Employee ID: 11949
Employee Name: Muthuraj Periyasamy

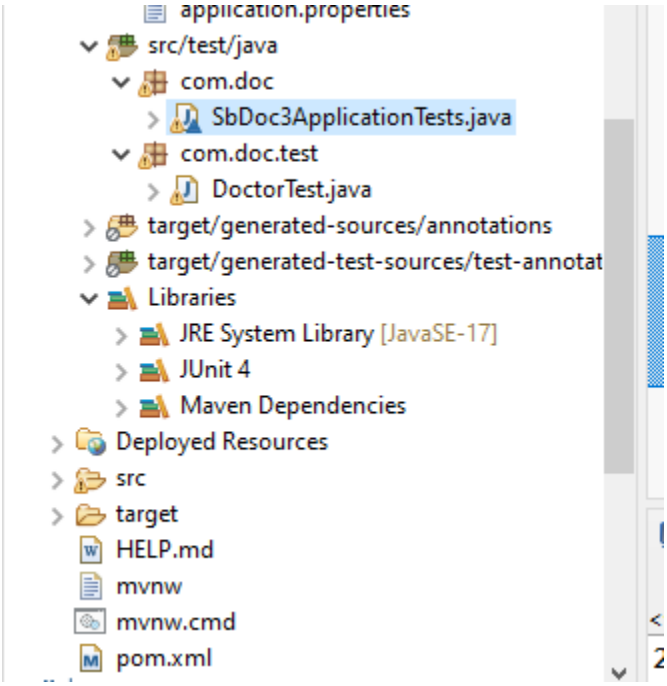
Date:12-01-2023

- Use your previous [Angular, Spring Boot & Microservices] lab assessment [dated 19-Dec-2023] project / application to write the following test cases.

White Box Testing (Unit Testing) : [2 + 2 = 4 Hours]

- Write the Unit Test cases by using JUnit for your Back-End [Spring Boot].
- Write the Unit Test cases by using Jasmine & Karma for your Front-End [Angular].

1.Write the Unit Test cases by using JUnit for your Back-End [Spring Boot]



Package name - com.doc.test
Class name - DoctorTest

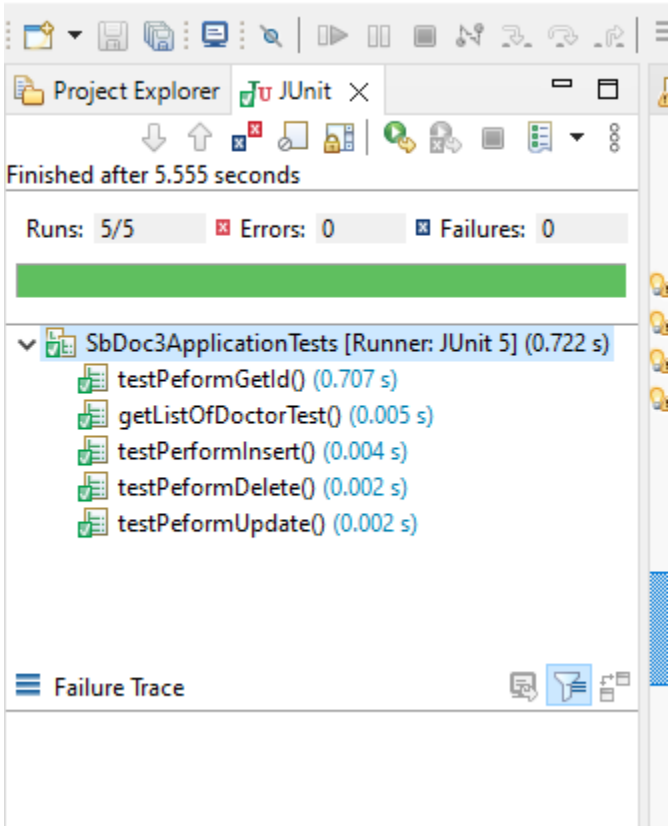
```
package com.doc.test;
import static org.junit.Assert.*;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import org.junit.Test;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.doc.bean.Doctor;
import com.doc.controller.DOCController;
import com.doc.dao.DoctorDAO;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import org.springframework.test.annotation.Rollback;
import org.assertj.core.api.Assertions;
import org.junit.jupiter.api.MethodOrderer;
import org.junit.jupiter.api.Order;
@SpringBootApplication
public class DoctorTest {

	private DOCController controller;
	private Doctor doctor;
	private DoctorDAO doctordao;


  @Test
  @Order(1)
  @Rollback(value = false)
  public void testPerformInsert(){
		Doctor obj=new Doctor();
		obj.setdId(1);
		obj.setdName("raja");
		obj.setdAge(23);
		obj.setdEmail("raja@gmail.com");
	    obj.setdSpecialization("Baby");
	    obj.setdPhno(678976);
	    obj.setdLocation("chennai");
	    obj=controller.performInsert(doctor);
	    org.junit.Assert.assertEquals(obj, doctor);
```

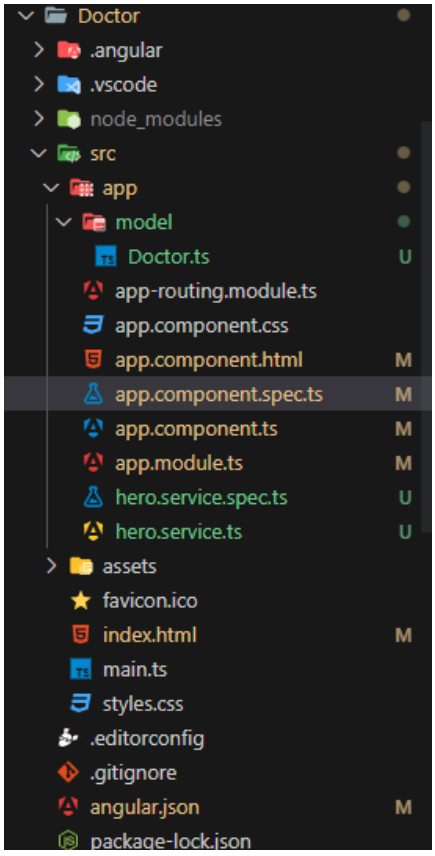
```
  }
@After
@Order(2)
@Rollback(value = false)
public void testPerformUpdate(){
                Doctor obj=new Doctor();
        obj.setdId(1);
        obj.setdName("kannan");
        obj.setdAge(23);
        obj.setdEmail("kannan@gmail.com");
  obj.setdSpecialization("Baby");
  obj.setdPhno(678976);
  obj.setdLocation("chennai");
  obj=controller.performInsert(doctor);
  org.junit.Assert.assertEquals(obj, doctor);
}
@Test
@Order(3)
@Rollback(value = false)
public void testPeformDelete(){
        Doctor obj=new Doctor();
  Doctor doctor = null;
  Optional<Doctor> optionalDoctor = doctor();
  if(optionalDoctor.isPresent()){
        doctor = optionalDoctor.get();
  }
  Assertions.assertThat(doctor).isNull();
}
@Test
@Order(4)
public void testPeformGetId(){
        Doctor obj=new Doctor();
  Assertions.assertThat(obj.getdId()).isEqualTo(1L);
}
@Test
@Order(5)
public void getListOfDoctorTest(){
        List<Doctor> list = new ArrayList<Doctor>();
  Assertions.assertThat(list.size()).isGreaterThan(0);
}
```

```
private Optional<Doctor> doctor() {
        // TODO Auto-generated method stub
        return null;
}
}
```

Output:



2.Write the Unit Test Cases by using Jasmine and Karma For you Front-End[Spring Boot]




```
import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { AppRoutingModule } from './app-routing.module';
import { BrowserModule } from '@angular/platform-browser';
import { Doctor } from './model/Doctor';

describe('InsertComponent ',()=>{


  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
      imports: [
        BrowserModule,
```

```
        AppRoutingModule,
        ReactiveFormsModule,
        FormsModule,
        HttpClientModule,

      ]
    }).compileComponents();
  });

  it('App Component Test',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const app =  fixture.nativeElement as HTMLElement;
    expect(app).toBeTruthy();
  })
  it('App Component GUI count',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const app =  fixture.nativeElement as HTMLElement;
    const inputElement = app.querySelectorAll("input");
    const buttonElement = app.querySelectorAll("button");
    expect(inputElement.length).toEqual(7);
    expect(buttonElement.length).toEqual(3);
  })
  it('Test Form Valid',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const Doctor =  fixture.componentInstance;
    Doctor.myForm.controls?.['dId'].setValue("1");
      Doctor.myForm.controls?.['dName'].setValue("raja");
      Doctor.myForm.controls?.['dAge'].setValue("23");
      Doctor.myForm.controls?.['dEmail'].setValue("raj@gmail.com");
      Doctor.myForm.controls?.['dSpecialization'].setValue("Baby");
      Doctor.myForm.controls?.['dPhno'].setValue("678976");
      Doctor.myForm.controls?.['dLocation'].setValue("chennai");

    expect(Doctor.doctor).toBeTruthy();
  })
});
describe('UpdateComponent ',()=>{

  beforeEach(async () => {
    await TestBed.configureTestingModule({
```

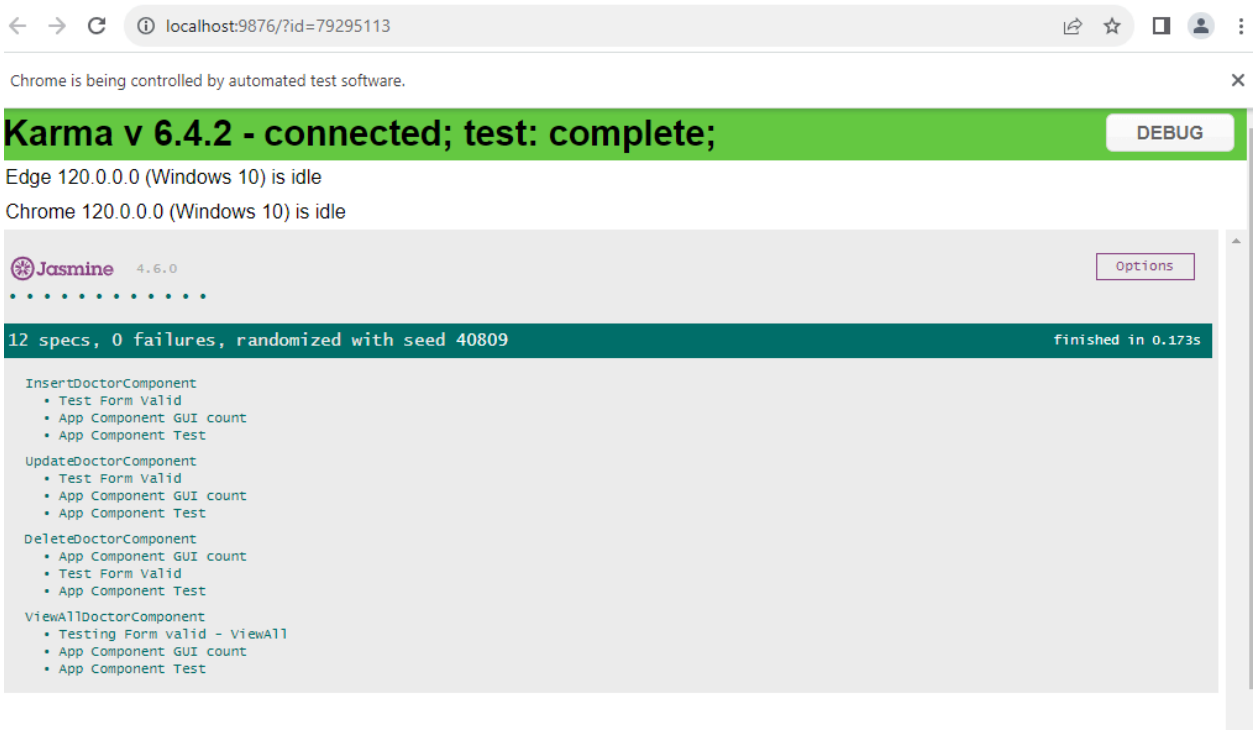
```
    declarations: [
      AppComponent
    ],
    imports: [
      BrowserModule,
      AppRoutingModule,
      ReactiveFormsModule,
      FormsModule,
      HttpClientModule,

    ]
  }).compileComponents();
});

it('App Component Test',() =>{
  const fixture = TestBed.createComponent(AppComponent);
  const app =  fixture.nativeElement as HTMLElement;
  expect(app).toBeTruthy();
})

it('App Component GUI count',() =>{
  const fixture = TestBed.createComponent(AppComponent);
  const app =  fixture.nativeElement as HTMLElement;
  const inputElement = app.querySelectorAll("input");
  const buttonElement = app.querySelectorAll("button");
  expect(inputElement.length).toEqual(7);
  expect(buttonElement.length).toEqual(3);
})
it('Test Form Valid',() =>{
  const fixture = TestBed.createComponent(AppComponent);
  const Doctor =  fixture.componentInstance;
  Doctor.myForm.controls?.['dId'].setValue("1");
    Doctor.myForm.controls?.['dName'].setValue("Kannan");
    Doctor.myForm.controls?.['dAge'].setValue("23");
    Doctor.myForm.controls?.['dEmail'].setValue("kanna@gmail.com");
    Doctor.myForm.controls?.['dSpecialization'].setValue("Baby");
    Doctor.myForm.controls?.['dPhno'].setValue("78965");
    Doctor.myForm.controls?.['dLocation'].setValue("chennai");

  expect(Doctor.doctor).toBeTruthy();
```

```
  })
});
describe('DeleteComponent ',()=>{

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
      imports: [
        BrowserModule,
        AppRoutingModule,
        ReactiveFormsModule,
        FormsModule,
        HttpClientModule,

      ]
    }).compileComponents();
  });

  it('App Component Test',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const app =  fixture.nativeElement as HTMLElement;
    expect(app).toBeTruthy();
  })

  it('App Component GUI count',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const app =  fixture.nativeElement as HTMLElement;
    const inputElement = app.querySelectorAll("input");
    const buttonElement = app.querySelectorAll("button");
    expect(inputElement.length).toEqual(7);
    expect(buttonElement.length).toEqual(3);
  })
  it('Test Form Valid',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const Doctor =  fixture.componentInstance;
    Doctor.myForm.controls?.['dId'].setValue("1");
      Doctor.myForm.controls?.['dName'].setValue("Kannan");
      Doctor.myForm.controls?.['dAge'].setValue("23");
```

```
      Doctor.myForm.controls?.['dEmail'].setValue("kanna@gmail.com");
      Doctor.myForm.controls?.['dSpecialization'].setValue("Baby");
      Doctor.myForm.controls?.['dPhno'].setValue("78965");
      Doctor.myForm.controls?.['dLocation'].setValue("chennai");

    expect(Doctor.doctor).toBeTruthy();
  })
});
describe('ViewAllComponent ',()=>{
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
      imports: [
        BrowserModule,
        AppRoutingModule,
        ReactiveFormsModule,
        FormsModule,
        HttpClientModule,

      ]
    }).compileComponents();
  });
  it('App Component Test',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const app =  fixture.nativeElement as HTMLElement;
    expect(app).toBeTruthy();
  })
  it('App Component GUI count',() =>{
    const fixture = TestBed.createComponent(AppComponent);
    const app =  fixture.nativeElement as HTMLElement;
    const inputElement = app.querySelectorAll("input");
    const buttonElement = app.querySelectorAll("button");
    expect(inputElement.length).toEqual(7);
    expect(buttonElement.length).toEqual(3);
  })
  it('Testing Form valid - ViewAll', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const Laptop = fixture.componentInstance;
```

```
    expect(Laptop.myForm.valid).toBeFalsy();
  });
});
```

Output:




```
TOTAL: 24 SUCCESS
✓ Browser application bundle generation complete.
Chrome 120.0.0.0 (Windows 10): Executed 12 of 12 SUCCESS (0.234 secs / 0.212 secs)
Edge 120.0.0.0 (Windows 10): Executed 12 of 12 SUCCESS (0.69 secs / 0.603 secs)
TOTAL: 24 SUCCESS
```

2.Black Box Testing:

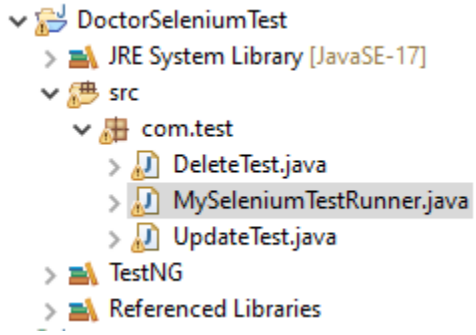
Functional Testing (Manual Testing): [1 Hour]

- Write the Manual Test Case for the Front-End of your application.

****

3.Functional Testing (Automated Testing): [2 Hours]

- Write the Automated Test cases by using Selenium for your application. And generate the test report using TestNG.



1.MySeleniumTestRunner - InsertedClass:

```
package com.test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class MySeleniumTestRunner {
private WebDriver driver;

  @BeforeTest
  public void initializer() {
     System.setProperty("webdriver.Chrome.driver", "driver/Chromedriver.exe");
     driver = new ChromeDriver();
  }
  @Test
  public void site() {
     driver.navigate().to("http://localhost:4200/");
  }
  @Test
  public void testdoctorFormSubmission() {
```

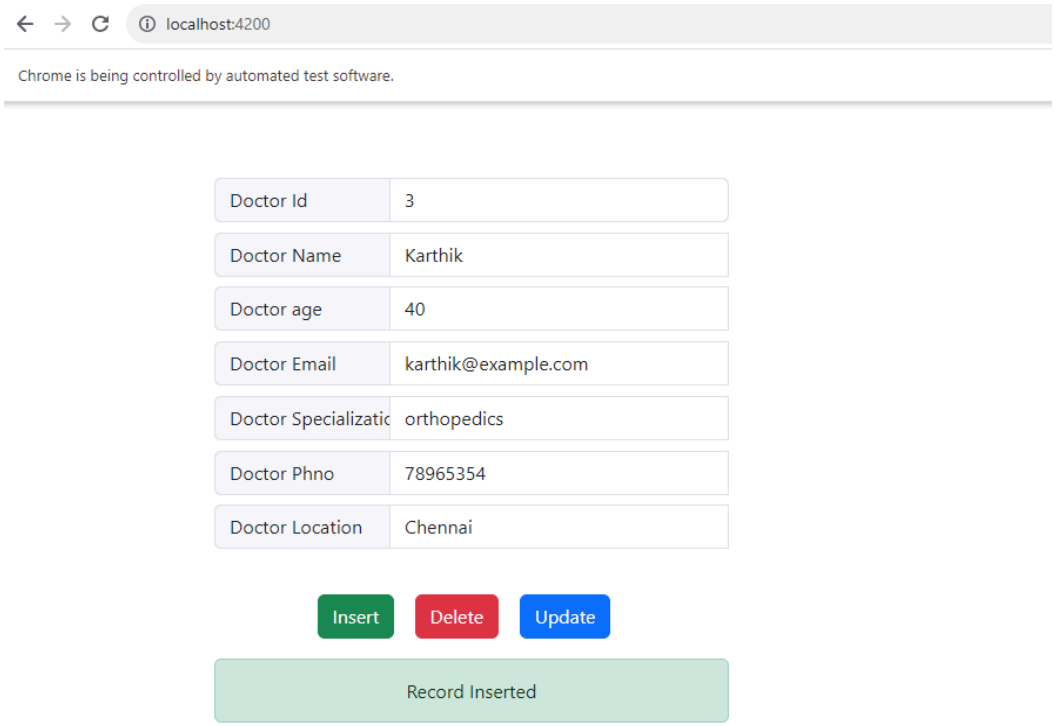
```
 WebElement doctorIdField = driver.findElement(By.name("id"));
    doctorIdField.sendKeys("3");
    WebElement doctorNameField = driver.findElement(By.name("name"));
    doctorNameField.sendKeys("Karthik");

    WebElement doctorAgeField = driver.findElement(By.name("age"));
    doctorNameField.sendKeys("40");

    WebElement doctorEmailField = driver.findElement(By.name("email"));
    doctorEmailField.sendKeys("karthik@example.com");
    WebElement doctorSpecializationField = driver.findElement(By.name("spl"));
    doctorSpecializationField.sendKeys("orthopedics");
    WebElement doctorPhnoField = driver.findElement(By.name("phn"));
    doctorPhnoField.sendKeys("78965354");

    WebElement doctorLocationField = driver.findElement(By.name("loc"));
    doctorLocationField.sendKeys("Chennai");

    WebElement insertButton =
driver.findElement(By.xpath("//button[@value='Insert']"));
    insertButton.click();
   }
}
```

1.Automated Inserted:
Before

Result :

1	raja	23	raj@gmail.com	Baby	678976	chennai
2	Dominc	50	dominc@gmail.com	Body	4568998	Chennai
4	Mathew	23	mathew@example.com	Baby	679873	Chennai
5	Jayavel	23	mathew@example.com	Baby	679873	Chennai

After 3 ID Automated Inserted:



Result :

1	raja	23	raj@gmail.com	Baby	678976	chennai
2	Dominc	50	dominc@gmail.com	Body	4568998	Chennai
4	Mathew	23	mathew@example.com	Baby	679873	Chennai
5	Jayavel	23	mathew@example.com	Baby	679873	Chennai
3	Karthik	40	karthik@example.com	orthopedics	78965354	Chennai

2.UPDATED :

```
package com.test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class UpdateTest {
private WebDriver driver;

    @BeforeTest
    public void initializer() {
        System.setProperty("webdriver.Chrome.driver", "driver/Chromedriver.exe");
```

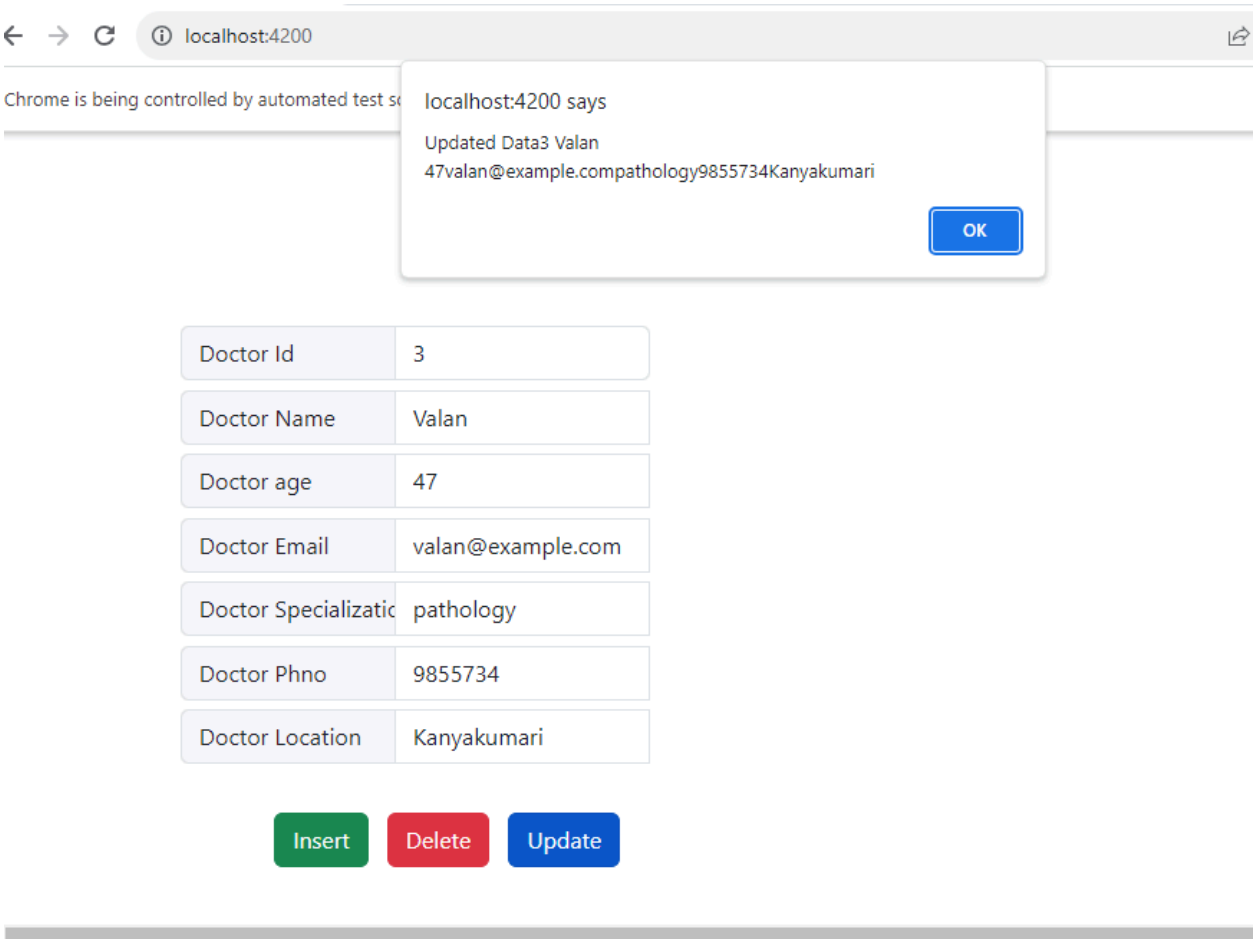
```
        driver = new ChromeDriver();
    }

    @Test
    public void site() {
        driver.navigate().to("http://localhost:4200/");
    }
    @Test
    public void testdoctorFormSubmission() {

        WebElement doctorIdField = driver.findElement(By.name("id"));
        doctorIdField.sendKeys("3");

        WebElement doctorNameField = driver.findElement(By.name("name"));
        doctorNameField.sendKeys("Valan");

        WebElement doctorAgeField = driver.findElement(By.name("age"));
        doctorNameField.sendKeys("47");


        WebElement doctorEmailField = driver.findElement(By.name("email"));
        doctorEmailField.sendKeys("valan@example.com");

        WebElement doctorSpecializationField = driver.findElement(By.name("spl"));
        doctorSpecializationField.sendKeys("pathology");

        WebElement doctorPhnoField = driver.findElement(By.name("phn"));
        doctorPhnoField.sendKeys("9855734");

        WebElement doctorLocationField = driver.findElement(By.name("loc"));
        doctorLocationField.sendKeys("Kanyakumari");

        WebElement updateButton = driver.findElement(By.xpath("//button[@value='Update']"));
        updateButton.click();

      }
}
```

Before:

Result :

1	raja	23	raj@gmail.com	Baby	678976	chennai
2	Dominc	50	dominc@gmail.com	Body	4568998	Chennai
4	Mathew	23	mathew@example.com	Baby	679873	Chennai
5	Jayavel	23	mathew@example.com	Baby	679873	Chennai
3	Karthik	40	karthik@example.com	orthopedics	78965354	Chennai

After:After 3 ID Automated Updated:



Result :

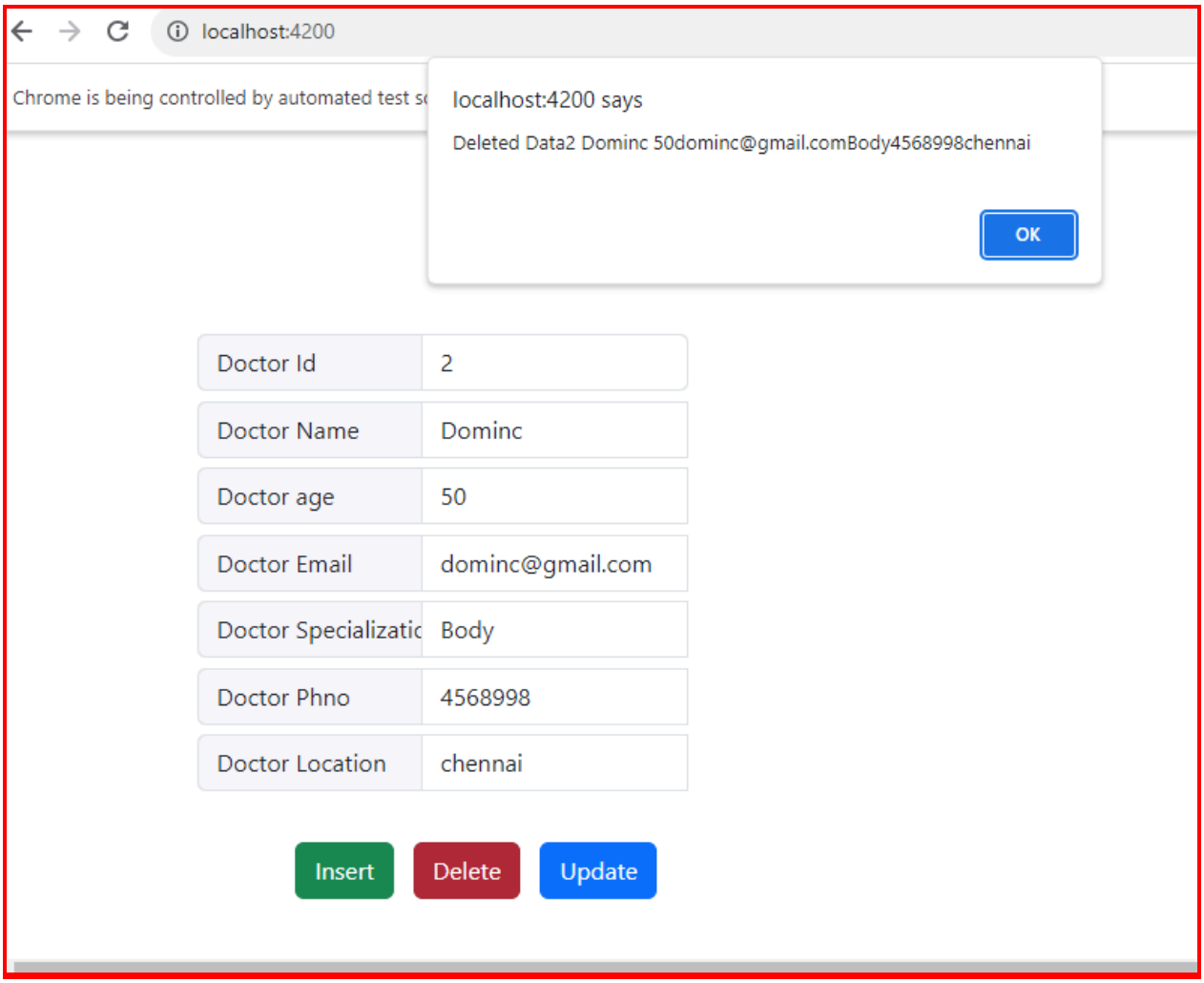
1	raja	23	raj@gmail.com	Baby	678976	chennai
2	Dominc	50	dominc@gmail.com	Body	4568998	Chennai
4	Mathew	23	mathew@example.com	Baby	679873	Chennai
5	Jayavel	23	mathew@example.com	Baby	679873	Chennai
3	Valan	47	valan@example.com	pathology	9855734	Kanyakumari

3.Delete
Before:

Result :

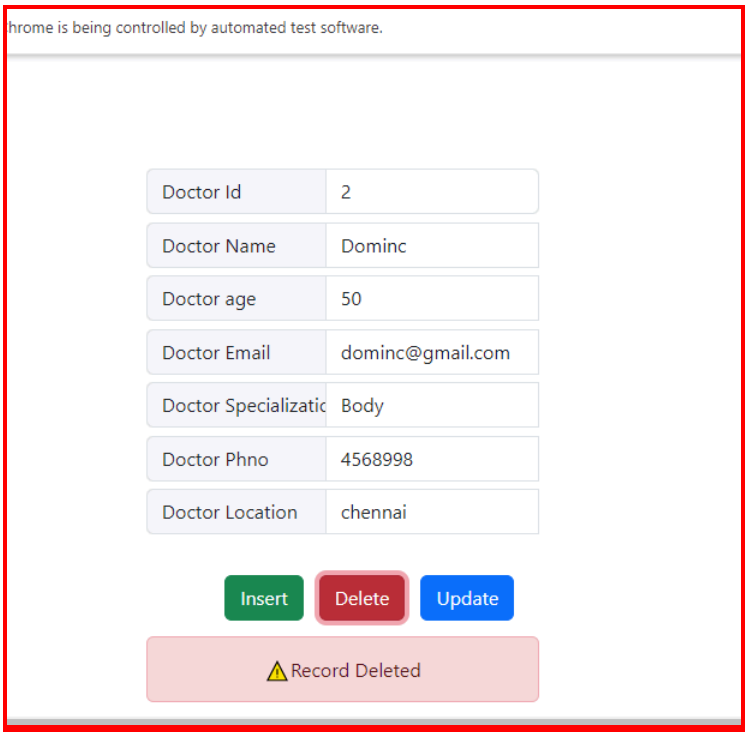
1	raja	23	raj@gmail.com	Baby	678976	chennai
2	Dominc	50	dominc@gmail.com	Body	4568998	Chennai
4	Mathew	23	mathew@example.com	Baby	679873	Chennai
5	Jayavel	23	mathew@example.com	Baby	679873	Chennai
3	Valan	47	valan@example.com	pathology	9855734	Kanyakumari

After 3 ID Automated Delete:

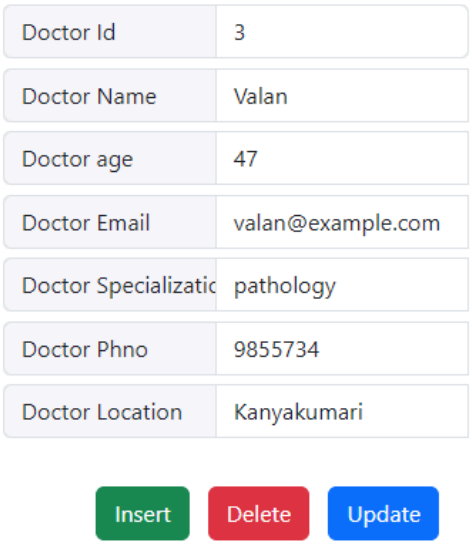


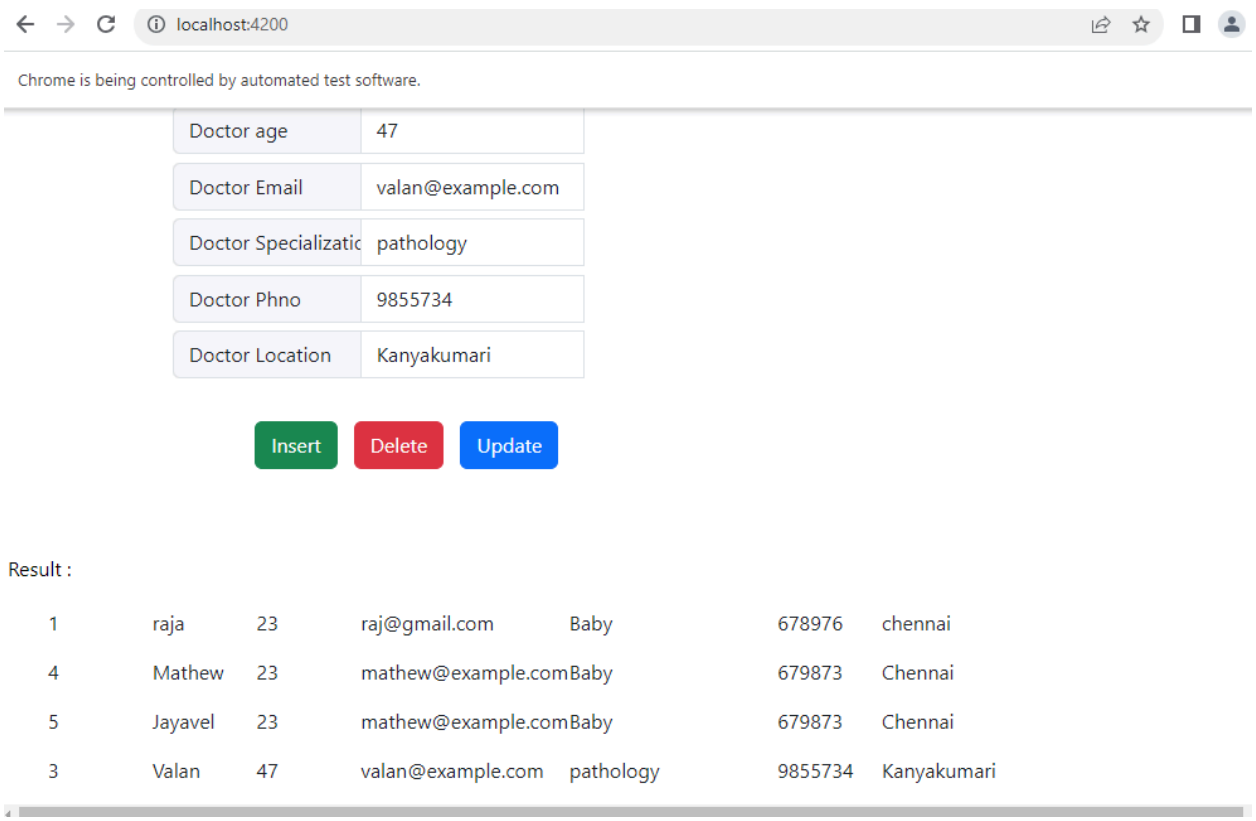
Result :

1	raja	23	raj@gmail.com	Baby	678976	chennai
4	Mathew	23	mathew@example.com	Baby	679873	Chennai
5	Jayavel	23	mathew@example.com	Baby	679873	Chennai
3	Valan	47	valan@example.com	pathology	9855734	Kanyakumari



4.GET PARTICULAR Doctor ID automatically result displayed:





Result :

1	raja	23	raj@gmail.com	Baby	678976	chennai
4	Mathew	23	mathew@example.com	Baby	679873	Chennai
5	Jayavel	23	mathew@example.com	Baby	679873	Chennai
3	Valan	47	valan@example.com	pathology	9855734	Kanyakumari