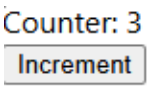
1.Write a simple React component that displays a counter initialized to 0. When the user clicks a button labeled "Increment", the counter should increase by 1 and update the display. Use the useState hook for managing the state.

=>**App.js**

```
import { useState } from 'react';

function App() {
  const [counter, setCounter] = useState(0);

  return (
<div>
     Counter: {counter}
           <br />
           <button onClick={() => setCounter(() => counter + 1)}>
            Increment
           </button>
</div>
  );
}
export default App;
```

OUTPUT:



2.Write a React component to display a list of employees in a table using useState. Add a button that appends a new employee to the list when clicked.

=>**Employees.js**

```
import { useState } from 'react';

function Employees() {
  const [employees, setEmployees] = useState([
    { empId: 101, name: "John", designation: "SE" },
    { empId: 102, name: "Tom", designation: "SSE" },
    { empId: 103, name: "Kevin", designation: "TA" }
  ]);

  const addEmployee = () => {
    setEmployees([...employees, { empId: 104, name: "Sanvi", designation: "TL" }]);
```

```
 };

 return (
<>
<table>
<thead>
<tr>
<th>EmpId</th>
<th>Name</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
        {employees.map((employee) => (
<tr key={employee.empId}>
<td>{employee.empId}</td>
<td>{employee.name}</td>
<td>{employee.designation}</td>
</tr>
        ))}
</tbody>
</table>
<button onClick={addEmployee}>Add an employee </button>
</>
 );
}
export default Employees;
```

OUTPUT:

EmpId	Name	Designation
101	John	SE
102	Tom	SSE
103	Kevin	TA
104	Sanvi	TL
104	Sanvi	TL

Add an employee

3. Build a React application using functional components that demonstrates the use of props for data sharing between components. For example Create a React app where employee data is maintained in the App component and passed to an Employees component using props. The Employees component should render the data in a table.

=> **App.js**

```
import { useState } from 'react';
import Employees from './Employees';

function App() {
  const [employees, setEmployees] = useState([
    { empId: 101, name: "John", designation: "SE" },
    { empId: 102, name: "Tom", designation: "SSE" },
    { empId: 103, name: "Kevin", designation: "TA" }
  ]);
  return (
<>
<Employees employees={employees} />
</>
  );
}
export default App;
```

Employees.js

```
function Employees(props) {

  return (
<>
<table>
<thead>
<tr>
<th>EmpId</th>
<th>Name</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
        {props.employees.map((employee) => (
<tr key={employee.empId}>
<td>{employee.empId}</td>
<td>{employee.name}</td>
<td>{employee.designation}</td>
```

```
</tr>
        ))}
</tbody>
</table>
</>
  );
}

export default Employees;
```

OUTPUT:

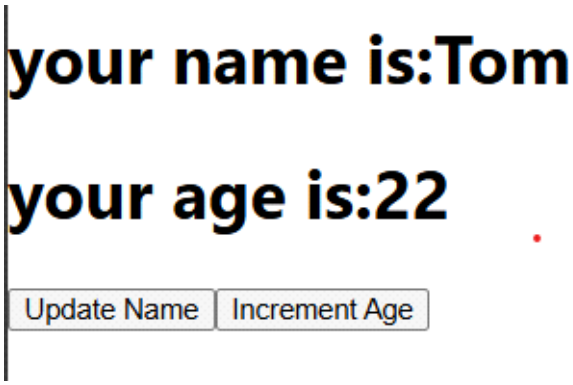
EmpId	Name	Designation
101	John	SE
102	Tom	SSE
103	Kevin	TA

4.Create a React component using useState to manage name and age.
Use useEffect to log a message whenever the component renders.
Provide buttons to update the name and increment the age.

=>App.js

```
function App() {
  const [name, setName] = useState("Jack");
  const [age, setAge] =useState(20);

  useEffect(() => {
    console.log("useEffect called")
  })

  return (
<>
<h1> your name is:{name}</h1>
<h1>your age is:{age}</h1>
<button onClick={() => setName("Tom")}> Update Name</button>
<button onClick={() => setAge(age+1)}>Increment Age</button>
</>
  );
}

export default App;
```

OUTPUT:



5. Create a React component that fetches user data from an API using Axios and displays the users' names and emails in a list.

=>App.js

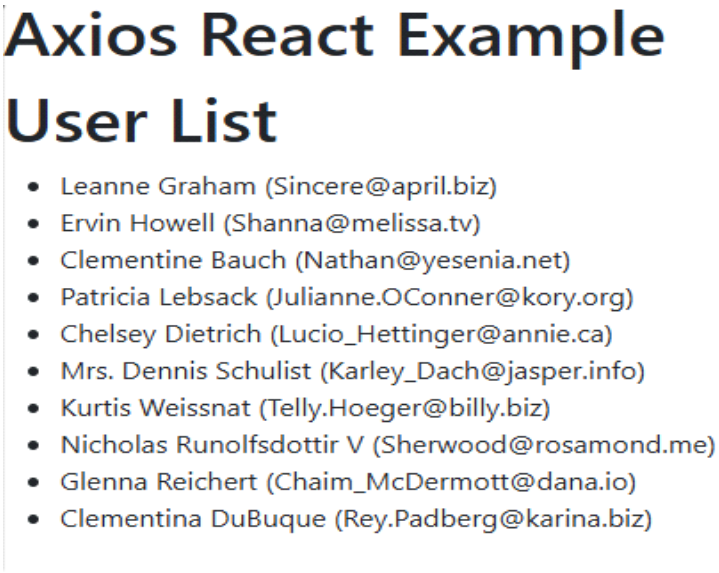
```
import React from 'react';
import UserList from './UserList';

function App() {
  return (
    <div className="App">
      <h1>Axios React Example</h1>
      <UserList />
    </div>
  );
}

export default App;
```

UserList.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";

function UserList() {
  const [users, setUsers] = useState([]);
  useEffect(() => {
    axios.get("https://jsonplaceholder.typicode.com/users")
      .then(response => {
        setUsers(response.data);
      })
      .catch(error => {
        console.log("Error fetching users", error);
      });
  }, []);

  return (
    <div>
      <h1>User List</h1>
      <ul>
        {users.map(user => (
          <li key={user.id}>
            {user.name}
            ({user.email})
          </li>
        ))}
      </ul>
    </div>
```

```
  );
}

export default UserList;
```

OUTPUT:



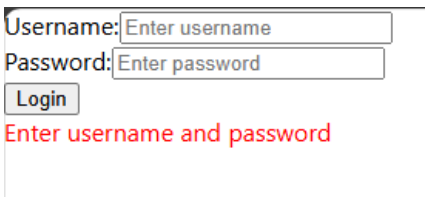
6.Create a React login form with controlled inputs for username and password. On submit, validate that both fields are filled:

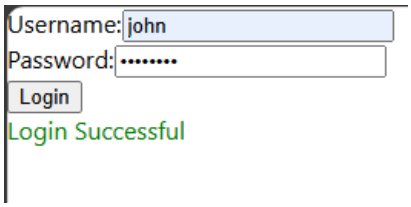
- **Show an error message if empty.**
- **Show a success message if both are entered.**

Login.js

```
import React, { useState } from 'react';

function Login() {
  const [status, setStatus] = useState(null);
  const [data, setData] = useState({
    userName: "",
    password: ""
  });

  const handleSubmit = (event) => {
    event.preventDefault();
    if (!data.userName || !data.password) {
      setStatus(false);
    } else {
      setStatus(true);
    }
  };

  const handleChange = (event) => {
    const { name, value } = event.target;
    setData({ ...data, [name]: value });
  };

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <div>
          <label htmlFor="userName">Username:</label>
          <input
            type="text"
            id="userName"
            name="userName"
            value={data.userName}
            onChange={handleChange}
```

```
          placeholder="Enter username"
        />
      </div>

      <div>
        <label htmlFor="password">Password:</label>
        <input
          type="password"
          id="password"
          name="password"
          value={data.password}
          onChange={handleChange}
          placeholder="Enter password"
        />
      </div>

      <button type="submit">Login</button>

      {status === false &&<div style={{ color: 'red' }}>Enter username and password</div>}
      {status === true &&<div style={{ color: 'green' }}>Login Successful</div>}
    </form>
  </div>
 );
}

export default Login;
```





7.Create a React app that lets you add student details via a form and shows them in a table.

Answer:

- **App holds student data state.**
- **StudentForm collects input and adds new students.**
- **StudentTable displays the list of students.**
- **Data and functions are passed via props between components.**

App.js

```
import React, { useState } from "react";
import StudentForm from "./StudentForm";
import StudentTable from "./StudentTable";

function App() {
  const [studentData, setStudentData] = useState([]);

  const addNewRecord = (registerNumber, name, semester, percentage) => {
    const newRecord = { registerNumber, name, semester, percentage };
          setStudentData(studentDate=>[...studentData, newRecord]);
  };

  return (
    <div>
      <h1>Student Management</h1>
      <StudentForm addNewRecord={addNewRecord} />
      <StudentTable studentData={studentData} />
    </div>
  );
}

export default App;
```

StudentForm.js

```
import React, { useState } from "react";

function StudentForm(props) {
```

```
const [registerNumber, setRegisterNumber] = useState("");
const [name, setName] = useState("");
const [semester, setSemester] = useState("");
const [percentage, setPercentage] = useState("");

const handleSubmit = (e) => {
  e.preventDefault();
  props.addNewRecord(registerNumber, name, semester, percentage);
};

return (

  <div>
    <form >
      <h2>Enter Student Details</h2>

      <label htmlFor="registerNumber">Register Number:</label>
      <input
        type="text"
        id="registerNumber"
        placeholder="Enter register number"
        value={registerNumber}
        onChange={(e) => setRegisterNumber(e.target.value)}
      /><br />

      <label htmlFor="name">Name:</label>
      <input
        type="text"
        id="name"
        placeholder="Enter name"
        value={name}
        onChange={(e) => setName(e.target.value)}
      /><br />

      <label htmlFor="semester">Semester:</label>
      <input
        type="text"
        id="semester"
        placeholder="Enter semester"
        value={semester}
        onChange={(e) => setSemester(e.target.value)}
      /><br />

      <label htmlFor="percentage">Percentage:</label>
```

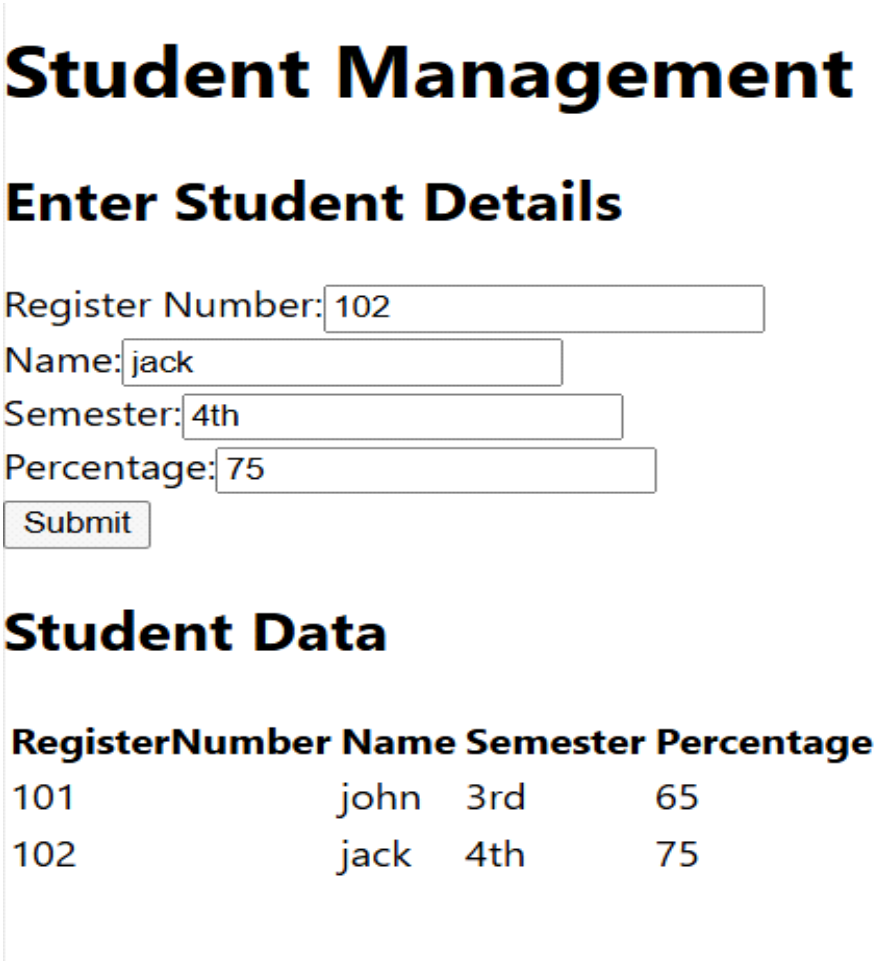
```
      <input
        type="text"
        id="percentage"
        placeholder="Enter percentage"
        value={percentage}
        onChange={(e) => setPercentage(e.target.value)}
      /><br />

      <input type="Submit" onClick={handleSubmit}></input>
    </form>
   </div>
  );
}

export default StudentForm;
```

StudentTable.js

```
import React from "react";

function StudentTable(props) {

  return (
   <div>
    <h2>Student Data</h2>
    <table>
     <thead>
      <tr>
       <th>RegisterNumber</th>
       <th>Name</th>
       <th>Semester</th>
       <th>Percentage</th>
      </tr>
     </thead>

     <tbody>
      {props.studentData.map((record) => (
       <tr key={record.registerNumber}>
        <td>{record.registerNumber}</td>
        <td>{record.name}</td>
        <td>{record.semester}</td>
        <td>{record.percentage}</td>
       </tr>
```

```
        ))}
      </tbody>
    </table>
  </div>
 );
}

export default StudentTable;
```

OUTPUT:



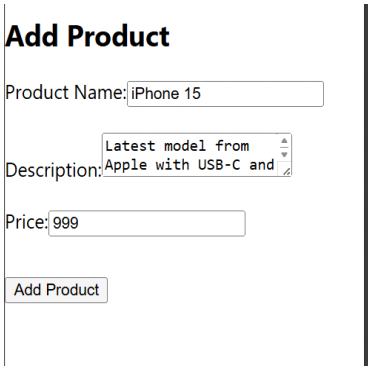
8. Create a form to add new product detail to the product catalogue using react js.

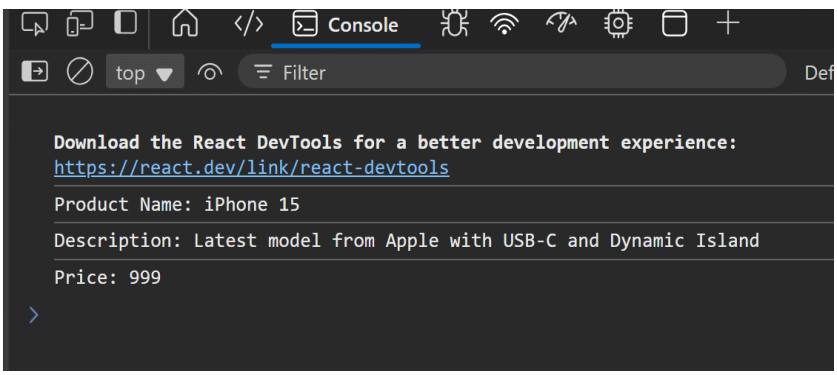
=>**ProductForm.js**

```
import React, { useState } from "react";
const ProductForm = () => {
 const [productName, setProductName] = useState("");
 const [description, setDescription] = useState("");
 const [price, setPrice] = useState("");

 const handleSubmit = (event) => {
  event.preventDefault();
  console.log("Product Name:", productName);
  console.log("Description:", description);
  console.log("Price:", price);
 };

 return (
  <form onSubmit={handleSubmit}>
   <h2>Add Product</h2>

   <div>
    <label htmlFor="productName">Product Name:</label>
    <input
     type="text"
     id="productName"
     value={productName}
     onChange={(e) => setProductName(e.target.value)}
     required
    />
   </div><br/>

  <div>
```

```
      <label htmlFor="description">Description:</label>
      <textarea
        id="description"
        value={description}
        onChange={(e) => setDescription(e.target.value)}
        required
      />
    </div><br/>

      <div style={{ marginBottom: "10px" }}>
        <label htmlFor="price">Price:</label>
        <input
          type="number"
          id="price"
          value={price}
          onChange={(e) => setPrice(e.target.value)}
          required
        />
      </div><br/>
      <button type="submit">Add Product</button>
    </form>
  );
};

export default ProductForm;
```

OUTPUT:





9.Implement programmatical navigation between different components using react router.

=> App.js

```
import "./App.css";
import { BrowserRouter, Route, Routes } from "react-router-dom";
import Home from "./Home";
import About from "./About";
import Contact from "./Contact";
import Navbar from "./Navbar";

function App() {
  return (
    <div>
      <BrowserRouter>
        <Navbar />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contact" element={<Contact />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

Navbar.js

```
import React from "react";
import { Link } from "react-router-dom";

const Navbar = () => {
  return (
    <nav>
      <Link to="/">HOME</Link> |
      <Link to="/about">ABOUT</Link> |
      <Link to="/contact"> CONTACT</Link>
    </nav>
  );
};

export default Navbar;
```

Home.js

```
import React from "react";

const Home = () => {
  return (
    <div>
      <h2>Welcome to Home page</h2>
    </div>
  );
};

export default Home;
```

About.js

```
import React from "react";

const About = () => {
  return (
    <div>
      <h2>This is ABOUT page</h2>
    </div>
  );
};

export default About;
```

Contact.js

```
import React from "react";

const Contact = () => {
  return (
    <div>
      <h2>This is CONTACT page</h2>
    </div>
  );
};

export default Contact;
```

OUTPUT:


HOME |ABOUT | CONTACT
Welcome to Home page


HOME |ABOUT | CONTACT
This is ABOUT page

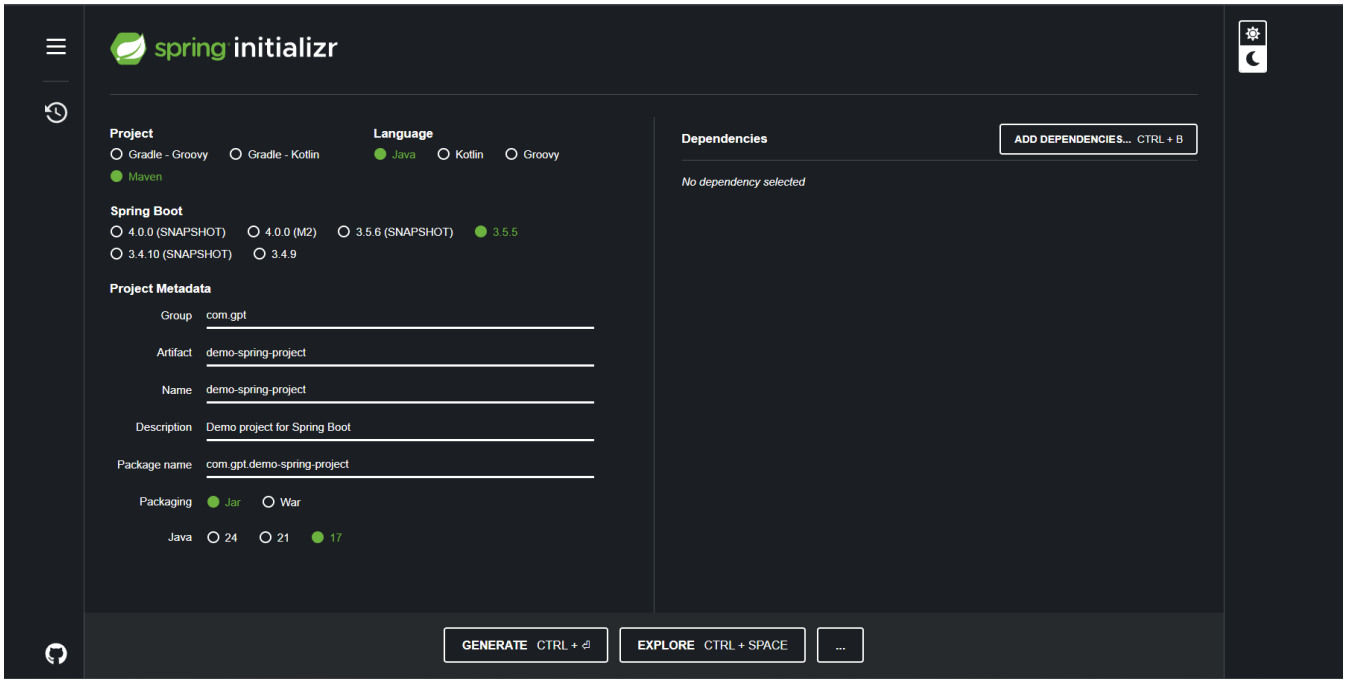

HOME |ABOUT | CONTACT
This is CONTACT page

⇨ Spring Initializr:

It is an online tool provided by Spring for generating Spring Boot applications which is accessible at http://start.spring.io. You can use it for creating a Spring Boot project using the following steps:

Step 1: Launch Spring Initializr to create your Spring Data Boot application and do the below.

Group and Artifact under Project Metadata stand for package name and project name respectively. Give them any valid value according to your project, retain other default values (Project, Language & Spring Boot version). Add dependencies required to run your Spring Boot application.



Note: This screen keeps changing depending on updates from Pivotal and change in the Spring Boot version.

Step 2: Select Project as Maven, Language as Java, and Spring Boot as 2.2.6 and the necessary dependencies for SpringORM as shown in the image, then enter the project details as follows:
Choose com.gpt as Group
Choose demo-spring-project as Artifact
Click on More options and choose com.gpt as package Name

Step 3: Click on Generate Project. This would download a zip file to the local machine.

Step 4: Unzip the zip file and extract to a folder.

Step 5: In Eclipse/ STS, Click File → Import → Existing Maven Project. Navigate or type in the path of

the folder where you extracted the zip file to the next screen.
We have successfully created a Spring Boot Maven-based project .

10.SpringIOC in spring

Write a Spring IoC application using annotation-based configuration to implement a CustomerService interface with a method createCustomer() that returns: "Customer is successfully created".

=>

CustomerService.java

```
package com.infy.service;

public interface CustomerService {
        public String createCustomer();
}
```

CustomerServiceImpl.java

```
package com.infy.service;
public class CustomerServiceImpl implements CustomerService {
   @Override
   public String createCustomer() {
      return "Customer is successfully created";
   }
}
```

SpringConfiguration.java

```
package com.infy.util;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import com.infy.service.CustomerService;
import com.infy.service.CustomerServiceImpl;

@Configuration
public class SpringConfiguration {

   @Bean(name = "customerService")
   public CustomerServiceImpl customerServiceImpl() {
           return new CustomerServiceImpl();
   }
}
```

Client.java

```
package com.infy;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.AbstractApplicationContext;
```

```
import com.infy.service.CustomerService;
import com.infy.util.SpringConfiguration;

public class Client {
   public static void main(String[] args) {
      AbstractApplicationContext context =
            new AnnotationConfigApplicationContext(SpringConfiguration.class);

      CustomerService service = (CustomerService) context.getBean("customerService");
      System.out.println(service.createCustomer());
      context.close();
   }
}
```

->Run the Client class

- Run the Client class as a **Java Application** (right-click > Run As > Java Application).
- This will load the Spring context, get the customerService bean, call createCustomer(), and print the output.

Output: Customer is successfully created

11. Implement a Spring IoC with Java-based configuration to manage an `Employee` service that supports `insert()` and `delete()` operations.

=>

Employee.java

```
package com.infy.service;

public interface Employee {
   public void insert();
   public void delete();
}
```

EmployeeImpl.java

```
package com.infy.service;

public class EmployeeImpl implements Employee {
   @Override
   public void insert() {
      System.out.println("Employee inserted");
   }
   @Override
   public void delete() {
      System.out.println("Employee deleted");
   }
}
```

SpringConfiguration.java

```
package com.infy.util;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import com.infy.service.EmployeeImpl;

@Configuration
public class SpringConfiguration {

   @Bean(name = "employeeImpl")
   public EmployeeImpl employeeImpl() {
            return new EmployeeImpl();
   }
}
```

App.java

```
package com.infy;

import org.springframework.context.ApplicationContext;
```

```
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import com.infy.service.EmployeeImpl;
import com.infy.util.SpringConfiguration;

public class App {
   public static void main(String[] args) {
      ApplicationContext context =
            new AnnotationConfigApplicationContext(SpringConfiguration.class);
      EmployeeImpl e = (EmployeeImpl) context.getBean("employeeImpl");
      e.insert();
      e.delete();
   }
}
```

OUTPUT: Employee inserted
Employee deleted

12. Demonstrate Constructor-Based Dependency Injection in Spring

Write a Spring application using Java-based configuration to implement constructor injection, where:

- **A CustomerRepository and an integer value are injected into CustomerServiceImpl via constructor.**
- **Use @Configuration and @Bean annotations to manage the beans.**
- **Use AnnotationConfigApplicationContext to retrieve and use the service bean.**

=>

SpringConfiguration.java

```
package com.infy.configuration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import com.infy.repository.CustomerRepository;
import com.infy.service.CustomerServiceImpl

@Configuration
public class SpringConfiguration {
   @Bean
   public CustomerServiceImpl customerService() {
      return new CustomerServiceImpl (repository(),20);
   }
   @Bean
   public CustomerRepository repository() {
      return new CustomerRepository();
   }
}
```

CustomerService.java

```
package com.infy.service;
public interface CustomerService {
   public String fetchCustomer();
   public String createCustomer();
}
```

CustomerServiceImpl .java

```
package com.infy.service;
import com.infy.repository.CustomerRepository;
public class CustomerServiceImpl implements CustomerService {
          private  int count;
          private CustomerRepository repository;
   public CustomerServiceImpl(CustomerRepository repository, int count ) {
```

```
        this.count = count ;
         this.repository = repository ;
 }

   public String fetchCustomer() {
        return repository.fetchCustomer(count);
   }

   public String createCustomer() {
        return repository. createCustomer();
   }
}
```

CustomerRepository.java

```
package com.infy.repository;
public class CustomerRepository {
        public String fetchCustomer(int count) {
                return "The number of Customer feched are : " + count;
        }
        public String createCustomer() {
                return "Customer is successfuly cancled";
        }
}
```

App.java

```
package com.infy
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.AbstractApplicationContext;
import com.infy.configuration.SpringConfiguration;
import com.infy.service.CustomerServiceImpl;
public class App
{
   public static void main( String[] args )
   {
        CustomerServiceImpl service = null ;
        AbstractApplicationContext context = new
AnnotationConfigApplicationContext(SpringConfiguration.class);
        service = (CustomerServiceImpl) context.getBean("customerService");
      System.out.println( service.fetchCustomer());
      context.close();
   }
}
```

Output: The number of Customer feched are : 20

13. Demonstrate Setter-Based Dependency Injection in Spring

Create a Spring application demonstrating setter-based dependency injection using Java configuration. The application should:

- **Define a CustomerService interface and CustomerServiceImpl class with properties count (int) and CustomerRepository.**
- **Inject dependencies into CustomerServiceImpl via setter methods.**
- **Use @Configuration and @Bean annotations to configure the beans.**
- **Fetch and print the number of customers fetched based on the injected count value.**

SpringConfiguration.java

```
package com.infy.util;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import com.infy.repository.CustomerRepository;
import com.infy.service.CustomerServiceImpl;
@Configuration
public class SpringConfiguration {
	@Bean // Setter Injection
	public CustomerRepository customerRepository() {
		CustomerRepository customerRepository = new CustomerRepository();
		return customerRepository;
	}
	@Bean // Setter Injection
	public CustomerServiceImpl customerService() {
		CustomerServiceImpl customerService = new CustomerServiceImpl();
		customerService.setCount(10);
		customerService.setRepository(customerRepository());
		return customerService;
	}
}
```

CustomerService.java

```
package com.infy.service;
public interface CustomerService {
	public String fetchCustomer();
	public String createCustomer();
}
```

CustomerServiceImpl.java

```
package com.infy.service;
import com.infy.repository.CustomerRepository;
public class CustomerServiceImpl implements CustomerService {
	private int count;
```

```
	private CustomerRepository repository;
	public CustomerServiceImpl() {
	}
	public void setCount(int count) {
		this.count = count;
	}
	public void setRepository(CustomerRepository repository) {
		this.repository = repository;
	}
	public String fetchCustomer() {
		return repository.fetchCustomer(count);
	}
	public String createCustomer() {
		return repository.createCustomer();
	}
}
```

CustomerRepository.java

```
package com.infy.repository;
public class CustomerRepository {
	public String fetchCustomer(int count) {
		return " The no of customers fetched are : " + count;
	}
	public String createCustomer() {
		return "Customer is successfully created";
	}
}
```

Client.java

```
package com.infy;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.AbstractApplicationContext;
import com.infy.service.CustomerServiceImpl;
import com.infy.util.SpringConfiguration;
public class Client {
	public static void main(String[] args) {
		CustomerServiceImpl service = null;
		AbstractApplicationContext context = new
AnnotationConfigApplicationContext(SpringConfiguration.class);
		service = (CustomerServiceImpl) context.getBean("customerService");
		System.out.println(service.fetchCustomer());
		context.close();
	}
}
```

Output : The no of customers fetched are : 10

14.Implement Constructor Injection ,For the Employee Management scenario, create a Spring Java configuration to maintain Employee details such as employee id, employee name, and department and perform the following database operations.

- **Insert a new employee details**
- **Remove employee details based on employee id**
- **Search all employee**

EmployeeDTO.java

```
// EmployeeDTO.java
package com.infy.Service;
public class EmployeeDTO {
   private int empId;
   private String empName;
   private String empDepartment;

   public EmployeeDTO() {}
   public EmployeeDTO(int empId, String empName, String empDepartment) {
      this.empId = empId;
      this.empName = empName;
      this.empDepartment = empDepartment;
   }
   public int getEmpId() {
      return empId;
   }
   public void setEmpId(int empId) {
      this.empId = empId;
   }
   public String getEmpName() {
      return empName;
   }
   public void setEmpName(String empName) {
      this.empName = empName;
   }
   public String getEmpDepartment() {
      return empDepartment;
   }
   public void setEmpDepartment(String empDepartment) {
      this.empDepartment = empDepartment;
   }
   @Override
   public String toString() {
      return "empID: " + empId +
            " empName: " + empName +
            " empDepartment: " + empDepartment;
   }
```

```
}
```

EmployeeRepository.java

```
// EmployeeRepository.java
package com.infy.Service;
import java.util.List;
public interface EmployeeRepository {
   public void insertEmployee(EmployeeDTO emp);
   public void removeEmployee(int empId);
   List<EmployeeDTO> fetchCustomers();
}
```

EmployeeRepositoryImpl.java

```
package com.infy.Service;
import java.util.List;
import java.util.ArrayList;
import jakarta.annotation.PostConstruct;

public class EmployeeRepositoryImpl implements EmployeeRepository {
List<EmployeeDTO> employees = null;
   @PostConstruct
   public void initializer() {
      EmployeeDTO employeeDTO = new EmployeeDTO();
      employeeDTO.setEmpId(101);
      employeeDTO.setEmpName("Sujata");
      employeeDTO.setEmpDepartment("CSE");
      employees = new ArrayList<>();
      employees.add(employeeDTO);
   }
   @Override
   public void insertEmployee(EmployeeDTO emp) {
      employees.add(emp);
   }
   @Override
   public void removeEmployee(int empId) {
      if (employees == null) return;
      employees.removeIf(emp -> emp.getEmpId() == empId);
   }
   @Override
   public List<EmployeeDTO> fetchCustomers() {
      return employees;
   }
}
```

EmployeeService.java

```
package com.infy.Service;
```

```
import java.util.List;

public interface EmployeeService {
   public void insert(EmployeeDTO emp);
   public void delete(int empId);
   List<EmployeeDTO> getAllCustomer();
}
```

EmployeeServiceImpl.java

```
package com.infy.Service;
import java.util.List;
public class EmployeeServiceImpl implements EmployeeService {
   private EmployeeRepository employeeDAO;

   public EmployeeServiceImpl(EmployeeRepository employeeRepository) {
      this.employeeDAO = employeeRepository;
   }
   @Override
   public void insert(EmployeeDTO emp) {
      employeeDAO.insertEmployee(emp);
   }
   @Override
   public void delete(int empId) {
      employeeDAO.removeEmployee(empId);
   }

   @Override
   public List<EmployeeDTO> getAllCustomer() {
      return employeeDAO.fetchCustomers();
   }
}
```

SpringConfiguration.java

```
package com.infy.Service;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SpringConfiguration {
   @Bean
   public EmployeeRepositoryImpl employeeRepository() {
      return new EmployeeRepositoryImpl();
   }
   @Bean
```

```
    public EmployeeServiceImpl employeeService() {
        return new EmployeeServiceImpl(employeeRepository());
    }
}
```

App.java

```
package com.infy;
import com.infy.Service.*;
import java.util.List;
import java.util.ArrayList;
import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App {
    public static void main(String[] args) {
        AbstractApplicationContext context =
                new AnnotationConfigApplicationContext(SpringConfiguration.class);
        EmployeeService service = (EmployeeService) context.getBean("employeeService");
        service.insert(new EmployeeDTO(103, "Sneha", "CS"));
        service.insert(new EmployeeDTO(104, "John", "EEE"));
        service.insert(new EmployeeDTO(105, "Kevin", "IT"));
        service.delete(101);
        service.delete(102);

        List<EmployeeDTO> empList = new ArrayList<>();
        empList = service.getAllCustomer();
        for (EmployeeDTO e : empList) {
            System.out.println(e);
        }
        context.close();
    }
}
```

Output : empID: 103 empName: Sneha empDepartment: CS
empID: 104 empName: John empDepartment: EEE
empID: 105 empName: Kevin empDepartment: IT

15.Implement Setter Injection, For the Employee Management scenario, create a Spring Java configuration to maintain Employee details such as employee id, employee name, and department and perform the following database operations.

- **Insert a new employee details**
- **Remove employee details based on employee id**
- **Search all employee**

EmployeeDTO.java

```
package com.infy.Service;
public class EmployeeDTO {
   private int empId;
   private String empName;
   private String empDepartment;

   public EmployeeDTO() {}
   public EmployeeDTO(int empId, String empName, String empDepartment) {
      this.empId = empId;
      this.empName = empName;
      this.empDepartment = empDepartment;
   }
   public int getEmpId() {
      return empId;
   }
   public void setEmpId(int empId) {
      this.empId = empId;
   }

   public String getEmpName() {
      return empName;
   }
   public void setEmpName(String empName) {
      this.empName = empName;
   }
   public String getEmpDepartment() {
      return empDepartment;
   }
   public void setEmpDepartment(String empDepartment) {
      this.empDepartment = empDepartment;
   }
   @Override
   public String toString() {
      return "empID: " + empId +
            " empName: " + empName +
            " empDepartment: " + empDepartment;
   }
```

```
}
```

EmployeeRepository.java

```
// EmployeeRepository.java
package com.infy.Service;
import java.util.List;
public interface EmployeeRepository {
   public void insertEmployee(EmployeeDTO emp);
   public void removeEmployee(int empId);
   List<EmployeeDTO> fetchCustomers();
}
```

EmployeeRepositoryImpl.java

```
package com.infy.Service;
import java.util.List;
import java.util.ArrayList;
import jakarta.annotation.PostConstruct;

public class EmployeeRepositoryImpl implements EmployeeRepository {
List<EmployeeDTO> employees = null;
   @PostConstruct
   public void initializer() {
      EmployeeDTO employeeDTO = new EmployeeDTO();
      employeeDTO.setEmpId(101);
      employeeDTO.setEmpName("Sujata");
      employeeDTO.setEmpDepartment("CSE");
      employees = new ArrayList<>();
      employees.add(employeeDTO);
   }
   @Override
   public void insertEmployee(EmployeeDTO emp) {
      employees.add(emp);
   }
   @Override
   public void removeEmployee(int empId) {
      if (employees == null) return;
      employees.removeIf(emp -> emp.getEmpId() == empId);
   }
   @Override
   public List<EmployeeDTO> fetchCustomers() {
      return employees;
   }
}
```

EmployeeService.java

```
package com.infy.Service;

import java.util.List;

public interface EmployeeService {
   public void insert(EmployeeDTO emp);
   public void delete(int empId);
   List<EmployeeDTO> getAllCustomer();
}
```

EmployeeServiceImpl.java

```
package com.infy.service;
import java.util.List;
public class EmployeeServiceImpl implements EmployeeService {
   private EmployeeRepository employeeDAO;

   // Setter Injection
   public void setEmployeeDAO(EmployeeRepository employeeRepository) {
      this.employeeDAO = employeeRepository;
   }
   @Override
   public void insert(EmployeeDTO emp) {
      employeeDAO.insertEmployee(emp);
   }
   @Override
   public void delete(int empId) {
      employeeDAO.removeEmployee(empId);
   }
   @Override
   public List<EmployeeDTO> getAllCustomer() {
      return employeeDAO.fetchCustomers();
   }
}
```

SpringConfiguration.java

```
package com.infy.Service;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SpringConfiguration {
   @Bean
   public EmployeeRepositoryImpl employeeRepository() {
      return new EmployeeRepositoryImpl();
```

```
    }
    @Bean
    public EmployeeServiceImpl employeeService() {
            EmployeeServiceImpl service = new EmployeeServiceImp();
Service.setEmployeeDTO(employeeRepository());
Return service;
    }
}
```

App.java

```
package com.infy;
import com.infy.Service.*;
import java.util.List;
import java.util.ArrayList;
import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App {
    public static void main(String[] args) {
        AbstractApplicationContext context =
                new AnnotationConfigApplicationContext(SpringConfiguration.class);
        EmployeeService service = (EmployeeService) context.getBean("employeeService");
        service.insert(new EmployeeDTO(103, "Sneha", "CS"));
        service.insert(new EmployeeDTO(104, "John", "EEE"));
        service.insert(new EmployeeDTO(105, "Kevin", "IT"));
        service.delete(101);
        service.delete(102);

        List<EmployeeDTO> empList = new ArrayList<>();
        empList = service.getAllCustomer();
        for (EmployeeDTO e : empList) {
            System.out.println(e);
        }
        context.close();
    }
}
```

Output : empID: 103 empName: Sneha empDepartment: CS
empID: 104 empName: John empDepartment: EEE
empID: 105 empName: Kevin empDepartment: IT

16. Implement Spring's component scanning with @ComponentScan and @Service can be used to automatically detect and configure beans.

CustomerService.java

```
package com.infy.service;

public interface CustomerService {
	public String fetchCustomer(int count);
	public String createCustomer();
}
```

CustomerServiceImpl.java

```
package com.infy.service;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
@Service("customerService")
public class CustomerServiceImpl implements CustomerService {
	@Value("10")
	private int count;
	public String fetchCustomer(int count) {
		return " The no of customers fetched are : " + count;
	}
	public String createCustomer() {
		return "Customer is successfully created";
	}
}
```

SpringConfiguration .java

```
package com.infy.util;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@Configuration
@ComponentScan(basePackages="com.infy")
public class SpringConfiguration {
}
```

App.java

```
package com.infy;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.AbstractApplicationContext;
import com.infy.service.CustomerServiceImpl;
import com.infy.util.SpringConfiguration;
```

```
public class App {
	public static void main(String[] args) {
		CustomerServiceImpl service = null;
		AbstractApplicationContext context = new
AnnotationConfigApplicationContext(SpringConfiguration.class);
		service = (CustomerServiceImpl) context.getBean("customerService");
		System.out.println(service.createCustomer());
		context.close();
	}
}
```

Output: Customer is successfully created

17.Spring boot with employee management. For the Employee Management scenario, create a Spring Java configuration to maintain Employee details such as employee id, employee name, and department and perform the following database operations.

- **Insert a new employee details**
- **Remove employee details based on employee id**
- **Search all employee**

EmployeeDTO.java

```
package com.infy.dto;

public class EmployeeDTO {
	private int empIdid;
	private String empName;
	private String empDept;

	public EmployeeDTO() {
		super();
	}

	public EmployeeDTO(int empIdid, String empName, String empDept) {
		this.empIdid = empIdid;
		this.empName = empName;
		this.empDept = empDept;
	}
	public int getEmpIdid() {
		return empIdid;
	}
	public void setEmpIdid(int empIdid) {
		this.empIdid = empIdid;
	}
	public String getEmpName() {
		return empName;
	}
	public void setEmpName(String empName) {
		this.empName = empName;
	}
	public String getEmpDept() {
		return empDept;
	}
	public void setEmpDept(String empDept) {
		this.empDept = empDept;
	}
	@Override
	public String toString() {
```

```
		return "EmployeeDTO [empIdid=" + empIdid + ", " + (empName != null ?
"empName=" + empName + ", " : "")
				+ (empDept != null ? "empDept=" + empDept : "") + "]";
	}

}
```

EmployeeRepository.java

```
package com.infy.repo;

import org.springframework.stereotype.Component;
import com.infy.dto.EmployeeDTO;
import java.util.ArrayList;
import java.util.List;
import jakarta.annotation.PostConstruct;

@Component
public class EmployeeRepository {
	List<EmployeeDTO> employee=null;

	@PostConstruct
	public void initilizer() {

		EmployeeDTO employeeDTO=new EmployeeDTO();
		employeeDTO.setEmpIdid(101);
		employeeDTO.setEmpName("Jack");
		employeeDTO.setEmpDept("IT");
		employee=new ArrayList<>();
		employee.add(employeeDTO);
	}


	public void insertEmployee(EmployeeDTO employeeDTO) {
		employee.add(employeeDTO);
	}


	public void removeEmployee(int empId) {
		employee.remove(empId);
	}

	public List<EmployeeDTO> fetchEmployee(){
		return employee;
	}
```

```
}
```

EmployeeService.java

```
package com.infy.service;
import java.util.List;
import com.infy.dto.EmployeeDTO;

public interface EmployeeService {
	public void insert(EmployeeDTO employeeDTO);
	public void delete (int empId);
	public List<EmployeeDTO> getALLEmployee();
}
```

EmployeeServiceImpl.java

```
package com.infy.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.infy.dto.EmployeeDTO;
import com.infy.repo.EmployeeRepository;

@Service("employeeService")
public class EmployeeServiceImpl implements EmployeeService {

		EmployeeRepository employeeDAO;

		@Autowired
		public EmployeeServiceImpl (EmployeeRepository repository) {
			employeeDAO=repository;
		}

		@Override
		public void insert(EmployeeDTO employeeDTO) {
			employeeDAO.insertEmployee(employeeDTO);
		}

		@Override
		public void delete(int empId) {
			employeeDAO.removeEmployee(empId);
		}

		@Override
		public List<EmployeeDTO> getALLEmployee(){
			return employeeDAO.fetchEmployee();
```

```
		}
}
```

App.java

```
package com.infy;
import java.util.ArrayList;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.support.AbstractApplicationContext;
import com.infy.dto.EmployeeDTO;
import com.infy.service.EmployeeServiceImpl;
import java.util.List;

@SpringBootApplication
public class App {

	public static void main(String[] args) {
		AbstractApplicationContext
context=(AbstractApplicationContext)SpringApplication.run(App.class, args);
		EmployeeServiceImpl
service=(EmployeeServiceImpl)context.getBean("employeeService");

		service.insert(new EmployeeDTO(102,"kevin","SC"));
		service.insert(new EmployeeDTO(103,"Jammes","CS"));
		service.insert(new EmployeeDTO(104,"Alice","IIT"));
		service.insert(new EmployeeDTO(105,"Leo","ETA"));

		service.delete(1);
		service.delete(2);

		List<EmployeeDTO> empList=new ArrayList<>();
		empList=service.getALLEmployee();
		for(EmployeeDTO e:empList) {
			System.out.println(e);
		}
	context.close();
	}

}
```

Output:

```
EmployeeDTO [empIdid=101, empName=Jack, empDept=IT]
EmployeeDTO [empIdid=103, empName=Jammes, empDept=CS]
EmployeeDTO [empIdid=105, empName=Leo, empDept=ETA]
```

18.Create a Spring Boot application using Spring Data JPA to manage students with fields: studentId, studentName, and studentAddress.

=> Student.java

```
package com.infy.domain;

import com.infy.dto.StudentDTO;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Student {
	@Id
	private Integer studentId;
	private String studentName;
	private String  StudentAddress;

	public Student() {
		super();
	}
	public Student(int studentId, String studentName, String studentAddress) {
		this.studentId = studentId;
		this.studentName = studentName;
		StudentAddress = studentAddress;
	}
	public int getStudentId() {
		return studentId;
	}
	public void setStudentId(int studentId) {
		this.studentId = studentId;
	}
	public String getStudentName() {
		return studentName;
	}
	public void setStudentName(String studentName) {
		this.studentName = studentName;
	}
	public String getStudentAddress() {
		return StudentAddress;
	}
	public void setStudentAddress(String studentAddress) {
		StudentAddress = studentAddress;
	}
	@Override
	public String toString() {
```

```
		return "Student [" + (studentId != null ? "studentId=" + studentId + ", " : "")
				+ (studentName != null ? "studentName=" + studentName + ", " : "")
					+ (StudentAddress != null ? "StudentAddress=" + StudentAddress :
""") + "]";
	}

	public static StudentDTO convertEntityToDto(Student student) {
		StudentDTO studentDTO=new StudentDTO();
		studentDTO.setStudentId(student.getStudentId());
		studentDTO.setStudentName(student.getStudentName());
		studentDTO.setStudentAddress(student.getStudentAddress());

		return studentDTO;
	}
}
```

StudentDTO.java

```
package com.infy.dto;

import com.infy.domain.Student;
public class StudentDTO {
	private Integer studentId;
	private String studentName;
	private String  StudentAddress;
	public StudentDTO() {
		super();

	}
	public StudentDTO(int studentId, String studentName, String studentAddress) {
		this.studentId = studentId;
		this.studentName = studentName;
		StudentAddress = studentAddress;
	}
	public int getStudentId() {
		return studentId;
	}
	public void setStudentId(int studentId) {
		this.studentId = studentId;
	}
	public String getStudentName() {
		return studentName;
	}
	public void setStudentName(String studentName) {
		this.studentName = studentName;
	}
```

```
	public String getStudentAddress() {
		return StudentAddress;
	}
	public void setStudentAddress(String studentAddress) {
		StudentAddress = studentAddress;
	}

	public static Student convertDtoToEntity(StudentDTO studentDTO) {
		Student student=new Student();
		student.setStudentId(studentDTO.getStudentId());
		student.setStudentName(studentDTO.getStudentName());
		student.setStudentAddress(studentDTO.getStudentAddress());

		return student;
	}

}
```

StudentRepository.java

```
package com.infy.repo;

import org.springframework.data.jpa.repository.JpaRepository;
import com.infy.domain.Student;
public interface StudentRepository extends JpaRepository<Student,Integer> {

}
```

StudentService.java

```
package com.infy.service;

import java.util.List;
import com.infy.dto.StudentDTO;

public interface StudentService {
	public void insertStudent(StudentDTO studentDTO);
	public List<StudentDTO> getALLStudent();
	public void deleteStudent(int studentId);
	public String updateStudent(int studentId, String newName);
}
```

StudentServiceImpl.java

```
package com.infy.service;

import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.infy.domain.Student;
import com.infy.dto.StudentDTO;
import com.infy.repo.StudentRepository;

@Service("employeeService")

public class StudentServiceImpl implements StudentService {
	@Autowired
	StudentRepository repository;

	public void insertStudent(StudentDTO studentDTO) {
		Student student=new Student();
		student=StudentDTO.convertDtoToEntity(studentDTO);
		repository.save(student);
	}
	public List<StudentDTO> getALLStudent(){
		List<Student> std=repository.findAll();
		List<StudentDTO> dto=new ArrayList<>();
		for(Student s:std) {
			dto.add(Student.convertEntityToDto(s));
		}
		return dto;
	}
	public void deleteStudent(int studentId) {
		repository.deleteById(studentId);
	}

	public String updateStudent(int studentId, String newName) {
		Optional<Student> optional=repository.findById(studentId);
		Student student=optional.get();
		student.setStudentName(newName);
		repository.save(student);
		return "The student with id:" + studentId + " has been updated successfully";
	}
}
```

App.java

```
package com.infy;

import java.util.ArrayList;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.infy.dto.StudentDTO;
import com.infy.service.StudentServiceImpl;
import java.util.List;
import java.util.Scanner;

@SpringBootApplication
public class App implements CommandLineRunner{

    public static void main(String[] args) {
            SpringApplication.run(App.class, args);
    }

    @Autowired
    StudentServiceImpl service;

    @Override
    public void run(String... args) throws Exception {
            service.insertStudent(new StudentDTO(101,"Alice","123 Main Street"));
            service.insertStudent(new StudentDTO(102,"Smith","New York"));
            service.insertStudent(new StudentDTO(103, "kevin","963 Ash Circle, Portland,"));
            service.insertStudent(new StudentDTO(104,"leo","234 sdf sdc"));

            List<StudentDTO> list=new ArrayList<>();
            list=service.getALLStudent();
            for(StudentDTO std:list) {
                    System.out.println(std);
            }

            Scanner s=new Scanner(System.in);
            System.out.println("Enter the student id to delete");
            int studentId1=s.nextInt();
            service.deleteStudent(studentId1);
            System.out.println("Enter the student id of the student whose current name has to
be update:");
            int studentId2=s.nextInt();
            System.out.println("Enter the new name for the student");
            String newName=s.next();
            String msg=service.updateStudent(studentId2, newName);
            System.out.println(msg);
            s.close();
    }
```

}

application.properties

```
spring.application.name=project
# Database configuration
spring.datasource.url=jdbc:mysql://localhost:3306/jpa
spring.datasource.username=root
spring.jpa.hibernate.ddl-auto=update
# Logging configuration
logging.level.root=INFO
logging.level.org.springframework=WARN
```

➔ **Run The Application:**

- Open xampp , start Apache and MySql server .
- Create a database named jpa in MySQL.
- Add 'Spring Data Jpa" (To work with JPA repositories and database) and "MySql Driver" dependencies while creating a maven project using spring initializr .
- Run the App class as a **Java Application** (right-click > Run As > Java Application).

OUTPUT:

```
Enter the student id to delete
104
Enter the student id of the student whose current name has to be update :
103
Enter the new name for the student
tom
The student with id:103 has been updated successfully
```

➔ The jpa data base is looks like as shown below:

19.Create a Spring Boot REST controller for student with GET, POST, PUT, DELETE methods that return simple success messages.

=>

StudentController.java

```
package com.gpt.controller;

import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

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

	@GetMapping
	public String fecthStudent() {
		return "Student fetched successfully!!";
	}

	@PostMapping
	public String createStudent() {
		return "Student added successfuly!";
	}

	@PutMapping
	public String updateStudent() {
		return "Student updated successfully!";
	}

	@DeleteMapping
	public String deleteStudent() {
		return "Student are deleted successfuly!";
	}
}
```

App .java

```
	package com.gpt;

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

```
@SpringBootApplication
public class App {

	public static void main(String[] args) {
		SpringApplication.run(App.class, args);
	}

}
```

application.properties

```
spring.application.name=project1
server.port=9090
server.servlet.context-path=/gpt
```

OUTPUT:
GET->http://localhost:9090/gpt/student
Student fetched successfully!!

POST->http://localhost:9090/gpt/student
Student added successfuly!

PUT->http://localhost:9090/gpt/student
Student updated successfully!

DELETE->http://localhost:9090/gpt/student
Student are deleted successfuly!

20. Create a Spring Boot REST controller with endpoint /greet that returns a welcome message including the current day of the week (e.g., “WELCOME TO TUESDAY SALE!!”).
=>
controller.java

```
package com.gpt.controller;

import java.time.DayOfWeek;
import java.time.LocalDate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/greet")
public class Controller {

	@GetMapping
	public String fecthStudent() {
		LocalDate today=LocalDate.now();
		DayOfWeek day=today.getDayOfWeek();
		return "WELCOME TO"+ day +"SALE!!";
	}

}
```

App.java

```
package com.gpt;

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

@SpringBootApplication
public class App {

	public static void main(String[] args) {
		SpringApplication.run(App.class, args);
	}

}
```

application.properties

```
spring.application.name=project1
server.port=9090
server.servlet.context-path=/gpt
```

➔ **Run The Application:**

- Add '**Spring Web"(**To build REST APIs or run Spring Boot applications.) while creating a maven project using spring initializr .
- Run the App class as a **Java Application** (right-click > Run As > Java Application).

- **Open Postman**
  Download from: https://www.postman.com/downloads

- **Use the correct URL & HTTP Methods**
- Your base URL is:
  http://localhost:9090/gpt/student
- Now use different **HTTP methods** for each operation:

OUTPUT:
GET->http://localhost:9090/gpt/greet
WELCOME TOTUESDAYSALE!!

21.Create a Spring Boot REST application to manage products with the following:

- **ProductDTO with fields: productCode, productName, productVendor, productPrice, productStock.**
- **REST controller with:**
  - **POST /product to add a product (accept JSON)**
  - **GET /product to fetch all products (return JSON)**

=> productDTO.java

```
package com.gpt.dto;

public class ProductDTO {
	private long productCode;
	private String productName;
	private String productVendor;
	private double productPrice;
	private int productStock;

	public ProductDTO() {
		super();

	}

	public ProductDTO(long productCode, String productName, String productVendor, double
productPrice,int productStock) {
		this.productCode = productCode;
		this.productName = productName;
		this.productVendor = productVendor;
		this.productPrice = productPrice;
		this.productStock = productStock;
	}

	public long getProductCode() {
		return productCode;
	}
	public void setProductCode(long productCode) {
		this.productCode = productCode;
	}
	public String getProductName() {
		return productName;
	}
	public void setProductName(String productName) {
		this.productName = productName;
	}
```

```
	public String getProductVendor() {
		return productVendor;
	}
	public void setProductVendor(String productVendor) {
		this.productVendor = productVendor;
	}
	public double getProductPrice() {
		return productPrice;
	}
	public void setProductPrice(double productPrice) {
		this.productPrice = productPrice;
	}
	public int getProductStock() {
		return productStock;
	}
	public void setProductStock(int productStock) {
		this.productStock = productStock;
	}
	@Override
	public String toString() {
		return "ProductDTO [productCode=" + productCode + ", "
				+ (productName != null ? "productName=" + productName + ", " : "")
					+ (productVendor != null ? "productVendor=" +
	productVendor + ", " : "") + "productPrice="+ productPrice + ", productStock=" +
	productStock + "]";
	}
}
```

ProductRepository.java

```
package com.gpt.repo;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Repository;
import com.gpt.dto.ProductDTO;
import jakarta.annotation.PostConstruct;

@Repository
public class ProductRepository {
	List<ProductDTO> product=null;

	@PostConstruct
	public void initializer() {
		ProductDTO productDTO=new ProductDTO();
		productDTO.setProductCode(1001L);
```

```
		productDTO.setProductName("smart phone");
		productDTO.setProductVendor("amazon");
		productDTO.setProductPrice(10000);
		productDTO.setProductStock(25);
		product=new ArrayList<>();
		product.add(productDTO);
	}

	public void insertProduct(ProductDTO productDTO) {
		product.add(productDTO);
	}

	public List<ProductDTO> fetchProduct(){
		return product;
	}
}
```

productService.java

```
package com.gpt.service;

import java.util.List;
import com.gpt.dto.ProductDTO;

public interface ProductService {
	public String insertProduct(ProductDTO productDTO);
	public List<ProductDTO> fetchProduct();
}
```

ProductServiceImpl.java

```
package com.gpt.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.gpt.dto.ProductDTO;
import com.gpt.repo.ProductRepository;
import java.util.List;

@Service
public class ProductServiceImpl implements ProductService {
	@Autowired
	ProductRepository repository;

	@Override
	public String insertProduct(ProductDTO productDTO) {
		repository.insertProduct(productDTO);
```

```
		return "Product with"+productDTO.getProductCode()+"added successfully";
	}

	@Override
	public List<ProductDTO> fetchProduct(){
		return repository.fetchProduct();
	}
}
```

ProductController.java

```
package com.gpt.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.gpt.dto.ProductDTO;
import com.gpt.service.ProductServiceImpl;

@RestController
@RequestMapping("/product")
public class ProductController {
	@Autowired
	private ProductServiceImpl productService;

	@PostMapping(consumes="application/json")
	public ResponseEntity<String> insertProduct(@RequestBody ProductDTO productDTO){
		String response=productService.insertProduct(productDTO);
		return ResponseEntity.ok(response);
	}

	@GetMapping(produces="application/json")
	public List<ProductDTO> fetchProduct(){
		return productService.fetchProduct();
	}

}
```

App.java

```
package com.gpt;
```

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

@SpringBootApplication
public class App {

	public static void main(String[] args) {
		SpringApplication.run(App.class, args);
	}

}
```

application.properties

```
spring.application.name=project1
server.port=9090
server.servlet.context-path=/gpt
```

OUTPUT:

POST->http://localhost:9090/gpt/product

```
{
    "productCode": 103,
    "productName": "Smart Watch ",
    "productVendor": "Acme Inc.",
    "productPrice": 299.99,
    "productStock": 150
}
```

Product with102added successfully

GET->http://localhost:9090/gpt/product

[{"productCode":1001,"productName":"smart phone","productVendor":"amazon","productPrice":10000.0,"productStock":25},{"productCode":102,"productName":"Smart Watch ","productVendor":"Acme Inc.","productPrice":299.99,"productStock":150}]

22.One-to-One Mapping Between Student and Book Using Spring Boot JPA.

App.js

```
package com.infy;

import java.util.Arrays;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.infy.Entity.Book;
import com.infy.Entity.Student;
import com.infy.repo.StudentRepo;

@SpringBootApplication
public class App implements CommandLineRunner {

   @Autowired
   private StudentRepo sr;

   public static void main(String[] args) {
      SpringApplication.run(App.class, args);
   }

   @Override
   public void run(String... args) throws Exception {
      // Create one-to-one mappings
      Student s1 = new Student(101, "John", new Book(1, "learningWithPython", "Python Basics"));
      Student s2 = new Student(102, "Ram", new Book(2, "java", "Java Fundamentals"));
      Student s3 = new Student(103, "Sanvi", new Book(3, "ds", "Data Structures"));
      Student s4 = new Student(104, "Laxmi", new Book(4, "html", "Web Design"));

      sr.saveAll(Arrays.asList(s1, s2, s3, s4));

      List<Student> sList = sr.findAll();
      for (Student s : sList)
         System.out.println(s);
   }
}
```

Book.js

```
package com.infy.Entity;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.OneToOne;

@Entity
public class Book {

   @Id
   private Integer bookId;
   private String title;
   private String bookName;

   @OneToOne(mappedBy = "book") // optional for bidirectional
   private Student student;

   public Book() {}

   public Book(Integer bookId, String title, String bookName) {
      this.bookId = bookId;
      this.title = title;
      this.bookName = bookName;
   }

   public Integer getBookId() {
      return bookId;
   }

   public void setBookId(Integer bookId) {
      this.bookId = bookId;
   }

   public String getTitle() {
      return title;
   }

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

   public String getBookName() {
      return bookName;
   }
```

```
    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    public Student getStudent() {
        return student;
    }

    public void setStudent(Student student) {
        this.student = student;
    }

    @Override
    public String toString() {
        return "Book [bookId=" + bookId + ", title=" + title + ", bookName=" + bookName + "]";
    }
}
```

Student.js

```
package com.infy.Entity;


import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.OneToOne;

@Entity
public class Student {
    @Id
    private Integer studentId;
    private String studentName;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "book_id") // foreign key column in student table
    private Book book;

    public Student() {}

    public Student(Integer studentId, String studentName, Book book) {
        this.studentId = studentId;
        this.studentName = studentName;
        this.book = book;
```

```
    }

    public Integer getStudentId() {
        return studentId;
    }

    public void setStudentId(Integer studentId) {
        this.studentId = studentId;
    }

    public String getStudentName() {
        return studentName;
    }

    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }

    public Book getBook() {
        return book;
    }

    public void setBook(Book book) {
        this.book = book;
    }

    @Override
    public String toString() {
        return "Student [studentId=" + studentId + ", studentName=" + studentName + ", book=" + book
+ "]";
    }
}
```

BookRepo.js

```
package com.infy.repo;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.infy.Entity.Book;

@Repository
public interface BookRepo extends  JpaRepository<Book,Integer>{
}
```

StudentRepo.js

```
package com.infy.repo;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.infy.Entity.Student;

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

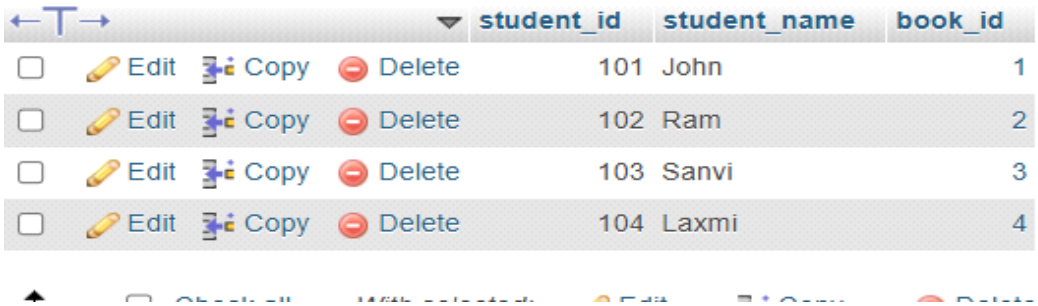
Application.properties

```
spring.application.name=OneToOne
spring.datasource.url=jdbc:mysql://localhost:3306/stdBook
spring.datasource.username=root
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql: true
```

➔ **Run The Application:**

- Open xampp , start Apache and MySql server .
- Create a database named stdBook in MySQL.
- Add 'Spring Data Jpa" (To work with JPA repositories and database) and "MySql Driver" dependencies while creating a maven project using spring initializr .
- Run the App class as a **Java Application** (right-click > Run As > Java Application).

OUTPUT:



	student_id	student_name	book_id
Edit Copy Delete	101	John	1
Edit Copy Delete	102	Ram	2
Edit Copy Delete	103	Sanvi	3
Edit Copy Delete	104	Laxmi	4

	book_id	book_name	title
Edit Copy Delete	1	Python Basics	learningWithPython
Edit Copy Delete	2	Java Fundamentals	java
Edit Copy Delete	3	Data Structures	ds
Edit Copy Delete	4	Web Design	html

Check all With selected: Edit Copy Delete Export

23. Many-to-one Mapping Between Student and Book Using Spring Boot JPA

App.js

```
package com.infy;


import java.util.Arrays;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.infy.Entity.Book;
import com.infy.Entity.Student;
import com.infy.repo.StudentRepo;

@SpringBootApplication
public class App implements CommandLineRunner {

   @Autowired
   private StudentRepo sr;

   public static void main(String[] args) {
      SpringApplication.run(App.class, args);
   }

   @Override
   public void run(String... args) throws Exception {
      // Create Book entities
           Book b1 = new Book(1, "learningWithPython", "Python Basics");
           Book b2 = new Book(2, "java", "Java Fundamentals");
           Book b3 = new Book(3, "ds", "Data Structures");

      // Create students and associate books
      Student s1 = new Student(101, "john", b1);
      Student s2 = new Student(102, "ram", b2);
      Student s3 = new Student(103, "sanvi",b1);
      Student s4 = new Student(104, "laxmi",b3);

      // Save students (books will be saved automatically because of cascade)
      sr.saveAll(Arrays.asList(s1, s2, s3, s4));

      // Fetch and print all students with their books
      List<Student> sList = sr.findAll();
      for (Student s : sList)
```

```
        System.out.println(s);
    }
}
```

Student.js

```
package com.infy.Entity;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
@Entity
public class Student {
    @Id
    private Integer studentId;
    private String studentName;
    // One student can have many books
    @ManyToOne( cascade = { CascadeType.ALL})
    private Book book;

    public Student() {
        super();
    }
    public Student(Integer studentId, String studentName, Book book) {
        super();
        this.studentId=studentId;
        this.studentName = studentName;
        this.book=book;
    }
    public Integer getStudentId() {
        return studentId;
    }
    public void setStudentId(Integer studentId) {
        this.studentId = studentId;
    }
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }

    public Book getBook() {
        return book;
```

```
    }

    public void setBook(Book book) {
        this.book = book;
    }

    @Override
    public String toString() {
        return "Student [StudentId=" + studentId + ", StudentName=" + studentName + ", book=" + book
+ "]";
    }
}
```

Book.js

```
package com.infy.Entity;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
@Entity
public class Book {
   @Id
    private Integer bookId;
    private String title;
    private String bookName;

    public Book() {
        // Default constructor
    }
    public Book(Integer bookId, String title, String bookName) {
        this.bookId = bookId;
        this.title = title;
        this.bookName = bookName;


    // Getters and setters
    public Integer getBookId() {
        return bookId;
    }
    public void setBookId(Integer bookId) {
        this.bookId = bookId;
    }

    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
```

```
        this.title = title;
    }
    public String getBookName() {
        return bookName;
    }
    public void setBookName(String bookName) {
        this.bookName = bookName;
    }
    @Override
    public String toString() {
        return "Book [bookId=" + bookId + ", title=" + title + ", bookName=" + bookName + "]";
    }
}
```

BookRepo.js

```
package com.infy.repo;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.infy.Entity.Book;

@Repository
public interface BookRepo extends  JpaRepository<Book,Integer>{
}
```

StudentRepo.js

```
package com.infy.repo;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.infy.Entity.Student;

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

Application.properties

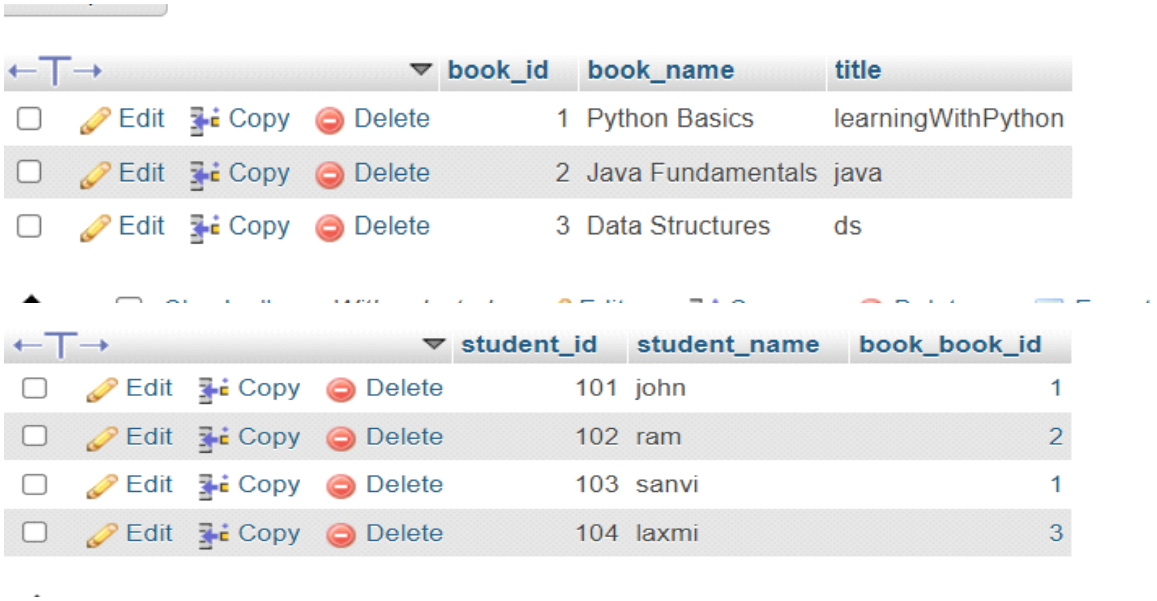
```
spring.application.name=ManyToOne
spring.datasource.url=jdbc:mysql://localhost:3306/stdBook
spring.datasource.username=root
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql: true
```

➔ **Run The Application:**

- Open xampp , start Apache and MySql server .
- Create a database named stdBook in MySQL.

- Add ‘Spring Data Jpa” (To work with JPA repositories and database) and “MySql Driver” dependencies while creating a maven project using spring initializr .
- Run the App class as a **Java Application** (right-click > Run As > Java Application).

OUTPUT:



	book_id	book_name	title
Edit Copy Delete	1	Python Basics	learningWithPython
Edit Copy Delete	2	Java Fundamentals	java
Edit Copy Delete	3	Data Structures	ds

	student_id	student_name	book_book_id
Edit Copy Delete	101	john	1
Edit Copy Delete	102	ram	2
Edit Copy Delete	103	sanvi	1
Edit Copy Delete	104	laxmi	3

24. Many-to-Many Mapping Between Student and Book Using Spring Boot JPA.

App.js

```
package com.infy;

import java.util.Arrays;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.infy.Entity.Book;
import com.infy.Entity.Student;
import com.infy.repo.StudentRepo;

@SpringBootApplication
public class App implements CommandLineRunner {

   @Autowired
   private StudentRepo sr;

   public static void main(String[] args) {
      SpringApplication.run(App.class, args);
   }

   @Override
   public void run(String... args) throws Exception {
      Book b1 = new Book(1, "learningWithPython", "Python Basics");
      Book b2 = new Book(2, "java", "Java Fundamentals");
      Book b3 = new Book(3, "ds", "Data Structures");

      Student s1 = new Student(101, "John", Arrays.asList(b1, b2));
      Student s2 = new Student(102, "Ram", Arrays.asList(b2, b3));
      Student s3 = new Student(103, "Sanvi", Arrays.asList(b1));
      Student s4 = new Student(104, "Laxmi", Arrays.asList(b1, b2, b3));

      sr.saveAll(Arrays.asList(s1, s2, s3, s4));

      List<Student> sList = sr.findAll();
      for (Student s : sList) {
         System.out.println(s);
      }
   }
}
```

Book.js

```
package com.infy.Entity;

import java.util.List;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;

@Entity
public class Book {
   @Id
   private Integer bookId;
   private String title;
   private String bookName;

  @ManyToMany(mappedBy = "books")
   private List<Student> students;
   public Book() {}

   public Book(Integer bookId, String title, String bookName) {
      this.bookId = bookId;
      this.title = title;
      this.bookName = bookName;
   }

   public Integer getBookId() {
      return bookId;
   }
   public void setBookId(Integer bookId) {
      this.bookId = bookId;
   }

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

   public String getBookName() {
      return bookName;
   }
   public void setBookName(String bookName) {
      this.bookName = bookName;
```

```
    }

    public List<Student> getStudents() {
        return students;
    }
    public void setStudents(List<Student> students) {
        this.students = students;
    }
    @Override
    public String toString() {
        return "Book [bookId=" + bookId + ", title=" + title + ", bookName=" + bookName + "]";
    }
}
```

Student.js

```
package com.infy.Entity;

import java.util.List;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinTable;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToMany;

@Entity
public class Student {
    @Id
    private Integer studentId;
    private String studentName;

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(
        name = "student_book",
        joinColumns = @JoinColumn(name = "student_id"),
        inverseJoinColumns = @JoinColumn(name = "book_id")
    )
    private List<Book> books;

    public Student() {}

    public Student(Integer studentId, String studentName, List<Book> books) {
        this.studentId = studentId;
        this.studentName = studentName;
        this.books = books;
```

```
    }

    public Integer getStudentId() {
        return studentId;
    }
    public void setStudentId(Integer studentId) {
        this.studentId = studentId;
    }

    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }

    public List<Book> getBooks() {
        return books;
    }
    public void setBooks(List<Book> books) {
        this.books = books;
    }

    @Override
    public String toString() {
        return "Student [studentId=" + studentId + ", studentName=" + studentName + ", books=" +
books + "]";
    }
}
```

BookRepo.js

```
package com.infy.repo;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.infy.Entity.Book;

@Repository
public interface BookRepo extends  JpaRepository<Book,Integer>{
}
```

StudentRepo.js

```
package com.infy.repo;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.stereotype.Repository;
import com.infy.Entity.Student;

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

Application.properties

```
spring.application.name=ManyToMany
spring.datasource.url=jdbc:mysql://localhost:3306/stdBook
spring.datasource.username=root
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql: true
```

➔ **Run The Application:**

- Open xampp , start Apache and MySql server .
- Create a database named stdBook in MySQL.
- Add 'Spring Data Jpa" (To work with JPA repositories and database) and "MySql Driver" dependencies while creating a maven project using spring initializr .
- Run the App class as a **Java Application** (right-click > Run As > Java Application).

OUTPUT:

		student_id	student_name	book_book_id
Edit Copy Delete		101	john	1
Edit Copy Delete		102	ram	2
Edit Copy Delete		103	sanvi	1
Edit Copy Delete		104	laxmi	3

student_id	book_id
101	1
101	2
102	2
102	3
103	1
104	1
104	2
104	3

25. Create a Spring Boot app to manage cab bookings with fields: cabId, cabName, cabPhoneNo, and address. Design the entity class, JPA repository, and REST controller for basic CRUD operations.

CabBooking.java

```
package com.infy.domain;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;


@Entity
public class CabBooking{
	@Id
	private Long cabPhoneNo;
	private String cabName;
	private int  cabId;
	private String address;
	public CabBooking() {
		super();
		// TODO Auto-generated constructor stub
	}
	public CabBooking(int cabId, String cabName, long cabPhoneNo, String address) {
		super();
		this.cabId = cabId;
		this.cabName = cabName;
		this.cabPhoneNo = cabPhoneNo;
		this.address = address;
	}
	public int getCabId() {
		return cabId;
	}
	public void setCabId(int cabId) {
		this.cabId = cabId;
	}
	public String getCabName() {
		return cabName;
	}
	public void setCabName(String cabName) {
		this.cabName = cabName;
	}
	public long getCabPhoneNo() {
		return cabPhoneNo;
	}
	public void setCabPhoneNo(long cabPhoneNo) {
```

```
		this.cabPhoneNo = cabPhoneNo;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	@Override
	public String toString() {
		return "CabBoking [cabId=" + cabId + ", cabName=" + cabName + ", cabPhoneNo=" +
cabPhoneNo + ", address="
				+ address + "]";
	}



}
```

CabBookingDTO.java

```
package com.infy.dto;

import com.infy.domain.CabBooking;

public class CabBookingDTO {
	private Long cabPhoneNo;
	private String cabName;
	private int  cabId;
	private String address;
	public CabBookingDTO() {
		super();
		// TODO Auto-generated constructor stub
	}
	public CabBookingDTO(int cabId, String cabName, long cabPhoneNo, String address) {
		super();
		this.cabId = cabId;
		this.cabName = cabName;
		this.cabPhoneNo = cabPhoneNo;
		this.address = address;
	}
	public int getCabId() {
		return cabId;
	}
	public void setCabId(int cabId) {
		this.cabId = cabId;
```

```
	}
	public String getCabName() {
		return cabName;
	}
	public void setCabName(String cabName) {
		this.cabName = cabName;
	}
	public long getCabPhoneNo() {
		return cabPhoneNo;
	}
	public void setCabPhoneNo(long cabPhoneNo) {
		this.cabPhoneNo = cabPhoneNo;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}

	public static CabBooking pre(CabBookingDTO cabBookingDTO) {
		CabBooking c=new CabBooking();
		c.setCabId(cabBookingDTO.getCabId());
		c.setCabName(cabBookingDTO.getCabName());
		c.setAddress(cabBookingDTO.getAddress());
		c.setCabPhoneNo(cabBookingDTO.getCabPhoneNo());

		return c;
	}
}
```

CabBookingRepository.java

```
package com.infy.repo;

import com.infy.domain.CabBooking;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface CabBookingRepository  extends CrudRepository<CabBooking, Long> {
}
```

CabBookingService.java

```
package com.infy.service;
import java.util.List;
import com.infy.domain.CabBooking;
```

```
import com.infy.dto.CabBookingDTO;
public interface CabBookingService {
	public String insert(CabBookingDTO cabBookingDTO);
	public List<CabBooking> fetch(long cabPhoneNO);
	public String cancel(int cabId);
}
```

CabBookingServiceImpl.java

```
package com.infy.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import com.infy.domain.CabBooking;
import com.infy.dto.CabBookingDTO;
import com.infy.repo.CabBookingRepo;

@Service
public class CabBookingServiceImpl implements CabBookingService{
	@Autowired
	CabBookingRepository  r;

	public String insert(CabBookingDTO cabBookingDTO) {
		CabBooking c=new CabBooking();
		c=CabBookingDTO.pre(cabBookingDTO);
		r.save(c);
		return "cab with"+cabBookingDTO.getCabId()+"added";
	}
	public List<CabBooking> fetch(long cabPhoneNo){
		return (List<CabBooking>) r.findAll();
	}
	public String cancel(int cabId) {
		r.deleteById((long) cabId);
		return "deleted";
	}
}
```

Controller.java

```
package com.infy.controller;

import org.springframework.core.env.Environment;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.infy.domain.CabBooking;
import com.infy.dto.CabBookingDTO;
import com.infy.service.CabBookingServiceImpl;

@RestController
@RequestMapping("/cab")
public class Controller {

	@Autowired
	CabBookingServiceImpl s;

	@Autowired
	Environment evn;

	@PostMapping
	public ResponseEntity<String> bookCab(@RequestBody CabBookingDTO cabBookingDTO){
		String add=(String) s.insert(cabBookingDTO);
		String r=evn.getProperty("message1")+" "+add;
		return new ResponseEntity<>(r,HttpStatus.CREATED);
	}


	@GetMapping("/{cabPhoneNo}")
  public ResponseEntity<List<CabBooking>> getBookingDetails(@PathVariable Long cabPhoneNo) {
    List<CabBooking> booking = s.fetch(cabPhoneNo);
    return new ResponseEntity<>(booking, HttpStatus.OK);
  }
	@DeleteMapping("/{cabId}")
	 public ResponseEntity<String> cancelBooking(@PathVariable int cabId) {
    s.cancel(cabId);
    String msg = evn.getProperty("message2");
    return new ResponseEntity<>(msg, HttpStatus.OK);
  }
}
```

App.java

```
package com.infy;
```

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

@SpringBootApplication
public class App {

	public static void main(String[] args) {
		SpringApplication.run(App.class, args);
	}

}
```

application.properties

```
spring.application.name=CabBooking

spring.application.name=my-spring-boot-app
spring.datasource.url=jdbc:mysql://localhost:3306/cab
spring.datasource.username=root

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

server.port=9090
server.servlet.context-path=/gpt
message1=Cab booked successfully
message2=cab is successfully canceled
```

➔ **Run The Application:**

- Open xampp , start Apache and MySql server .
- Create a database named cab in MySQL.
- Add 'Spring Data Jpa" (To work with JPA repositories and database) , "MySql Driver" and "Spring Web**"(**To build REST APIs or run Spring Boot applications.) dependencies while creating a maven project using spring initializr .
- Run the App class as a **Java Application** (right-click > Run As > Java Application).
- **Open Postman**
  Download from: https://www.postman.com/downloads

- **Use the correct URL & HTTP Methods**
- Your base URL is:
  http://localhost:9090/gpt/cab/124
- Now use different **HTTP methods** for each operation:

OUTPUT:

GET->http://localhost:9090/gpt/cab/124


```
[
  {
    "cabPhoneNo": 123,
    "cabName": "City Cabs",
    "cabId": 101,
    "address": "123 Main Street, Springfield"
  },
  {
    "cabPhoneNo": 124,
    "cabName": "State Cabs",
    "cabId": 102,
    "address": "123 Main Street, Springfield"
  }
]
```


POST->http://localhost:9090/gpt/cab


```
{
 "cabPhoneNo": 124,
 "cabName": "State Cabs",
 "cabId": 102,
 "address": "123 Main Street, Springfield"
}
```


Cab booked successfully cab with102added

DELETE->http://localhost:9090/gpt/cab/102

cab is successfully canceled

26.Create a Spring Boot app managing Student details with MongoDB which Supports CRUD operations via REST endpoints.

Student.java

```
package com.gpt.domain;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;
import com.gpt.dto.StudentDTO;

@Document(collection = "student")
public class Student {
   @Id
   private Integer studentId;
   private String studentName;
   private String studentCourse;

   public Student() {}

   public Student(Integer studentId, String studentName, String studentCourse) {
      this.studentId = studentId;
      this.studentName = studentName;
      this.studentCourse = studentCourse;
   }

   // Getters and Setters
   public Integer getStudentId() {
      return studentId;
   }
   public void setStudentId(Integer studentId) {
      this.studentId = studentId;
   }
   public String getStudentName() {
      return studentName;
   }
   public void setStudentName(String studentName) {
      this.studentName = studentName;
   }
   public String getStudentCourse() {
      return studentCourse;
   }
```

```
    public void setStudentCourse(String studentCourse) {
        this.studentCourse = studentCourse;
    }

    public static StudentDTO convertEntityToDto(Student student) {
        return new StudentDTO(
            student.getStudentId(),
            student.getStudentName(),
            student.getStudentCourse()
        );
    }

    @Override
    public String toString() {
        return "Student [studentId=" + studentId +
                ", studentName=" + studentName +
                ", studentCourse=" + studentCourse + "]";
    }
}
```

StudentDTO.java

```
package com.gpt.dto;

import com.gpt.domain.Student;

public class StudentDTO {
    private Integer studentId;
    private String studentName;
    private String studentCourse;

    public StudentDTO() {}

    public StudentDTO(Integer studentId, String studentName, String studentCourse) {
        this.studentId = studentId;
        this.studentName = studentName;
        this.studentCourse = studentCourse;
    }

    // Getters and Setters
    public Integer getStudentId() {
        return studentId;
```

```
    }
    public void setStudentId(Integer studentId) {
        this.studentId = studentId;
    }
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }
    public String getStudentCourse() {
        return studentCourse;
    }
    public void setStudentCourse(String studentCourse) {
        this.studentCourse = studentCourse;
    }

    public static Student convertDtoToEntity(StudentDTO studentDTO) {
        return new Student(
            studentDTO.getStudentId(),
            studentDTO.getStudentName(),
            studentDTO.getStudentCourse()
        );
    }
}
```

StudentRepository.java

```
package com.gpt.repository;
import org.springframework.data.mongodb.repository.MongoRepository;
import com.gpt.domain.Student;

public interface StudentRepository extends MongoRepository<Student, Integer> {
}
```

StudentService.java

```
package com.gpt.service;

import java.util.List;
import com.gpt.dto.StudentDTO;

public interface StudentService {
```

```
    String insertRecord(StudentDTO studentDTO);
    List<StudentDTO> getAllRecord();
    String deleteRecord(Integer studentId);
    String updateRecord(Integer studentId, String newName);
}
```

StudentServiceImpl.java

```
package com.gpt.service;

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.gpt.domain.Student;
import com.gpt.dto.StudentDTO;
import com.gpt.repository.StudentRepository;

@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    private StudentRepository studentRepository;

    @Override
    public String insertRecord(StudentDTO studentDTO) {
        studentRepository.save(StudentDTO.convertDtoToEntity(studentDTO));
        return "Student with ID " + studentDTO.getStudentId() + " added successfully!";
    }

    @Override
    public List<StudentDTO> getAllRecord() {
        List<Student> students = studentRepository.findAll();
        List<StudentDTO> dtoList = new ArrayList<>();

        for (Student student : students) {
            dtoList.add(Student.convertEntityToDto(student));
        }
        return dtoList;
```

```
    }

    @Override

        public String deleteRecord(Integer studentId) {
        studentRepository.deleteById(studentId);
                return "student with"+studentId+"deleted";
        }

        public String updateRecord(Integer  studentId,String newName) {
                Optional<Student> optional = studentRepository.findById(studentId);
                Student std=optional.get();
                std.setStudentName(newName);
                studentRepository.save(std);
                                return "The customer with id"+studentId+"has been updated
successfully";
        }

}
```

StudentController.java

```
package com.gpt.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import com.gpt.dto.StudentDTO;
import com.gpt.service.StudentService;

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

    @Autowired
    private StudentService studentService;

    @PostMapping
    public String addStudent(@RequestBody StudentDTO studentDTO) {
      return studentService.insertRecord(studentDTO);
```

```
    }

    @GetMapping
    public List<StudentDTO> getAllStudents() {
        return studentService.getAllRecord();
    }

    @DeleteMapping("/{studentId}")
    public String deleteStudent(@PathVariable Integer studentId) {
        return studentService.deleteRecord(studentId);
    }

    @PutMapping("/{studentId}")
    public String updateStudentName(@PathVariable Integer studentId, @RequestBody String
newName) {
        return studentService.updateRecord(studentId, newName);
    }
}
```

App.java

```
package com.infy;

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

@SpringBootApplication
public class App {

        public static void main(String[] args) {
                SpringApplication.run(AssociationApplication.class, args);
        }
}
```

application.properties

```
spring.application.name=demo1

spring.data.mongodb.uri=mongodb://localhost:27017/mongoDB

server.port=9090

server.servlet.context-path=/gpt
```

➔ **Run The Application:**

- **1. Install MongoDB (if not already installed)**
  Make sure MongoDB is installed and running locally on default port 27017.
- To check:
  mongod
- Or for MongoDB Compass users – just ensure the service is running.

- **2. Create MongoDB Database (optional)**
- MongoDB will **auto-create** the mongoDB database and student collection when you insert a document.
- So no manual setup needed unless you want to inspect via Compass or shell.

- Add "Spring Data MongoDB" dependencies while creating a maven project using spring initializr .
- Run the App class as a **Java Application** (right-click > Run As > Java Application).

- **Open Postman**
  Download from: https://www.postman.com/downloads

- **Use the correct URL & HTTP Methods**
- Your base URL is:
  http://localhost:9090/gpt/student
- Now use different **HTTP methods** for each operation:

OUTPUT:

POST->http://localhost:9090/gpt/student

{

"studentId": 101,

"studentName": "ssc",

"studentCourse": "Computer Science"

}

Student with ID 101 added successfully!

GET->http://localhost:9090/gpt/student

[{"studentId":103,"studentName":"Jack","studentCourse":"Computer Science"},{"studentId":102,"studentName":"Jammes","studentCourse":"Computer Science"},{"studentId":101,"studentName":"ssc","studentCourse":"Computer Science"}]

DELETE->http://localhost:9090/gpt/student/101

student with101deleted

PUT->http://localhost:9090/gpt/student/102/Alice

The customer with id101has been updated successfully.

Now , go to MongoDB database and see the changes!..