) Headers (3) Test Results Status

Pretty Raw Preview Auto

```
Bonjour
```


localhost:8080/

GET localhost:8080/

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

```
Good Morning
```

Q5. Return XML Response when new Student is created.(1 Mark)

Solution

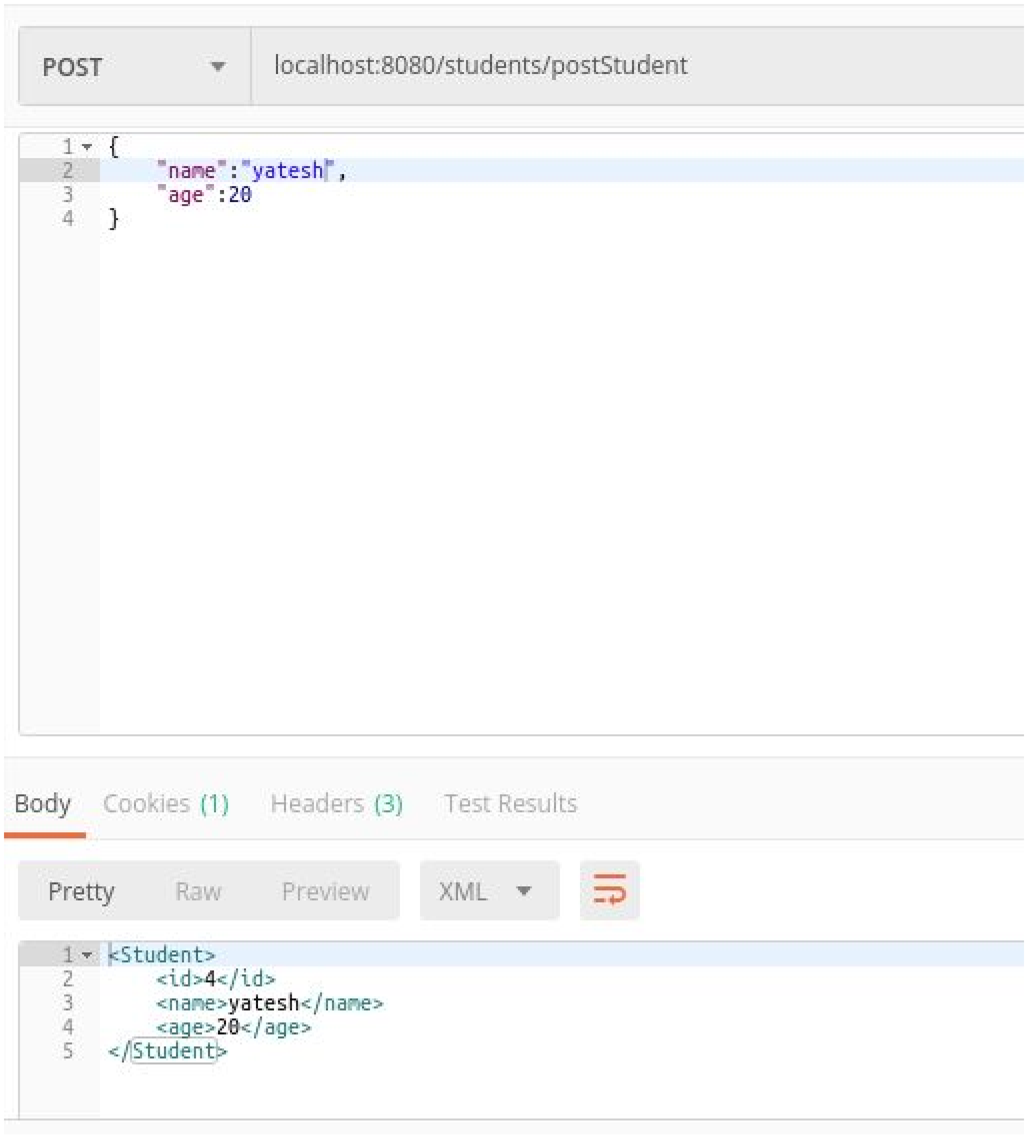
Build.gradle

```
compile ('com.fasterxml.jackson.dataformat:jackson-dataformat-xml')
```

StudentController.java

```java
package com.demo.restWithSpring.student.controller;

import com.demo.restWithSpring.student.Student;
import com.demo.restWithSpring.student.exception.DuplicateStudentFoundException;
import com.demo.restWithSpring.student.exception.StudentNotFoundException;
import com.demo.restWithSpring.student.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;
import java.util.Locale;

@RestController
public class StudentController {

  @Autowired
  StudentService studentService;

  @Autowired
  MessageSource messageSource;

//    Added @Valid for question3
//Added produces for question5
  @PostMapping(value = "/students/postStudent",produces = "application/xml")
  public Student addStudent(@Valid @RequestBody Student student) {
      Student studentToBeSaved=studentService.getStudentByName(student.getName());
      if(!(studentToBeSaved==null))
          throw new DuplicateStudentFoundException("Student with this name already exists");
      return studentService.addStudent(student);
  }

}
```

Output



localhost:8080/students/postStudent

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

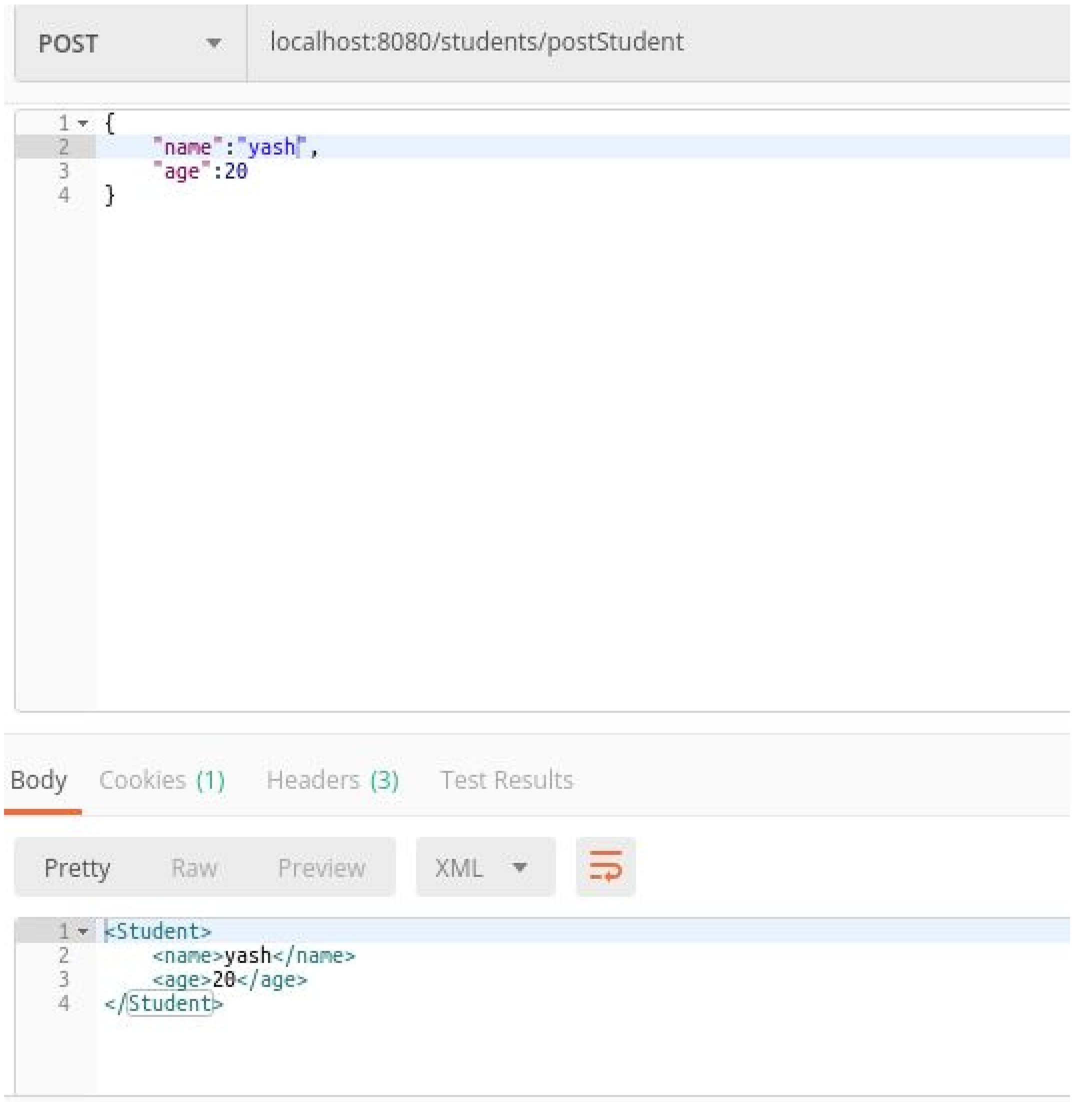
```
<Student>
    <id>4</id>
    <name>yatesh</name>
    <age>20</age>
</Student>
```

Q6. Ignore ID field in the Response.(1 Mark)

Student.java

```java
package com.demo.restWithSpring.student;

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.Valid;
import javax.validation.constraints.Size;

@Entity
public class Student {
  //Added @JsonIgnore for Question6
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @JsonIgnore
   private Long id;

}
```

Output


POST localhost:8080/students/postStudent

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

```
<Student>
    <name>yash</name>
    <age>20</age>
</Student>
```

Q7. Create 2 versions of your API one take represent name of the Student as single string and other showing firstname and lastname separately. (Create the Versions of the API using URI, parameter and header versioning)

Solution

StudentV1.java

```java
package com.demo.restWithSpring.student.versioning;



public class StudentV1 {
  private String name;

  public StudentV1(String name) {
     this.name = name;
  }

  public String getName() {
     return name;
  }

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

StudentV2.java

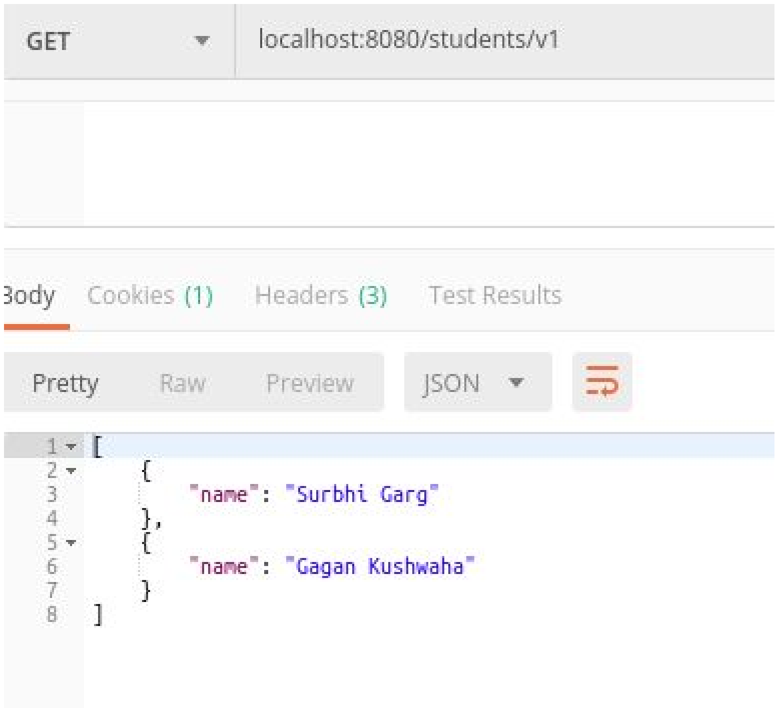
```java
package com.demo.restWithSpring.student.versioning;

public class StudentV2 {
  private String firstName;
  private String lastName;

  public StudentV2(String firstName, String lastName) {
     this.firstName = firstName;
     this.lastName = lastName;
  }

  public String getFirstName() {
     return firstName;
  }

  public void setFirstName(String firstName) {
     this.firstName = firstName;
  }

  public String getLastName() {
     return lastName;
  }

  public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
```

StudentVersioningController.java

```
package com.demo.restWithSpring.student.controller;

import com.demo.restWithSpring.student.versioning.StudentV1;
import com.demo.restWithSpring.student.versioning.StudentV2;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.Arrays;
import java.util.List;
@RequestMapping("/students")
@RestController
public class StudentVersioningController {
    //using uri
    @GetMapping("/v1")
    public List<StudentV1> getStudents()
    {
        return Arrays.asList(new StudentV1("Surbhi Garg"),new StudentV1("Gagan Kushwaha"));
    }
    @GetMapping("/v2")
    public List<StudentV2> getVersion2Students()
    {
        return Arrays.asList(new StudentV2("Surbhi","Garg"),new StudentV2("Rama","Garg"));
    }

    //Parameter Versioning
    @GetMapping(value = "/param",params = "version=1")
    public List<StudentV1> getStudentsV1ByParam() {
        return Arrays.asList(new StudentV1("Surbhi Garg"),new StudentV1("Gagan Kushwaha"));
    }

    @GetMapping(value = "/param",params = "version=2")
    public List<StudentV2> getStudentsV2ByParam() {
       return Arrays.asList(new StudentV2("Surbhi","Garg"),new StudentV2("Rama","Garg"));
    }

    //Header Versioning
    @GetMapping(value = "/header",headers = "API-VERSION=1")
    public List<StudentV1> getPersonHeader1() {
        return Arrays.asList(new StudentV1("Surbhi Garg"),new StudentV1("Gagan Kushwaha"));
    }

    @GetMapping(value = "/header",headers = "API-VERSION=2")
    public List<StudentV2>getPersonHeader2() {
        return Arrays.asList(new StudentV2("Surbhi","Garg"),new StudentV2("Rama","Garg"));
    }
```

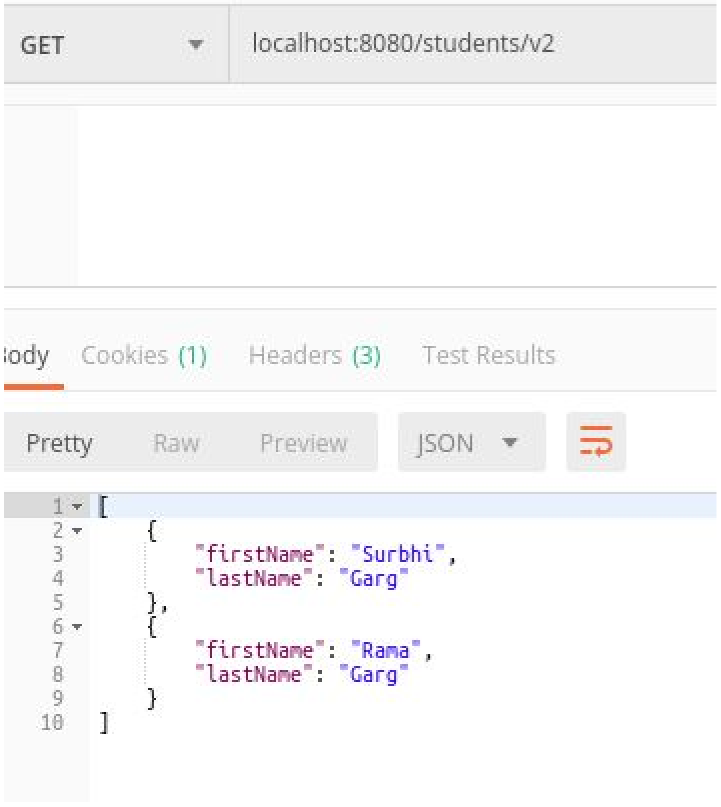
}

Output



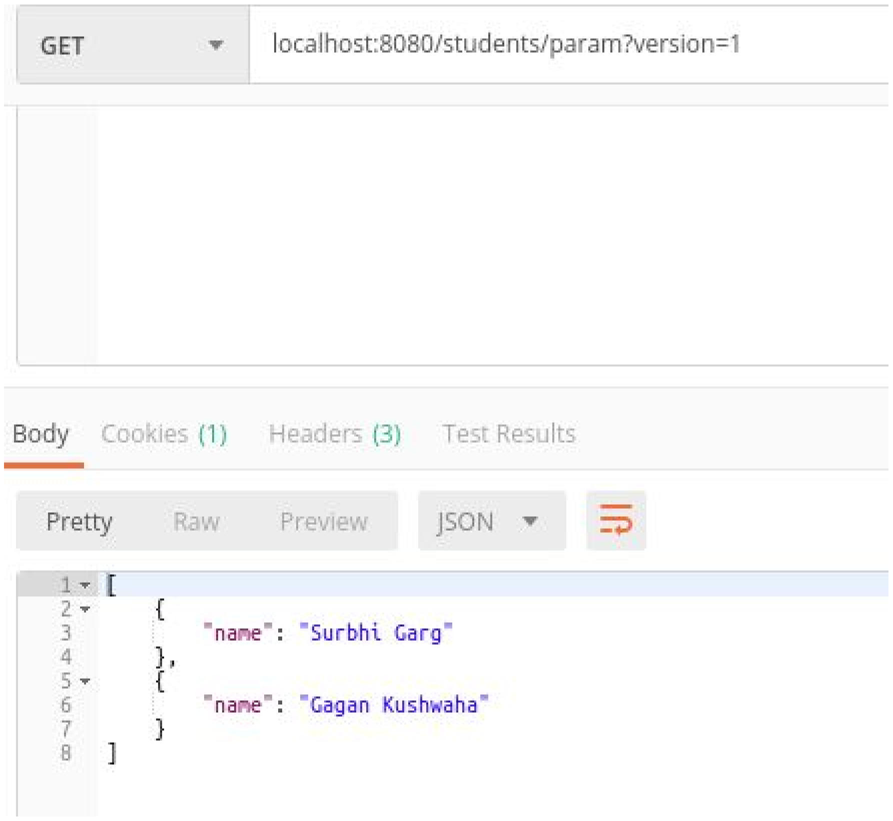
localhost:8080/students/v1

```
[
    {
        "name": "Surbhi Garg"
    },
    {
        "name": "Gagan Kushwaha"
    }
]
```



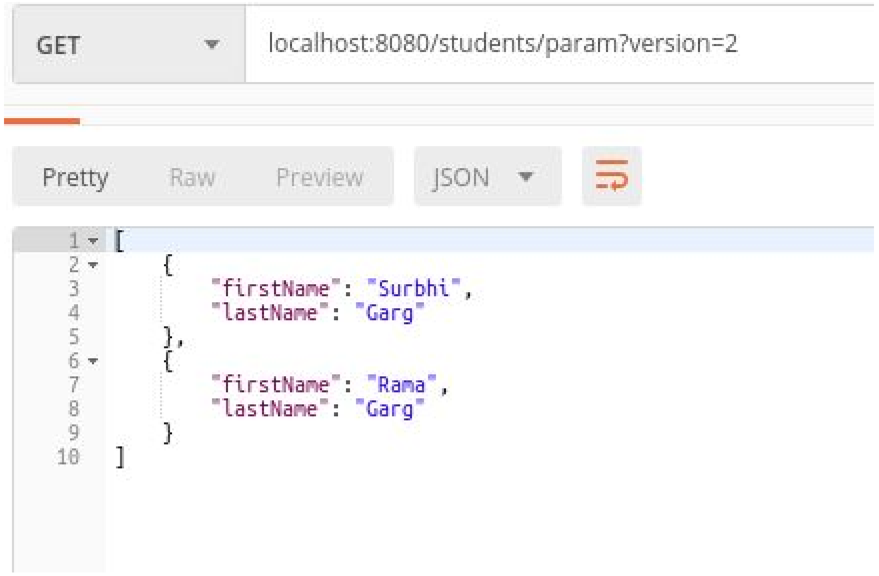
localhost:8080/students/v2

```
[
    {
        "firstName": "Surbhi",
        "lastName": "Garg"
    },
    {
        "firstName": "Rama",
        "lastName": "Garg"
    }
]
```



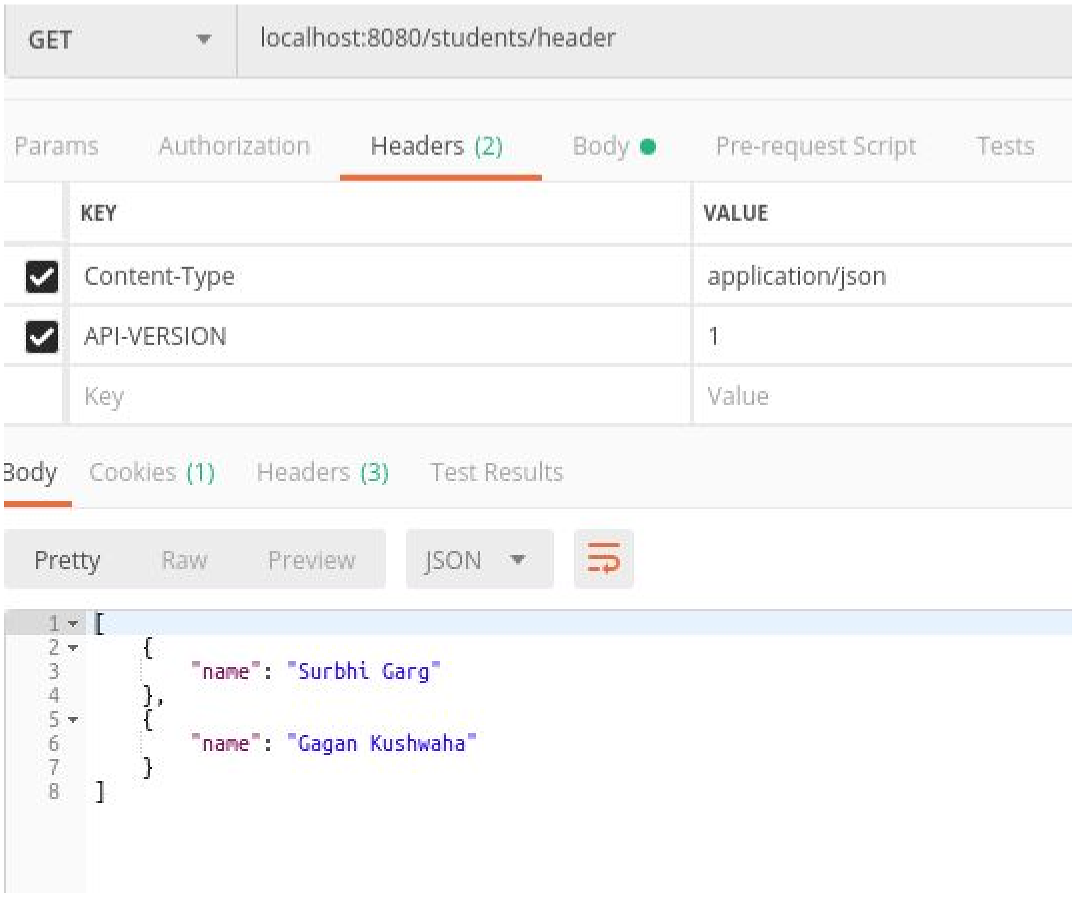
localhost:8080/students/param?version=1

```
[
    {
        "name": "Surbhi Garg"
    },
    {
        "name": "Gagan Kushwaha"
    }
]
```



localhost:8080/students/param?version=2

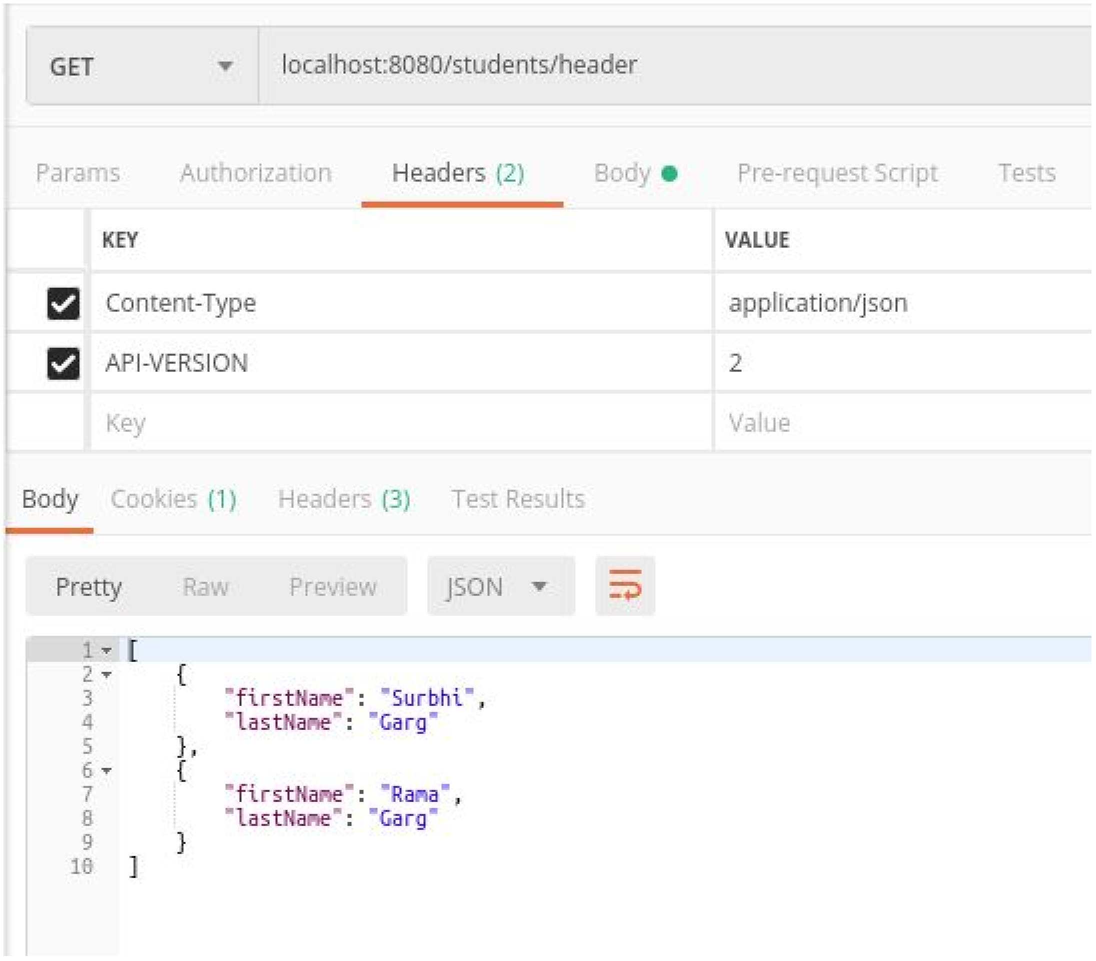
```
[
    {
        "firstName": "Surbhi",
        "lastName": "Garg"
    },
    {
        "firstName": "Rama",
        "lastName": "Garg"
    }
]
```



localhost:8080/students/header

KEY	VALUE
✓ Content-Type	application/json
✓ API-VERSION	1
Key	Value

```
[
    {
        "name": "Surbhi Garg"
    },
    {
        "name": "Gagan Kushwaha"
    }
]
```



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

```
[
    {
        "firstName": "Surbhi",
        "lastName": "Garg"
    },
    {
        "firstName": "Rama",
        "lastName": "Garg"
    }
]
```

Q8. Perform CRUD operations on the resource below using   RestTemplate

Solution

Post.java

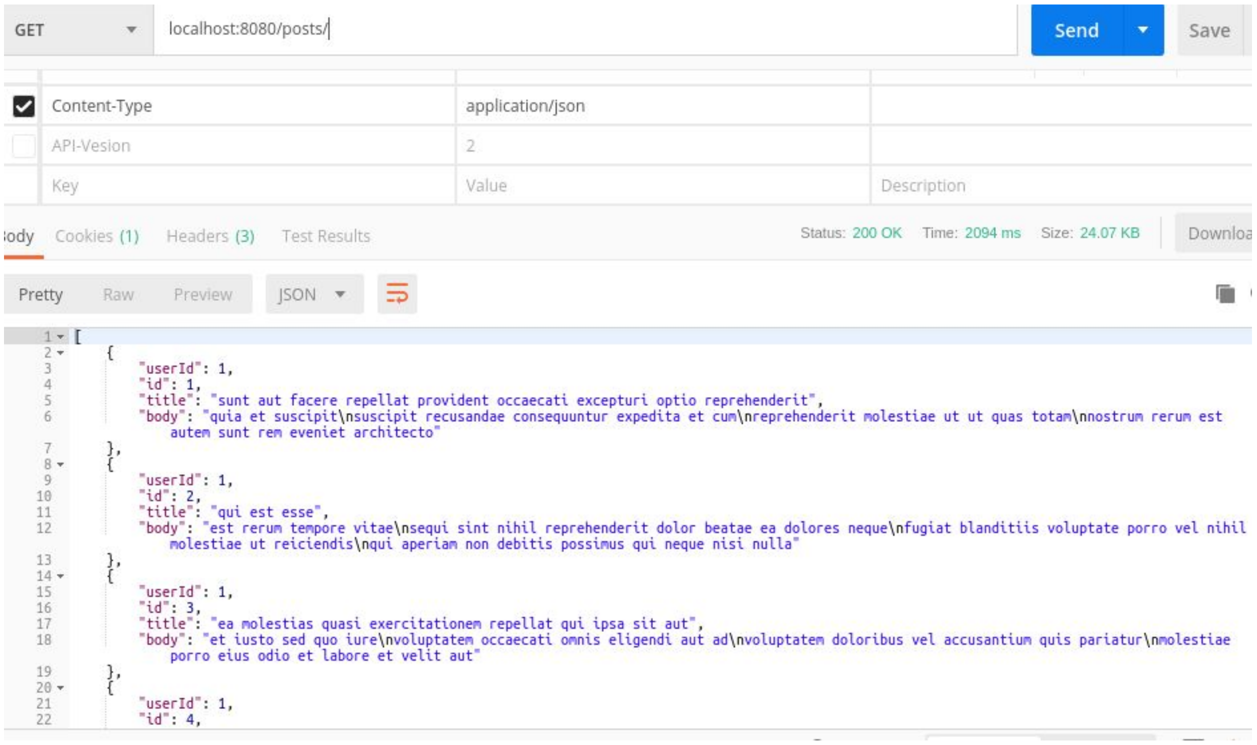
```
package com.demo.restWithSpring.post;

public class Post {
  Long userId;
  Long id;
  String title;
  String body;
  public Post(){}
  public Post(Long userId, Long id, String title, String body) {
      this.userId = userId;
      this.id = id;
      this.title = title;
      this.body = body;
  }

  public Long getUserId() {
      return userId;
  }

  public void setUserId(Long userId) {
      this.userId = userId;
  }

  public Long getId() {
      return id;
  }

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

  public String getTitle() {
      return title;
  }

  public void setTitle(String title) {
      this.title = title;
  }

  public String getBody() {
      return body;
  }

  public void setBody(String body) {
      this.body = body;
  }
}
```

PostController.java

```
package com.demo.restWithSpring.post.controller;

import com.demo.restWithSpring.post.Post;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

import java.lang.reflect.ParameterizedType;
import java.util.List;

@RestController
@RequestMapping("/posts")
public class PostController {
  @GetMapping("/")
  public List<Post> getPostsList() {
      RestTemplate restTemplate = new RestTemplate();
      String uri = "https://jsonplaceholder.typicode.com/posts";
      ResponseEntity<List<Post>> responseEntity = restTemplate.exchange(uri, HttpMethod.GET, null, new
ParameterizedTypeReference<List<Post>>() {
      });
      return responseEntity.getBody();
  }
  @GetMapping("/{id}")
  public Post getPost(@PathVariable Long id)
  {
      RestTemplate restTemplate=new RestTemplate();
      String uri = "https://jsonplaceholder.typicode.com/posts/"+id;
      ResponseEntity<Post>responseEntity=restTemplate.getForEntity(uri,Post.class);
      return responseEntity.getBody();

  }
  @PostMapping("/createPost")
  public ResponseEntity<Post> createPost(@RequestBody Post post)
  {
      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<>(post,httpHeaders);
      Post postCreated=restTemplate.postForObject(url,request,Post.class);
      System.out.println(post);
      return new ResponseEntity<Post>(postCreated, HttpStatus.CREATED);
  }
  @PutMapping("/updatePost/{id}")
  public ResponseEntity<Post> updatePost(@PathVariable Long id){
      String url="https://jsonplaceholder.typicode.com/posts/"+id;
      RestTemplate restTemplate= new RestTemplate();
      HttpHeaders httpHeaders= new HttpHeaders();
```

```
        httpHeaders.add("Content-type","application/json; charset=UTF-8");
        HttpEntity<Post> request=new HttpEntity<>(new Post(id,100L,"title1","description1"),httpHeaders);
        return restTemplate.exchange(url,HttpMethod.PUT,request,Post.class);
    }

    @DeleteMapping("/deletePost/{id}")
    public void deletePost(@PathVariable Long id){
        String url="https://jsonplaceholder.typicode.com/posts/"+id;
        RestTemplate restTemplate= new RestTemplate();
        HttpHeaders httpHeaders= new HttpHeaders();
        httpHeaders.add("Content-type","application/json; charset=UTF-8");
        HttpEntity<Post> request=new HttpEntity<>(httpHeaders);
        restTemplate.exchange(url,HttpMethod.DELETE,request,Post.class);
    }

}
```

Output

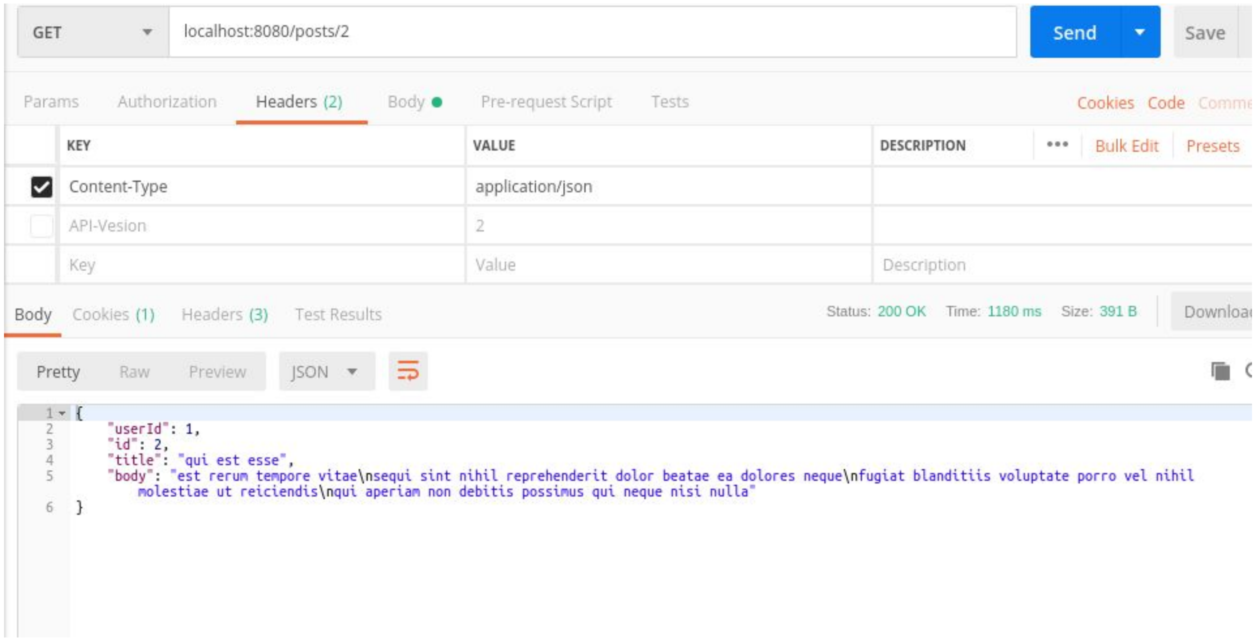


GET localhost:8080/posts/

	Key	Value	Description
✓	Content-Type	application/json	
	API-Vesion	2	

Status: 200 OK Time: 2094 ms Size: 24.07 KB

```
[
    {
        "userId": 1,
        "id": 1,
        "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
        "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est
            autem sunt rem eveniet architecto"
    },
    {
        "userId": 1,
        "id": 2,
        "title": "qui est esse",
        "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil
            molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
    },
    {
        "userId": 1,
        "id": 3,
        "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
        "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae
            porro eius odio et labore et velit aut"
    },
    {
        "userId": 1,
        "id": 4,
```

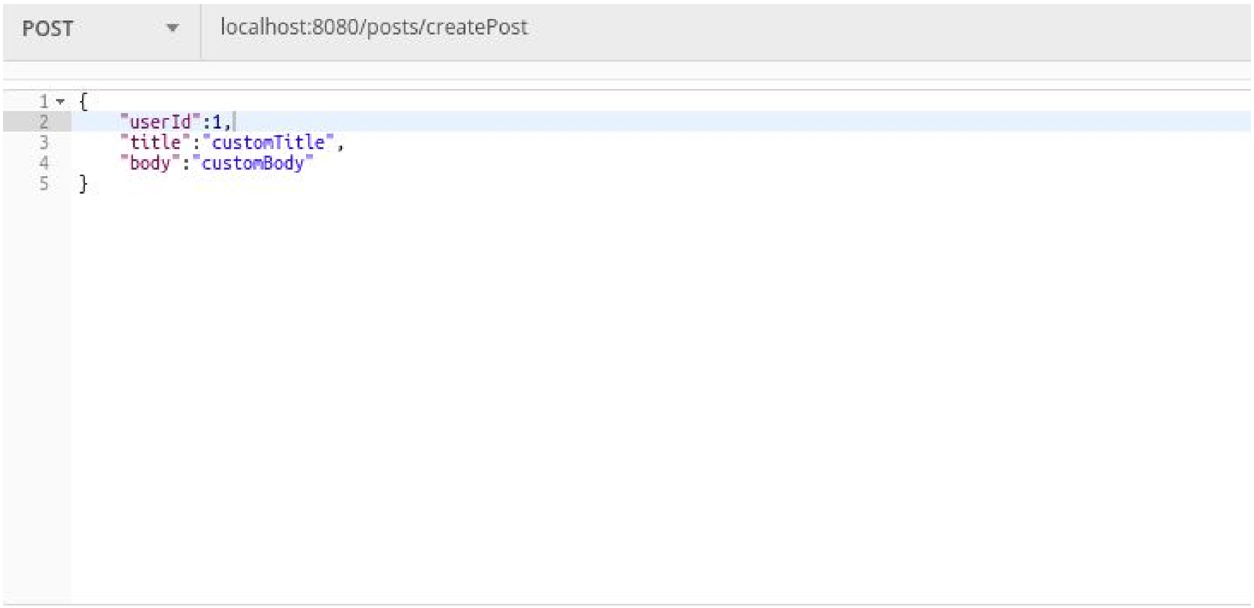


localhost:8080/posts/2

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

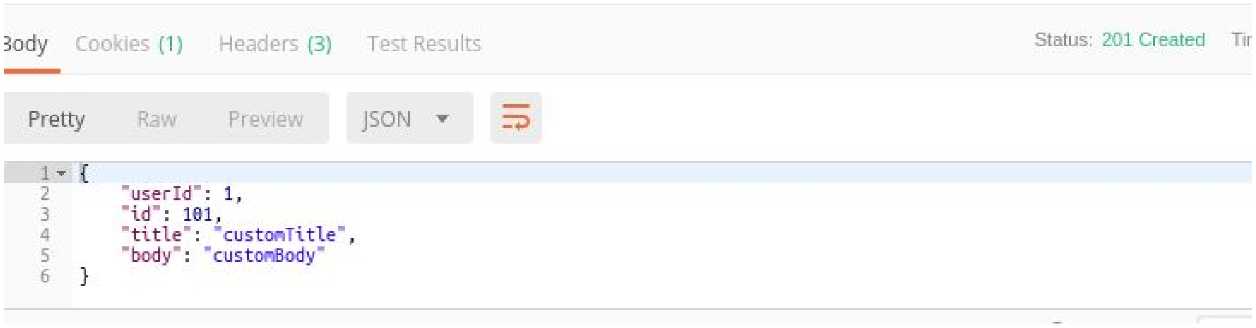
Status: 200 OK Time: 1180 ms Size: 391 B

```
{
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil
        molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
}
```



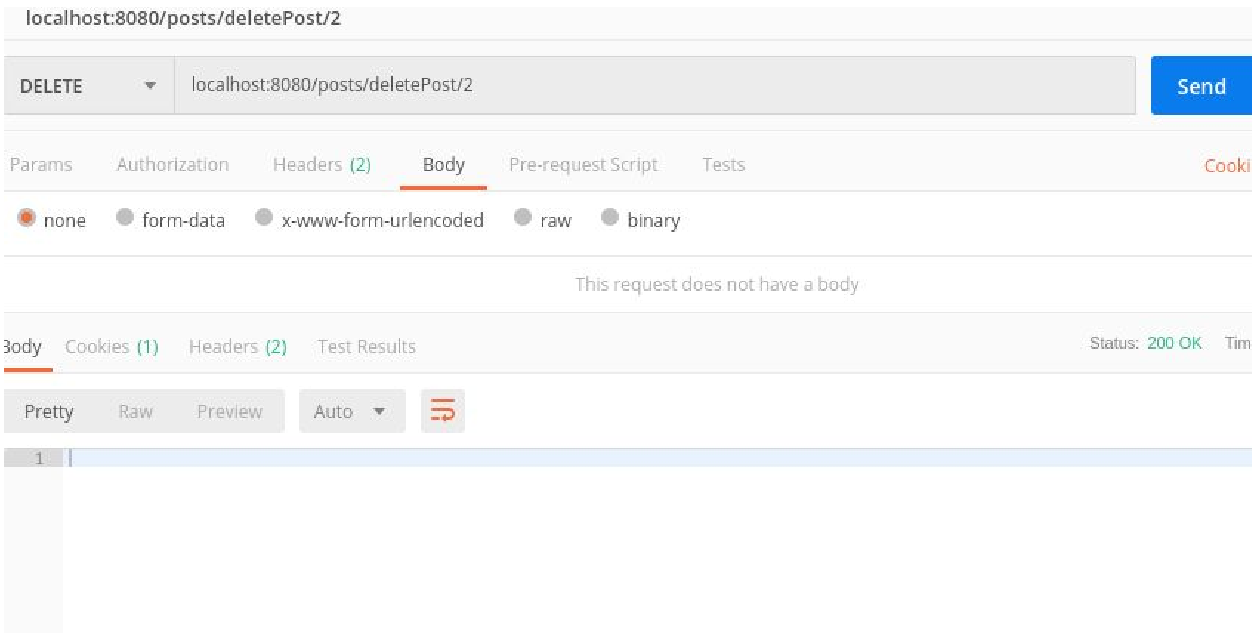
localhost:8080/posts/createPost

```
{
    "userId":1,
    "title":"customTitle",
    "body":"customBody"
}
```



Status: 201 Created

```
{
    "userId": 1,
    "id": 101,
    "title": "customTitle",
    "body": "customBody"
}
```

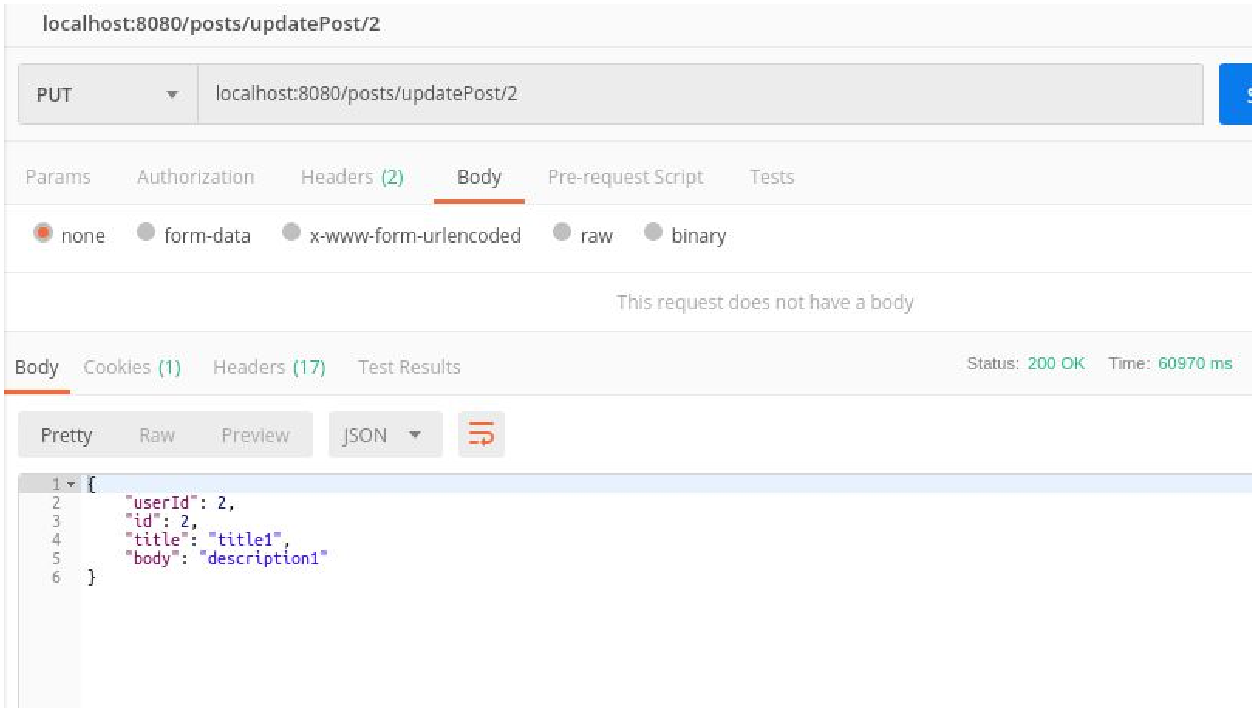


DELETE localhost:8080/posts/deletePost/2

[✓ none] form-data x-www-form-urlencoded raw binary

This request does not have a body

Status: 200 OK



PUT localhost:8080/posts/updatePost/2

[✓ none] form-data x-www-form-urlencoded raw binary

This request does not have a body

Status: 200 OK Time: 60970 ms

```
{
    "userId": 2,
    "id": 2,
    "title": "title1",
    "body": "description1"
}
```