Vendor Management System

Jeban Ignesh J-11963

White Box Testing (Unit Testing):

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

Code For JUnit Test:

```
package com.vms;
import static org.junit.jupiter.api.Assertions.assertEquals;
import java.util.ArrayList;
import java.util.List;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.vms.bean.Contract;
import com.vms.bean.Login;
import com.vms.bean.Product;
import com.vms.bean.Response;
import com.vms.bean.Vendor;
import com.vms.controller.CONcontroller;
import com.vms.controller.PROcontroller;
import com.vms.controller.VMScontroller;
import com.vms.service.LOGservice;
@SpringBootTest
class SbVmsApplicationTests {
	@Autowired
	CONcontroller contract;
	@Autowired
	LOGservice login;
	@Autowired
	PROcontroller product;
	@Autowired
	VMScontroller vendor;
	String insert;
	String update;
	String delete;
	String result;
```

```
String result1;
List<Vendor> list = new ArrayList<Vendor>();
List<Vendor> list1 = new ArrayList<Vendor>();
Vendor ven1 = new Vendor();
Vendor ven2;
@Test
@Order(1)
void contextLoads() {
}
// ********Login Module******//
@Test
@Order(2)
void performLogin() {
	Login log = new Login("jeban", "11963", "j@gmail.com", "9078567867");
	Response rs1 = new Response("User Success");
	Response rs = login.login(log);
	rs1.equals(rs);
}
@Test
@Order(3)
void performLoginFail() {
	Login log1 = new Login("sara", "9087", "s@gmail.com", "9078567867");
	Response rs1 = new Response("Login Fail");
	Response rs = login.login(log1);
	rs1.equals(rs);
}
@Test
@Order(4)
public void testVendorInsert() {
	Vendor ven = new Vendor();
	ven.setId(11907);
	ven.setName("Jeban");
	ven.setCountry("India");
	ven.setCompany("12344");
	ven.setSsymbol("#123");
	ven.setRelationship("1999");
	ven.setEmail("j@gmail.com");
	ven.setMobile("9078564567");
	insert = "Inserted";
```

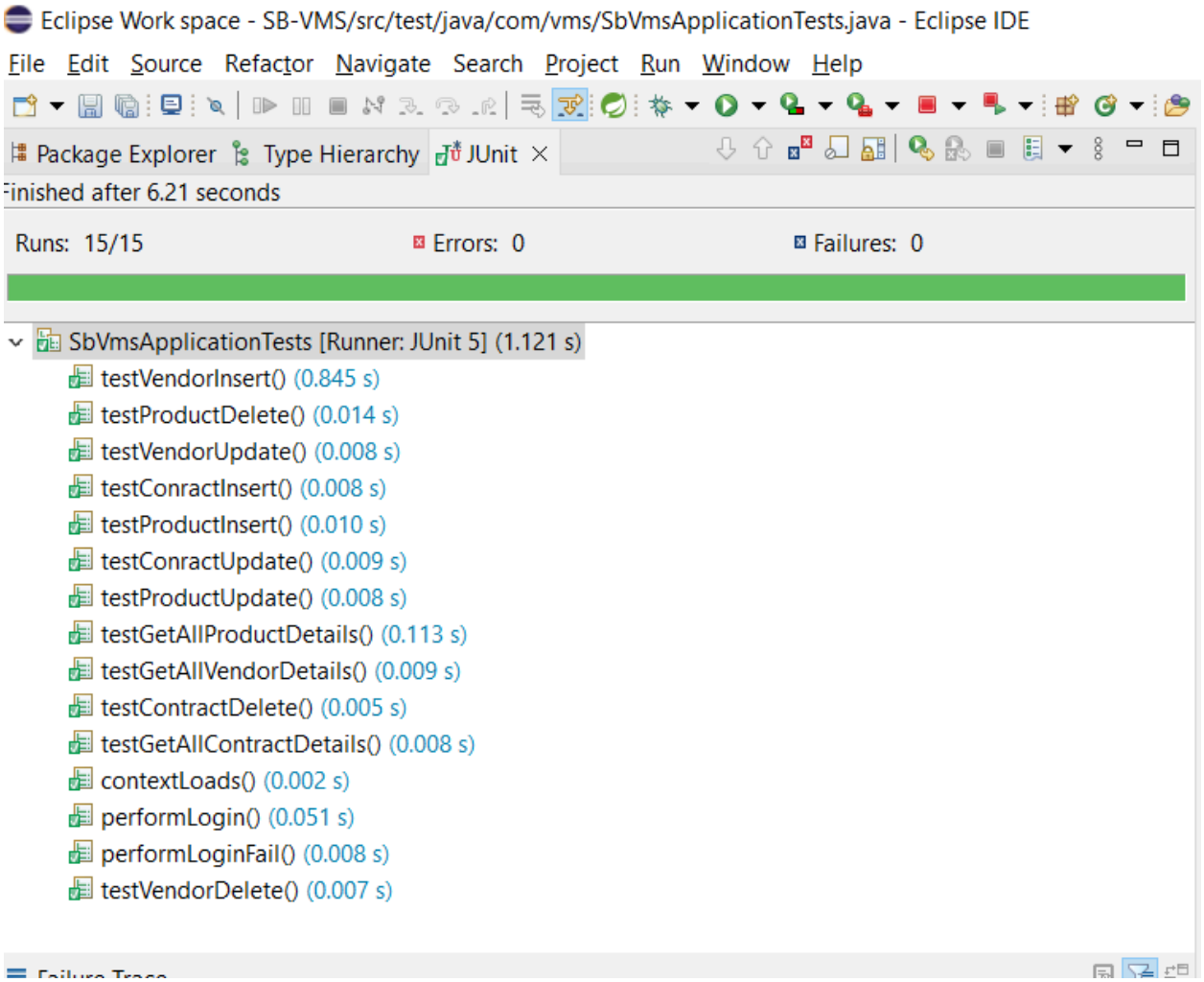
```
		result = vendor.performinsertVendors(ven);
		assertEquals(insert, result);
	}
	@Test
	@Order(5)
	public void testVendorUpdate() {
		Vendor ven = new Vendor();
		ven.setId(11907);
		ven.setName("Ignesh");
		ven.setCountry("India");
		ven.setCompany("12344");
		ven.setSsymbol("#123");
		ven.setRelationship("1999");
		ven.setEmail("j@gmail.com");
		ven.setMobile("9078564567");
		update = "Updated";
		result = vendor.performupdateVendors(ven);
		assertEquals(update, result);
	}
	@Test
	@Order(6)
	public void testVendorDelete() {
		delete = "Deleted";
		result = vendor.performdeleteVendor(106);
		assertEquals(delete, result);
	}
	@Test
	@Order(7)
	public void testGetAllVendorDetails() {
		list = vendor.getAllVendors();
		ven1 = new Vendor(11901, "Domnic", "India", "12345", "@123", "1999",
"d@gmail.com", "907875467");
		ven2 = new Vendor(11902, "Mathew", "India", "675467", "$123", "2000",
"m@gmail.com", "89679087");
		list1.add(ven1);
		list1.add(ven2);
		list.equals(list1);
	}
	// ******Contract******////
```

```
List<Contract> conlist = new ArrayList<Contract>();
List<Contract> conlist1 = new ArrayList<Contract>();
Contract con1 = new Contract();
Contract con2;
@Test
@Order(8)
public void testConractInsert() {
	Contract con = new Contract();
	con.setId(11902);
	con.setName("Sara");
	con.setCno(1234);
	con.setCatogory("Whole sale");
	con.setProducts("Shoes");
	con.setStatus("Onboarding");
	con.setDescribtion("****");
	insert = "Inserted";
	result = contract.performinsertContracts(con);
	assertEquals(insert, result);
}
@Test
@Order(9)
public void testConractUpdate() {
	Contract con = new Contract();
	con.setId(11902);
	con.setName("Shaji");
	con.setCno(1234);
	con.setCatogory("Whole sale");
	con.setProducts("Shoes");
	con.setStatus("Onboarding");
	con.setDescribtion("****");
	update = "Updated";
	result = contract.performupdateContracts(con);
	assertEquals(update, result);
}
@Test
@Order(10)
public void testContractDelete() {
	delete = "Deleted";
	result = contract.performdeleteContracts((long) 11901);
```

```
		assertEquals(delete, result);
	}
	@Test
	@Order(11)
	public void testGetAllContractDetails() {
		conlist = contract.getAllContracts();
		con1 = new Contract(11907, "Mathew", 12345, "Text", "Vichle",
"Onboarding", "Nothig");
		con2 = new Contract(11902, "Domnic", 12789, "Automobile", "Motor",
"Onboarding", "Nothig");
		conlist.add(con1);
		conlist1.add(con2);
		conlist.equals(conlist1);
	}
	// *****Product*******//
	List<Product> prolist = new ArrayList<Product>();
	List<Product> prolist1 = new ArrayList<Product>();
	Product pro1 = new Product();
	Product pro2;
	@Test
	@Order(12)
	public void testProductInsert() {
		Product pro = new Product();
		pro.setId(101);
		pro.setCno(1234);
		pro.setPname("Shoes");
		pro.setPrice("10000");
		pro.setBrand("Adidas");
		insert = "Inserted";
		result = product.performinsertProduct(pro);
		assertEquals(insert, result);
	}
	@Test
	@Order(13)
	public void testProductUpdate() {
		Product pro = new Product();
		pro.setId(101);
		pro.setCno(1234);
		pro.setPname("T shirts");
```

```
		pro.setPrice("10000");
		pro.setBrand("Adidas");
		update = "Updated";
		result = product.performupdateProduct(pro);
		assertEquals(update, result);
	}
	@Test
	@Order(14)
	public void testProductDelete() {
		delete = "Deleted";
		result = product.performdeleteProduct((long) 11901);
		assertEquals(delete, result);
	}
	@Test
	@Order(15)
	public void testGetAllProductDetails() {
		prolist = product.getAllProduct();
		pro1 = new Product(11902, 1234, "Shoes", "120000", "Adidas");
		pro2 = new Product(11903, 6745, "T shirt", "100000", "Adidas");
		prolist.add(pro1);
		prolist1.add(pro2);
		prolist.equals(prolist1);
	}
}
```

Screen Shots:



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