UNIVERSITY OF SRI JAYEWARDENEPURA

Faculty of Technology

Department of Information and Communication Technology

# ITS 4243 - Microservices and Cloud Computing

Assignment 01

Name: A.H.M.M. SILVA

Index No: ICT/21/922

## Part 01:

### 1. What is Spring Boot and why is it used?

- Spring Boot: It is an open-source java-based framework built on top of the **Spring Framework** that simplifies Java application development.
  It provides pre-configured settings, embedded servers (like Tomcat, Jetty or Undertow), and production-ready features to help developers build and run applications quickly without complex XML configurations.

- Why it's used:

  1. **Fast Development:** Quickly build production ready apps with minimal configuration.
  2. **Auto-Configuration:** Reduces boilerplate code, you can focus on business logic.
  3. **Microservices Friendly:** Perfect for building and managing distributed systems.
  4. **Seamless Integrations:** Works easily with databases, security, messaging and cloud platforms.
  5. **Wide Adoption:** Backed by Spring ecosystem and used in real world enterprise applications.

### 2. Explain the difference between Spring Framework and Spring Boot.

| Spring Framework | Spring Boot |
|---|---|
| Core framework providing comprehensive infrastructure support | Framework built on top of Spring Framework |
| Requires explicit configuration (XML, Java config) | Auto-configuration reduces manual setup |
| No embedded server by default | Incudes embedded servers (Tomcat and Jetty etc.) |
| To create a Spring application, the developers write lots of code. | It reduces the lines of code. |
| Doesn't provide support for the in-memory | Provides support for the in-memory database |

| | |
|---|---|
| database. | such as H2. |
| Developers must define dependencies manually in the pom.xml file. | pom.xml file internally handles the required dependencies. |

**3. What is Inversion of Control (IoC) and Dependency Injection (DI)?**

- Inversion of Control (IOC): It is a design principle where the control of object creation and lifecycle is managed by a framework or container rather than by the developer. Spring IOC Container is responsible for creating, configuring, and managing the lifecycle of objects called beans.
  *Spring IoC is achieved through Dependency Injection.*

- Dependency Injection: It is a design pattern and a part of IOC container. It allows objects to be injected with their dependencies rather than creating those dependencies themselves.

**4. What is the purpose of application.properties / application.yml?**

- These files are used to configure application settings in Spring Boot.

  Define parameters like:

  - Server port (server.port=8081)
  - Database credentials (spring.datasource.url=...)
  - Logging levels, file paths, etc.

5. **Explain what a REST API is and list HTTP methods used.**

- REST (Representational State Transfer) APIs enable you to develop all kinds of web applications having all possible CRUD (create, retrieve, update, delete) operations.

  Core Principles:
  - Uses standard HTTP methods.
  - Resources identified by URIs.
  - Stateless, cacheable, uniform interface.

- HTTP Methods:

| Method | Purpose |
|---|---|
| GET | Retrieve data from the server |
| POST | Create new data/resource |
| PUT | Update existing data completely |
| PATCH | Update partially |
| DELETE | Remove data/resource |

6. **What is Spring Data JPA? What is an Entity and a Repository?**

- Spring Data JPA: Spring Data JPA, part of the larger Spring Data family, makes it easy to easily implement JPA-based (Java Persistence API) repositories. It makes it easier to build Spring-powered applications that use data access technologies.
  Spring Data JPA aims to significantly improve the implementation of data access layers by reducing the effort to the amount that's actually needed (Reduce boilerplate codes).

- Entity: A class that represents a database table. Each instance corresponds to a row. Annotated with @Entity.

- Repository: An interface that handles data access operations (CRUD) for entities. Extends JpaRepository or CrudRepository.

  Ex:

```
@Entity
class Student {
   @Id
    private Long id;
    private String name;
    private String email;
    private String course;
}

@Repository
    interface StudentRepository extends JpaRepository<Student, Long> {


    }
```

7. **What is the difference between @Component, @Service, @Repository, @Controller, @RestController?**

| **@Component Annotation** | **@Service Annotation** | **@Repository Annotation** | **@Controller Annotation** | **@RestController Annotation** |
|---|---|---|---|---|
| @Component is a general-purpose Spring annotation used to make any class a Spring-managed bean. | @Service is used with classes that contain business logic or main service functions. | @Repository is used with classes that handle database operations like save, update, delete, or find. | @Controller is used with classes that handle web page requests and return HTML views. | @RestController is used with classes that handle REST API requests and return JSON or XML responses. |
| It is the base (parent) stereotype annotation. | It is a special type of @Component. | It is a special type of @Component. | It is a special type of @Component. | It is a special type of @Component. |
| Used to mark a generic Spring bean that doesn't fit into service, repository, or controller categories. | Used to mark a class as a service provider. | Used to mark a class as a DAO (Data Access Object) provider. | Used to mark a class as a web request handler (for web apps). | Used to mark a class as a REST web request handler (for APIs). |
| It is a stereotype for general beans. | It is a stereotype for the service layer. | It is a stereotype for the data access layer. | It is a stereotype for the presentation (MVC) layer. | It is a stereotype for the REST API layer. |

| | | | | |
|---|---|---|---|---|
| Can be used anywhere as a common bean, no restrictions. | Can be replaced by @Component (not recommended). | Can be replaced by @Component (not recommended). | Cannot be switched with @Service or @Repository. | Cannot be switched with other annotations. |
| It is a Stereotype Annotation. | It is a Stereotype Annotation. | It is a Stereotype Annotation. | It is a Stereotype Annotation. | It is a Stereotype Annotation. |

- *Stereotype annotations are special annotations in Spring used to auto-detect and register beans in the application context.*
- *A bean:  Is a Java object that is instantiated, configured, and managed by the Spring IoC (Inversion of Control) container. Essentially, beans are the fundamental building blocks of a Spring application.*

**8. What is @Autowired? When should we avoid it?**

- @Autowired is used by Spring to automatically inject dependencies into beans (IoC/DI concept).
- Avoid it when:
  - Using field injection (harder to test and maintain). Prefer constructor injection.
  - In configuration classes where explicit bean wiring is clearer.
  - When there are multiple bean candidates (use @Qualifier instead).

**9. Explain how Exception Handling works in Spring Boot (@ControllerAdvice).**

- @ControllerAdvice is a global exception handler that catches exceptions across multiple controllers. It handle exceptions from multiple controllers in a centralized way.
- Don’t have to write the same try–catch code repeatedly.

EX:

- Imagine you are building a Student Management System.You have a controller that searches for a student by ID.

```
@RestController
public class StudentController {

  @GetMapping("/students/{id}")
  public String getStudent(@PathVariable int id) {
    if (id != 1) { // Only student with ID=1 exists
      throw new ResourceNotFoundException("Student not found!");
    }
    return "Student found!";
  }
}
```

- If you try to open: http://localhost:8080/students/2, You will get an error (because student ID 2 doesn’t exist).
- Without @ContollerAdvice
  - You would need to add try-catch in every controller.
- With @ ContollerAdvice
  - Write one global class to handle all such errors.

```
@ControllerAdvice
public class GlobalExceptionHandler {

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

- Now, whenever any controller throws a ResourceNotFoundException, this handler will automatically catch it and return a proper message.

- *@ControllerAdvice - Tells Spring this class will handle exceptions globally.*
- *@ExceptionHandler - Tells which type of exception to catch.*
- *ResponseEntity - Lets send a custom error message and HTTP status code.*

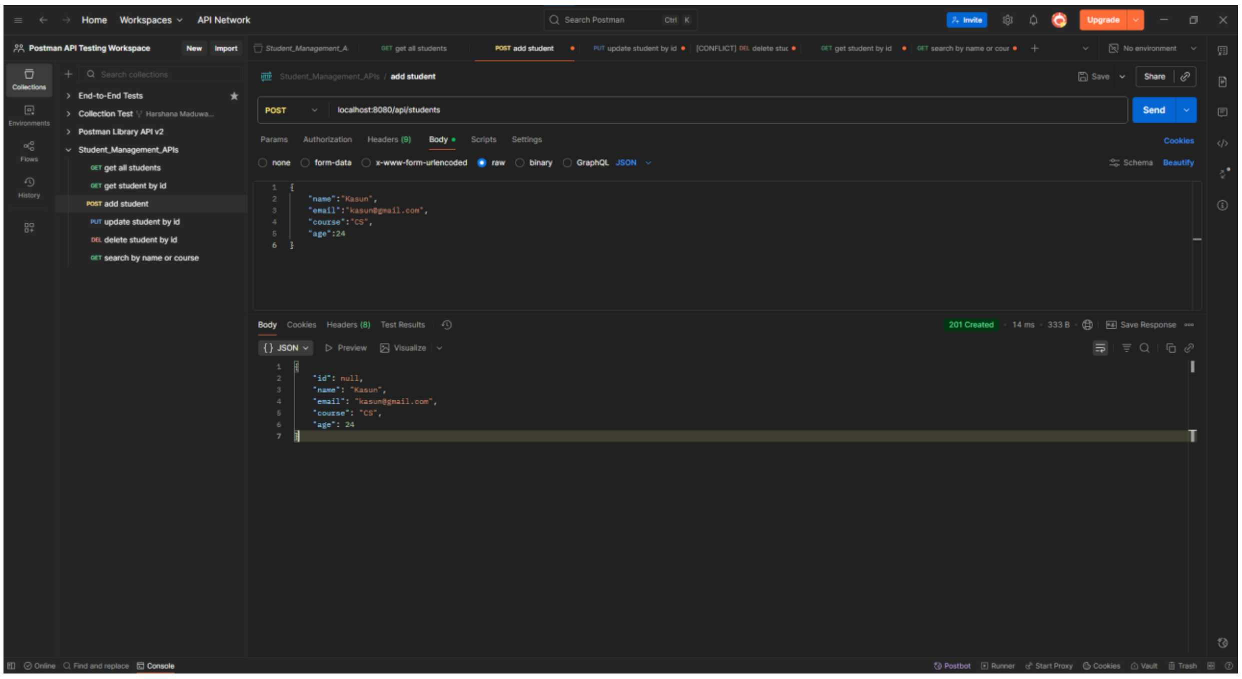
**10. What is the role of Maven/Gradle in a Spring Boot project?**

- Both Maven and Gradle are build automation tools used to manage dependencies, build, and package Spring Boot projects.

    - Manage project dependencies via pom.xml (Maven) or build.gradle (Gradle).
    - Compile source code, run tests, and package the app (.jar or .war).
    - Simplify project builds with a single command (mvn clean install, gradle build).

## Part 02:

### Add Student



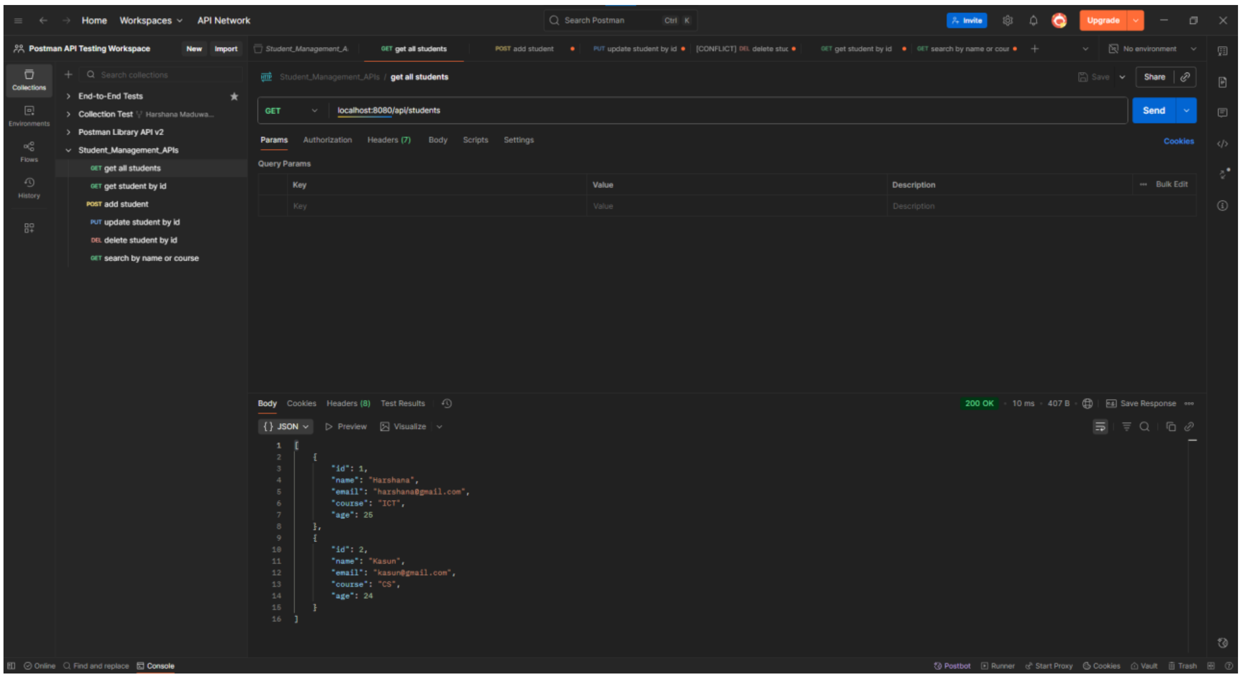
POST localhost:8080/api/students

```
{
    "name":"Kasun",
    "email":"kasun@gmail.com",
    "course":"CS",
    "age":24
}
```

201 Created · 14 ms · 333 B

```
{
    "id": null,
    "name": "Kasun",
    "email": "kasun@gmail.com",
    "course": "CS",
    "age": 24
}
```

### Get All Students

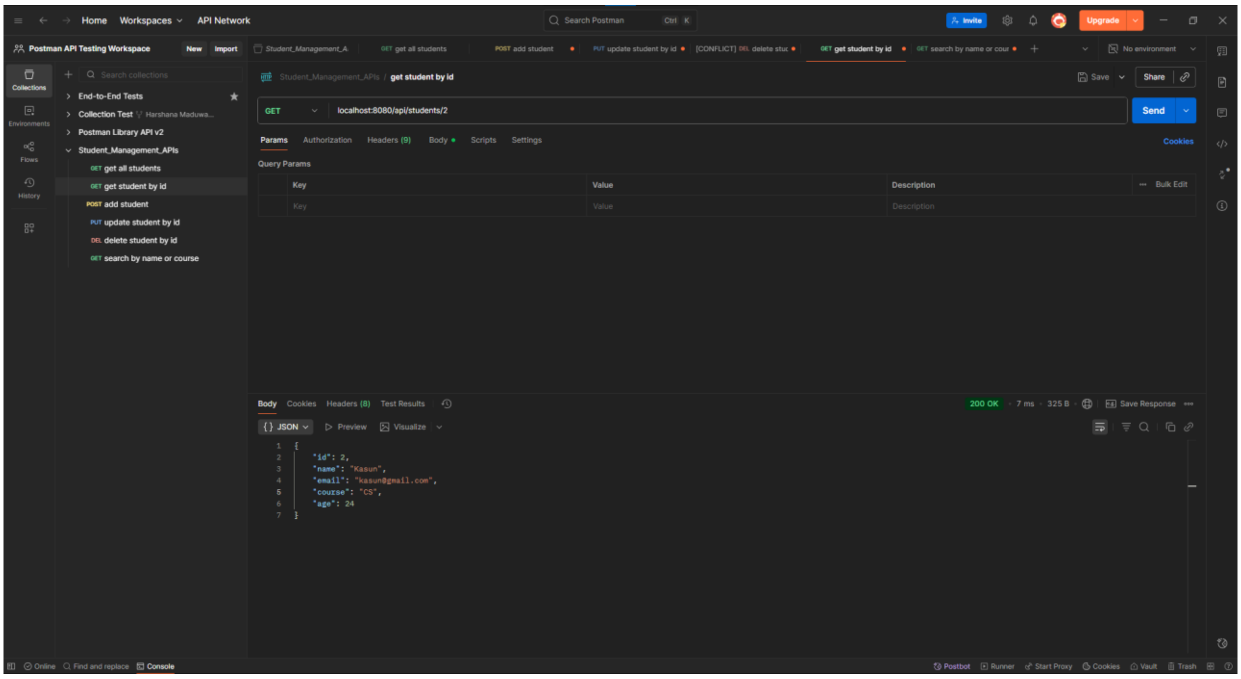


GET localhost:8080/api/students

200 OK · 10 ms · 407 B

```
[
    {
        "id": 1,
        "name": "Harshana",
        "email": "harshana@gmail.com",
        "course": "ICT",
        "age": 25
    },
    {
        "id": 2,
        "name": "Kasun",
        "email": "kasun@gmail.com",
        "course": "CS",
        "age": 24
    }
]
```

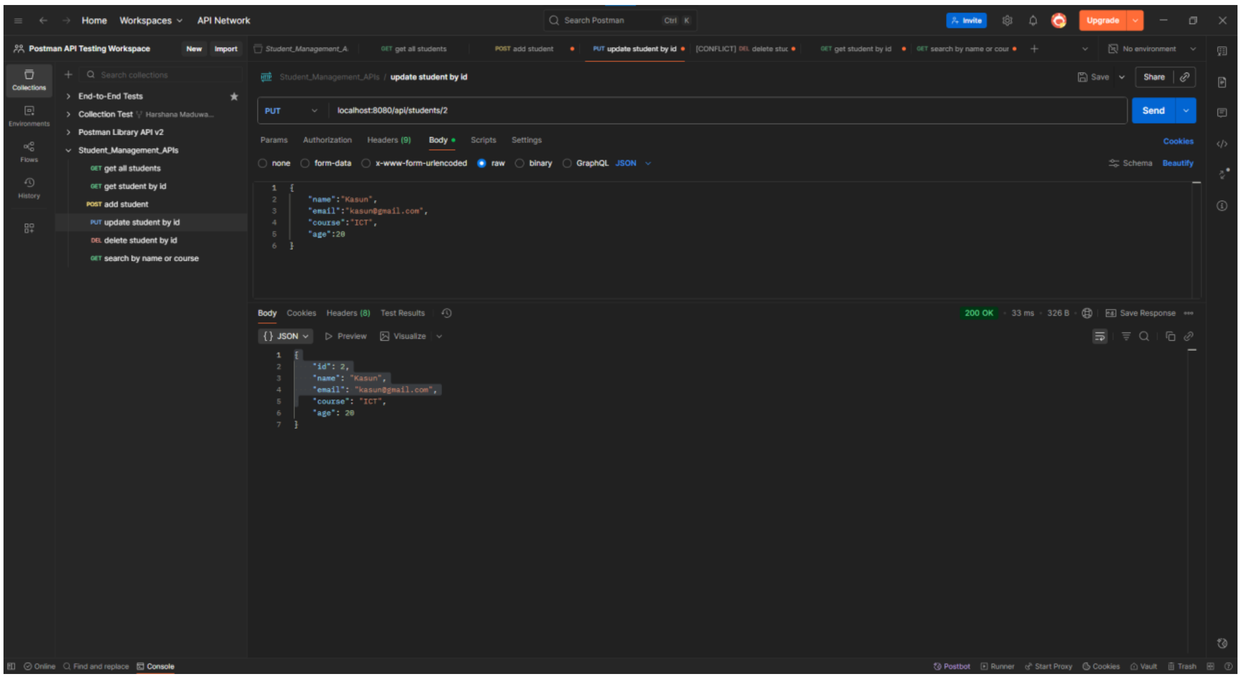
Get a student by Id


```
GET  localhost:8080/api/students/2
```

200 OK · 7 ms · 325 B

```
{
    "id": 2,
    "name": "Kasun",
    "email": "kasun@gmail.com",
    "course": "CS",
    "age": 24
}
```

Update a student by id


```
PUT  localhost:8080/api/students/2
```

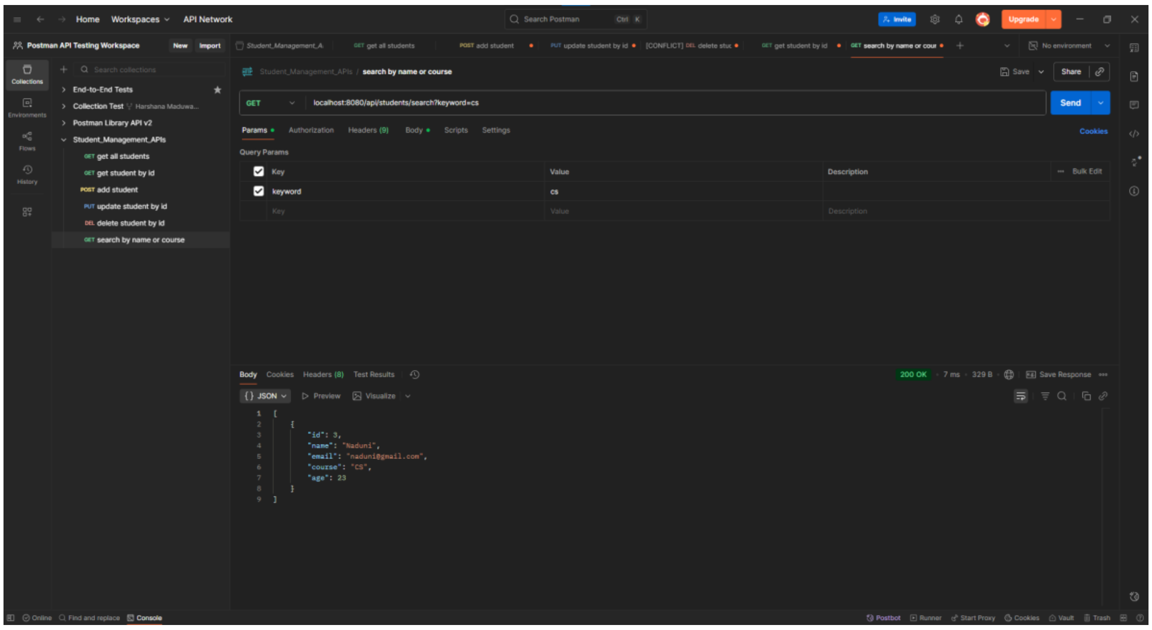
```
{
    "name":"Kasun",
    "email":"kasun@gmail.com",
    "course":"ICT",
    "age":20
}
```

200 OK · 33 ms · 326 B

```
{
    "id": 2,
    "name": "Kasun",
    "email": "kasun@gmail.com",
    "course": "ICT",
    "age": 20
}
```

Search by name or course

By course:



Student_Management_APIs / search by name or course

GET localhost:8080/api/students/search?keyword=cs

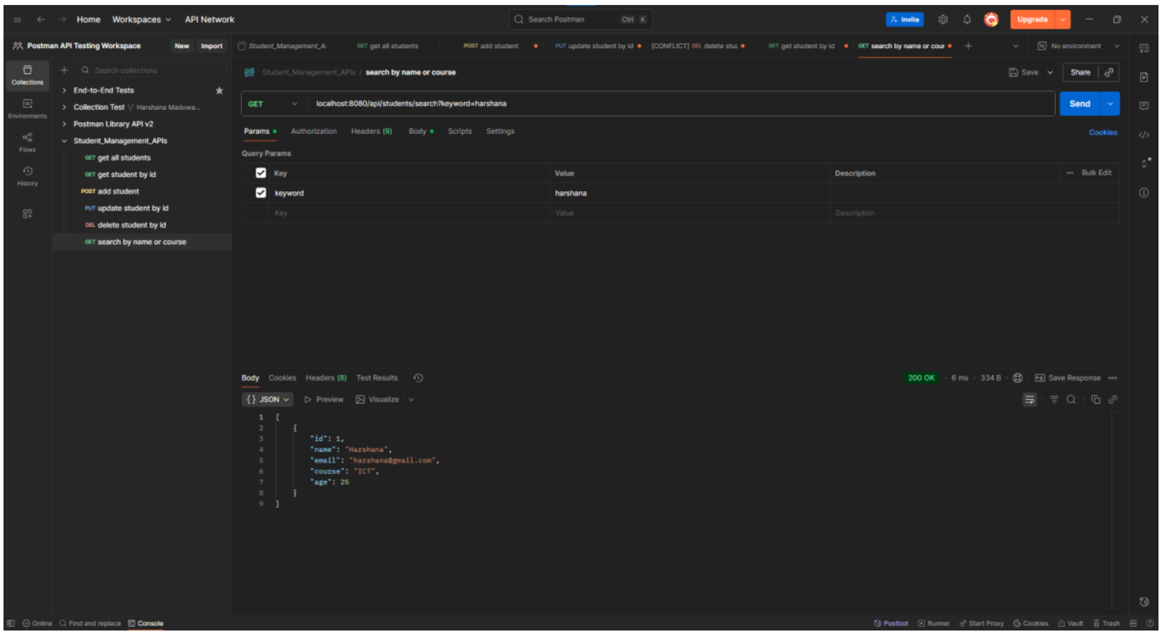
Query Params

| Key | Value | Description |
|---|---|---|
| keyword | cs | |

200 OK · 7 ms · 329 B

```
[
    {
        "id": 3,
        "name": "Naduni",
        "email": "naduni@gmail.com",
        "course": "CS",
        "age": 23
    }
]
```

By name:



Student_Management_APIs / search by name or course

GET localhost:8080/api/students/search?keyword=harshana

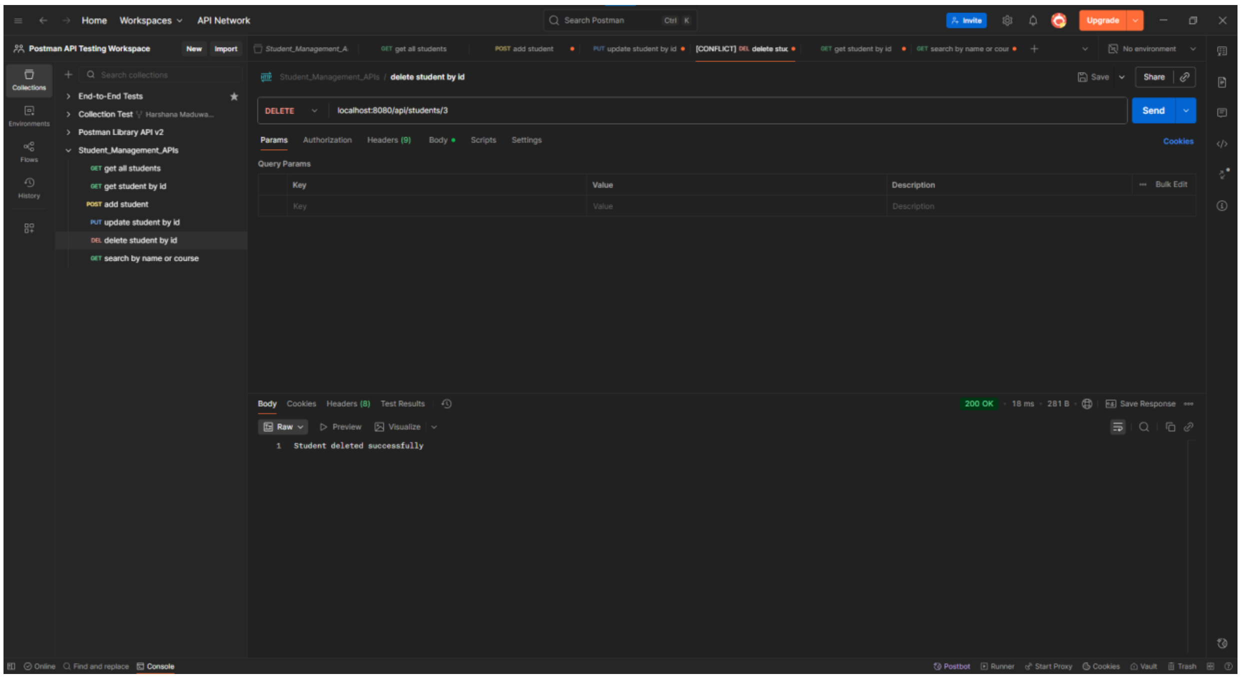
Query Params

| Key | Value | Description |
|---|---|---|
| keyword | harshana | |

200 OK · 6 ms · 334 B

```
[
    {
        "id": 1,
        "name": "Harshana",
        "email": "harshana@gmail.com",
        "course": "ICT",
        "age": 25
    }
]
```

Delete a student by id

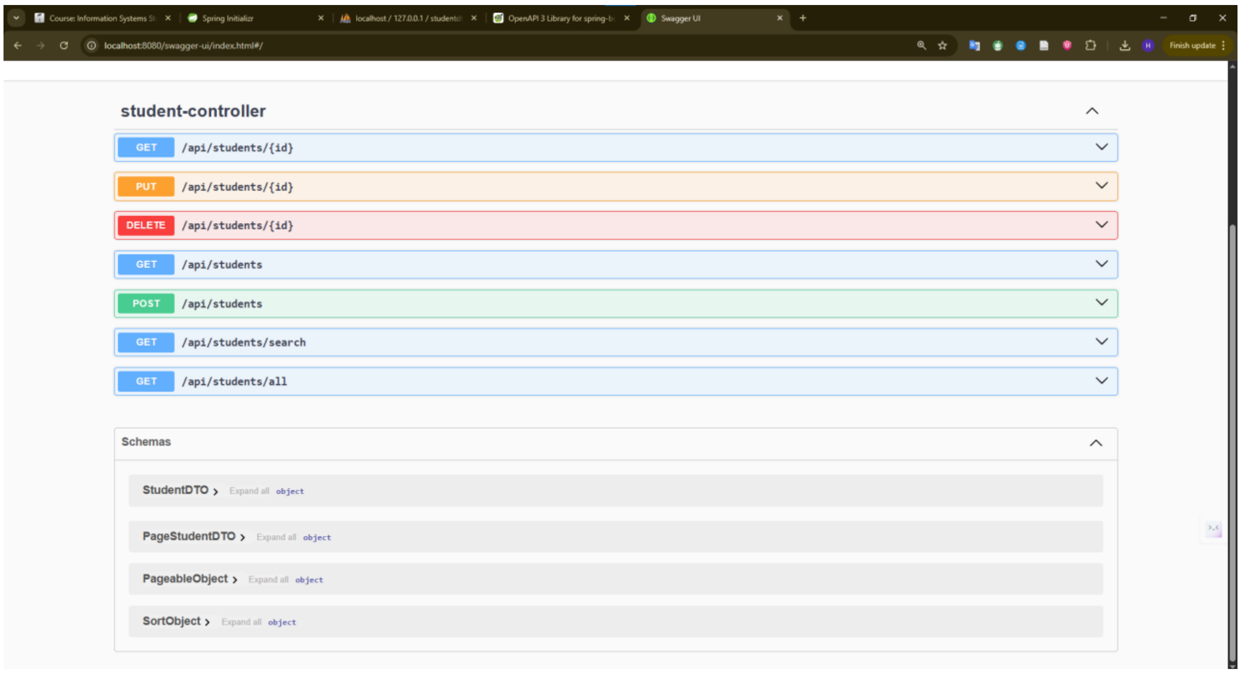


DELETE localhost:8080/api/students/3

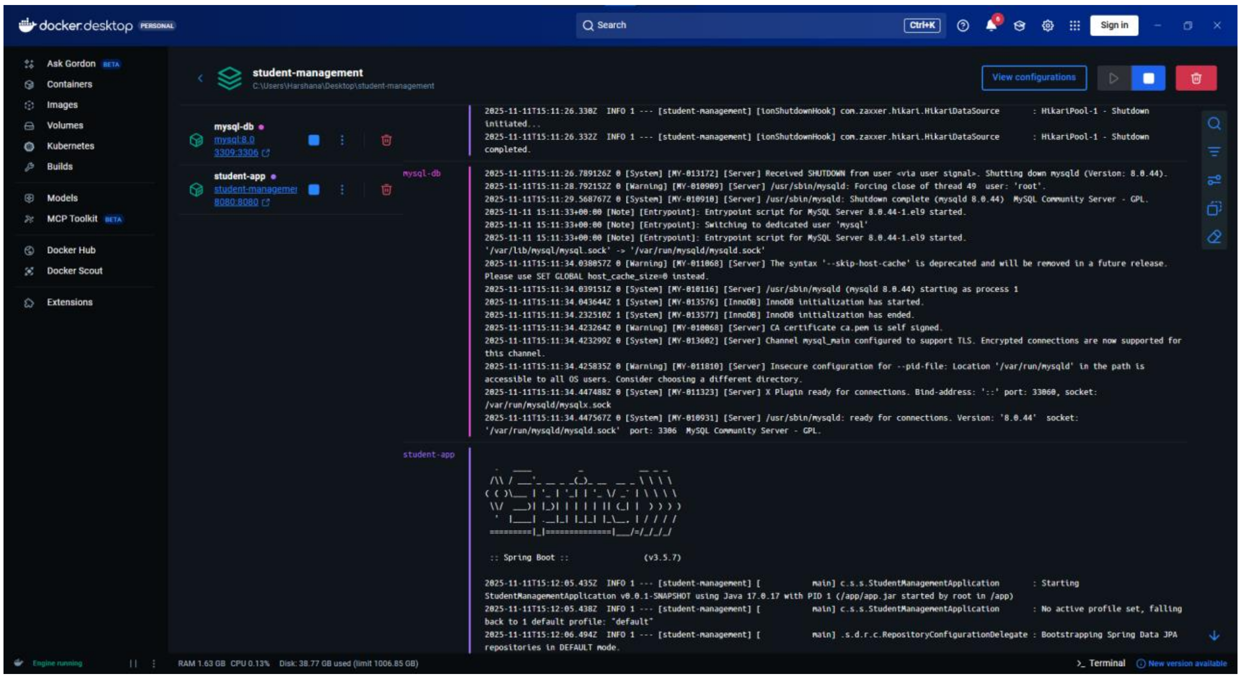
200 OK · 18 ms · 281 B

```
Student deleted successfully
```

Swagger UI


localhost:8080/swagger-ui/index.html#/
student-controller
GET /api/students/{id}
PUT /api/students/{id}
DELETE /api/students/{id}
GET /api/students
POST /api/students
GET /api/students/search
GET /api/students/all
Schemas
StudentDTO Expand all object
PageStudentDTO Expand all object
PageableObject Expand all object
SortObject Expand all object

Running on Docker



```
2025-11-11T15:11:26.330Z  INFO 1 --- [student-management] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown
initiated...
2025-11-11T15:11:26.332Z  INFO 1 --- [student-management] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown
completed.
```

```
2025-11-11T15:11:26.789126Z 0 [System] [MY-013172] [Server] Received SHUTDOWN from user <via user signal>. Shutting down mysqld (Version: 8.0.44).
2025-11-11T15:11:28.792152Z 0 [Warning] [MY-010909] [Server] /usr/sbin/mysqld: Forcing close of thread 49  user: 'root'.
2025-11-11T15:11:29.568767Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.44)  MySQL Community Server - GPL.
2025-11-11 15:11:33+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.44-1.el9 started.
2025-11-11 15:11:33+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2025-11-11 15:11:33+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.44-1.el9 started.
'/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
2025-11-11T15:11:34.038057Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release.
Please use SET GLOBAL host_cache_size=0 instead.
2025-11-11T15:11:34.039151Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.44) starting as process 1
2025-11-11T15:11:34.043644Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2025-11-11T15:11:34.232510Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2025-11-11T15:11:34.423264Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2025-11-11T15:11:34.423299Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for
this channel.
2025-11-11T15:11:34.425835Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is
accessible to all OS users. Consider choosing a different directory.
2025-11-11T15:11:34.447488Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket:
/var/run/mysqld/mysqlx.sock
2025-11-11T15:11:34.447567Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.44'  socket:
'/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server - GPL.
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.7)

2025-11-11T15:12:05.435Z  INFO 1 --- [student-management] [           main] c.s.s.StudentManagementApplication       : Starting
StudentManagementApplication v0.0.1-SNAPSHOT using Java 17.0.17 with PID 1 (/app/app.jar started by root in /app)
2025-11-11T15:12:05.438Z  INFO 1 --- [student-management] [           main] c.s.s.StudentManagementApplication       : No active profile set, falling
back to 1 default profile: "default"
2025-11-11T15:12:06.494Z  INFO 1 --- [student-management] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA
repositories in DEFAULT mode.
```