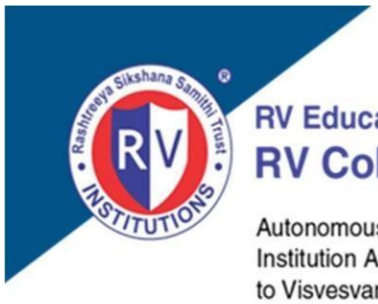


RV Educational Institutions ®
RV College of Engineering ®

Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE, New Delhi

Go, change the world

Report
On

Java Full Stack Based Software Solutions
20MCA000

Digital Bouquet Shopping Platform

Submitted in Partial Fulfillment of the Requirement

for the II Semester MCA

MASTER OF COMPUTER APPLICATIONS

By

1RVPPPPPPP	PIKU MAITY

Under the Incharge
of

Dr. Prashanth K

Department of Master of Computer Applications
RV College of Engineering®, Mysuru Road
RV Vidyanikethan Post, Bengaluru – 560059

October -2022

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF

MASTER OF COMPUTER APPLICATIONS

Bengaluru– 560059

CERTIFICATE

This is to certify that **PIKU MAITY(1RVPPPPPPP)** of 2nd semester Master of Computer Applications program has satisfactorily completed the Assignment titled "**Digital Bouquet Shopping Platform"** in **Java Based Software Solution – 20MCA000** as a part of Continuous Internal Assessment.

Dr. Prashanth k
Associate Professor Department of
MCARVCE, Bengaluru –59

Dr. Andhe Dharani
Professor and Director
Department of MCA
RVCE,Bengaluru–59

Assignment Marks

Assignment-I

SLNO		MAX MARKS	MARKS
1			
2			
3			

Assignment-II

SLNO		MAX MARKS	MARKS
1			
2			
3			

CONTENTS---

Technology

Programming language Java

What is Java?

Java is a widely used object-oriented programming language and software platform that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many others. The rules and syntax of Java are based on the C and C++ languages.

One major advantage of developing software with Java is its portability. Once you have written code for a Java program on a notebook computer, it is very easy to move the code to a mobile device. When the language was invented in 1991 by James Gosling of Sun Microsystems (later acquired by Oracle), the primary goal was to be able to "write once, run anywhere."

It's also important to understand that Java is much different from JavaScript. Javascript does not need to be compiled, while Java code does need to be compiled. Also, Javascript only runs on web browsers while Java can be run anywhere.

How Java works

Before exploring the reasons for Java's enduring popularity, let's review what Java is in more detail and its importance for enterprise application development.

Java is a technology consisting of both a programming language and a software platform. To create an application using Java, you need to download the Java Development Kit (JDK), which is available for Windows, macOS, and Linux. You write the program in the Java programming language, then a compiler turns the program into Java bytecode—the instruction set for the Java Virtual Machine (JVM) that is a part of the Java runtime environment (JRE). Java bytecode runs without modification on any system that supports JVMs, allowing your Java code to be run anywhere.

The Java software platform consists of the JVM, the Java API, and a complete development environment. The JVM parses and runs (interprets) the Java bytecode. The Java API consists of an extensive set of libraries including basic objects, networking and security functions; Extensible Markup Language (XML) generation; and web services. Taken together, the Java language and the Java software platform create a powerful, proven technology for enterprise software development.

Why Java matters

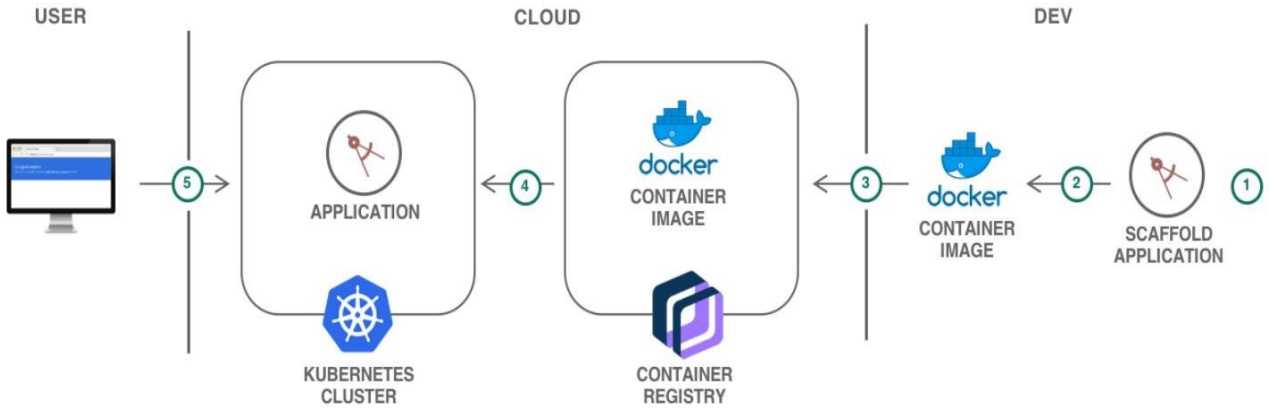
If you are an enterprise application developer, you already know what Java is, and it's likely that your organization already has thousands, even millions, of lines of production code written in Java. You will likely need some level of Java expertise to allow you to troubleshoot, maintain, and upgrade your existing codebase.

However, it would be a mistake to view Java only in terms of legacy applications. The Java language forms the heart of the Android operating system, which powers by far the largest share of the world's smartphones. Java is also among the most popular languages for machine learning and data science applications. Its robustness, ease of use, cross-platform capabilities and security make Java the language of choice for internet solutions in many enterprise shops.

Technical benefits

When it comes to choosing a programming language and environment for your next enterprise application, there are solid technical reasons to consider Java, including interoperability, scalability, and adaptability.

The core philosophy behind its creation—interoperability across disparate devices—remains the strongest argument for favoring Java for new enterprise applications. Java's object-oriented architecture allows you to create modular programs and reusable code, shortening development cycles and extending the longevity of enterprise applications.
Platform scalability is a key attribute of Java. With Java, you can use one single system across a broad range of use cases. Existing desktop applications can be easily adapted to run on smaller devices that have limited resources. You can also migrate applications from mobile to desktop, developing business apps for the Android platform and then integrating them into your current desktop software, bypassing lengthy and expensive development cycles.
Java also wins points with strategic planners for its ability to adapt to new use cases. For example, Java is widely considered to be an ideal platform for the Internet of Things (IoT). The typical IoT application interconnects a large number of disparate devices, a task that is greatly simplified by the fact that billions of devices run Java. Furthermore, Java's extensive ecosystem of developers is constantly developing and sharing new libraries with functionality specifically targeted at IoT application development.



An architecture for developing and deploying a Java-based web application on a Kubernetes cluster

Business benefits

The technical arguments for Java are compelling, but the business reasons to choose Java are equally strong: a large talent pool, a short learning curve, and a wide range of integrated development environments (IDEs).
As more companies use connected devices, machine learning algorithms, and cloud solutions, the demand for skilled developers continues to grow. Many analysts foresee a scarcity of senior-level programmers in the near future, making it difficult to staff new software initiatives. Demand for mobile app developers could soon easily exceed the available supply.
In addition to senior-level developers, major software initiatives also require large numbers of junior contributors. While Java remains a popular introductory programming language in university computer science curriculums, many graduates lack the proficiency to be productive on day one. Java is easier to learn and master than many other programming languages, leading to a shorter learning curve and faster ramp-up to productivity. Java's extensive online community of developer forums, tutorials, and user groups helps beginners get up to speed rapidly and provides seasoned programmers with effective, proven problem-solving tools.
In the area of programming tools, Java offers a range of IDEs. Experienced Java developers can quickly ramp up on a new environment, which frees development managers to choose the IDE that best fits the type of project, budget, development methodology and programmer

skill level. Many seasoned Java programmers think of NetBeans, Eclipse, and IntelliJ IDEA as the top three IDEs for enterprise application development. But there are cases where a more lightweight IDE such as DrJava, BlueJ, JCreator, or Eclipse Che is the best choice.

JSP

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

The Lifecycle of a JSP Page

The JSP pages follow these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (the container invokes jspInit() method).
- Request processing (the container invokes _jspService() method).
- Destroy (the container invokes jspDestroy() method).

As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted

into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

Relational Data base

A relational database is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

A relational database example

Here's a simple example of two tables a small business might use to process orders for its products. The first table is a customer info table, so each record includes a customer's name, address, shipping and billing information, phone number, and other contact information. Each bit of information (each attribute) is in its own column, and the database assigns a unique ID (a key) to each row. In the second table—a customer order table—each record includes the ID of the customer that placed the order, the product ordered, the quantity, the selected size and color, and so on—but not the customer's name or contact information.

These two tables have only one thing in common: the ID column (the key). But because of that common column, the relational database can create a relationship between the two tables. Then, when the company's order processing application submits an order to the database, the database can go to the customer order table, pull the correct information about the product order, and use the customer ID from that table to look up the customer's billing and shipping information in the customer info table. The warehouse can then pull the correct product, the customer can receive timely delivery of the order, and the company can get paid.

How relational databases are structured

The relational model means that the logical data structures—the data tables, views, and indexes—are separate from the physical storage structures. This separation means that database administrators can manage physical data storage without affecting access to that data as a logical structure. For example, renaming a database file does not rename the tables stored within it.

The distinction between logical and physical also applies to database operations, which are clearly defined actions that enable applications to manipulate the data and structures of the database. Logical operations allow an application to specify the content it needs, and physical operations determine how that data should be accessed and then carries out the task.

To ensure that data is always accurate and accessible, relational databases follow certain integrity rules. For example, an integrity rule can specify that duplicate rows are not allowed in a table in order to eliminate the potential for erroneous information entering the database.

Benefits of relational database management system

The simple yet powerful relational model is used by organizations of all types and sizes for a broad variety of information needs. Relational databases are used to track inventories, process ecommerce transactions, manage huge amounts of mission-critical customer information, and

much more. A relational database can be considered for any information need in which data points relate to each other and must be managed in a secure, rules-based, consistent way.

Relational databases have been around since the 1970s. Today, the advantages of the relational model continue to make it the most widely accepted model for databases.

Relational model and data consistency

The relational model is the best at maintaining data consistency across applications and database copies (called instances). For example, when a customer deposits money at an ATM and then looks at the account balance on a mobile phone, the customer expects to see that deposit reflected immediately in an updated account balance. Relational databases excel at this kind of data consistency, ensuring that multiple instances of a database have the same data all the time.

It's difficult for other types of databases to maintain this level of timely consistency with large amounts of data. Some recent databases, such as NoSQL, can supply only "eventual consistency." Under this principle, when the database is scaled or when multiple users access the same data at the same time, the data needs some time to "catch up." Eventual consistency is acceptable for some uses, such as to maintain listings in a product catalog, but for critical business operations such as shopping cart transactions, the relational database is still the gold standard.

ACID properties and RDBMS

Four crucial properties define relational database transactions: atomicity, consistency, isolation, and durability—typically referred to as ACID.

- Atomicity defines all the elements that make up a complete database transaction.
- Consistency defines the rules for maintaining data points in a correct state after a transaction.
- Isolation keeps the effect of a transaction invisible to others until it is committed, to avoid confusion.

Stored procedures and relational databases

Data access involves many repetitive actions. For example, a simple query to get information from a data table may need to be repeated hundreds or thousands of times to produce the desired result. These data access functions require some type of code to access the database. Application developers don't want to write new code for these functions in each new application. Luckily, relational databases allow stored procedures, which are blocks of code that can be accessed with a simple application call. For example, a single stored procedure can provide consistent record tagging for users of multiple applications. Stored procedures can also help developers ensure that certain data functions in the application are implemented in a specific way.

What to look for when selecting a relational database

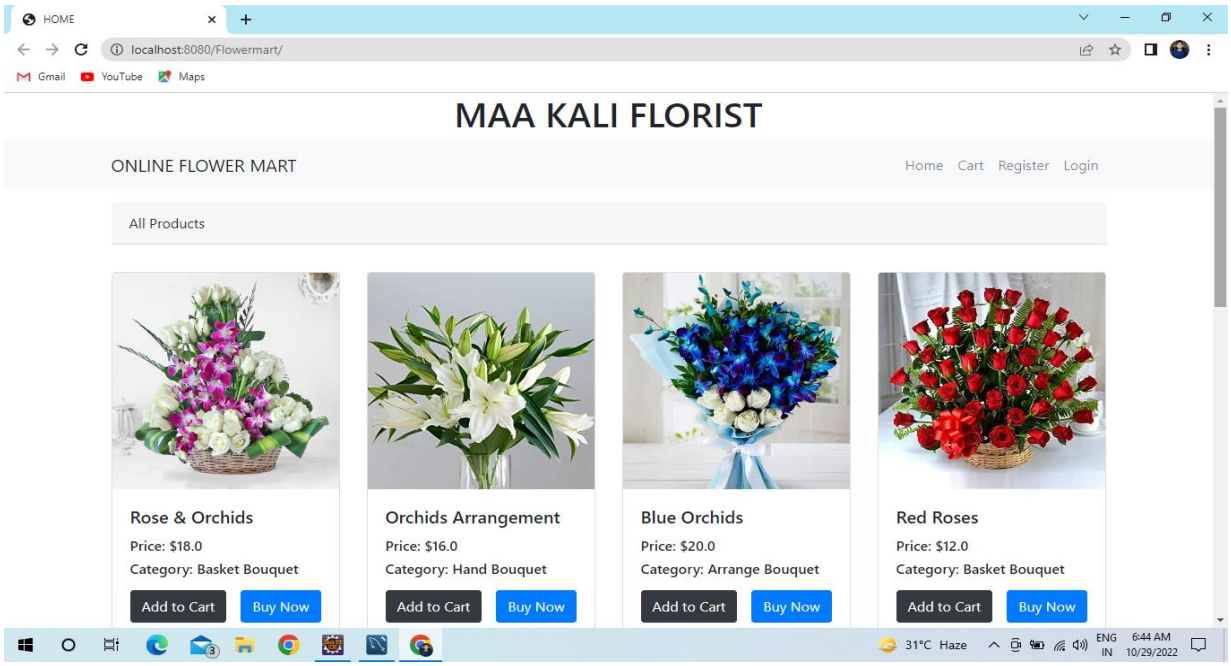
The software used to store, manage, query, and retrieve data stored in a relational database is called a relational database management system (RDBMS). The RDBMS provides an interface between users and applications and the database, as well as administrative functions for managing data storage, access, and performance.

Several factors can guide your decision when choosing among database types and relational database products. The RDBMS you choose will depend on your business needs. Ask yourself the following questions:

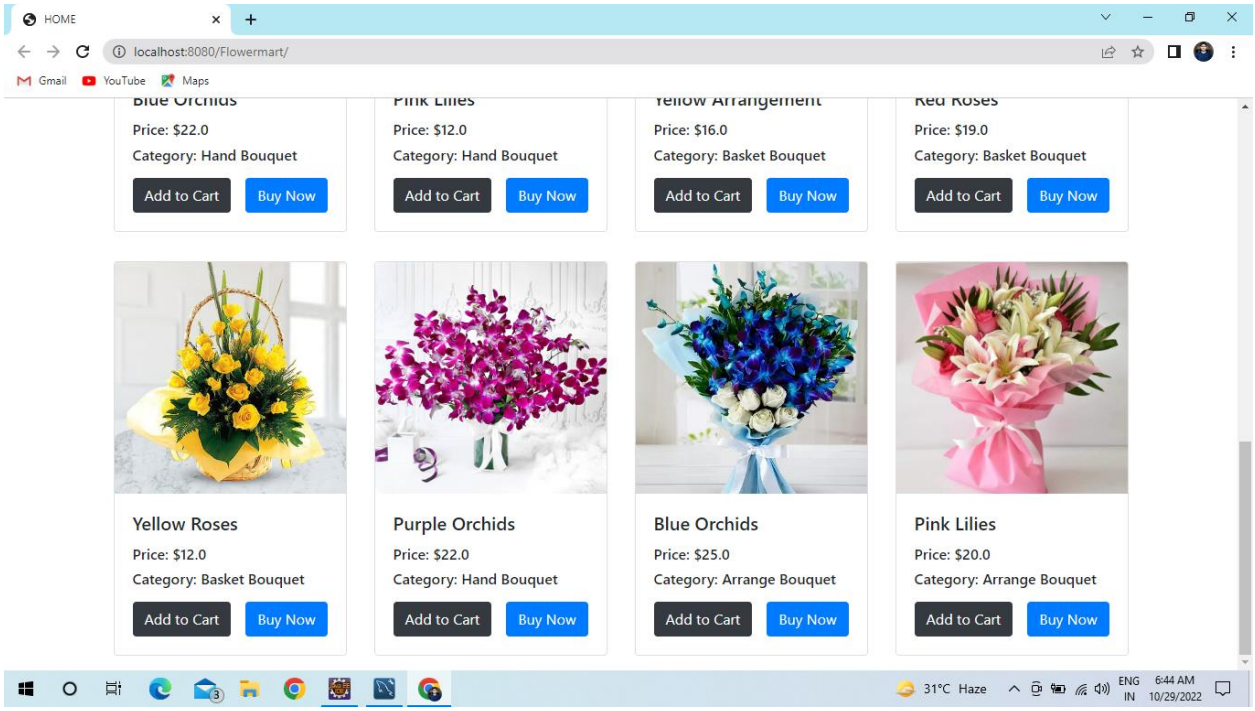
- What are our data accuracy requirements? Will data storage and accuracy rely on business logic? Does our data have stringent requirements for accuracy (for example, financial data and government reports)?
- Do we need scalability? What is the scale of the data to be managed, and what is its anticipated growth? Will the database model need to support mirrored database copies (as separate instances) for scalability? If so, can it maintain data consistency across those instances?
- How important is concurrency? Will multiple users and applications need simultaneous data access? Does the database software support concurrency while protecting the data?
- What are our performance and reliability needs? Do we need a high-performance, high-reliability product? What are the requirements for query-response performance? What are the vendor's commitments for service level agreements (SLAs) or unplanned downtime?

GUI Snapshots

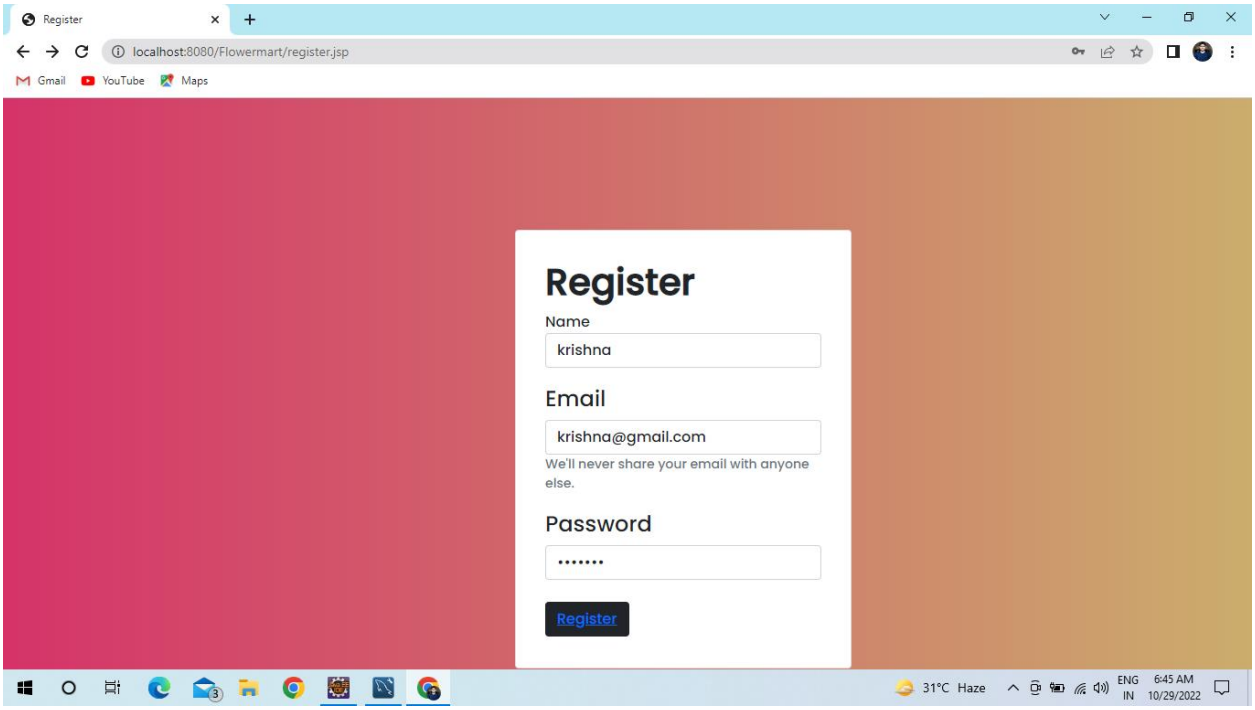
1. Home Page



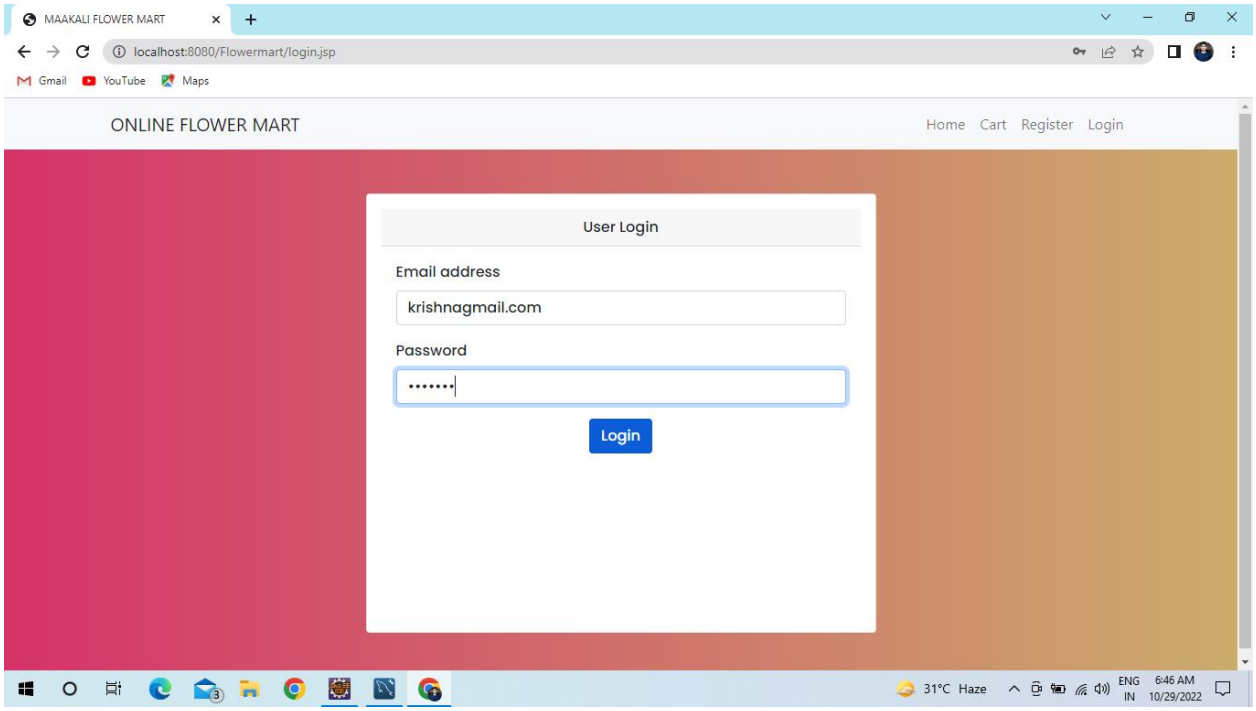
2. Home Page



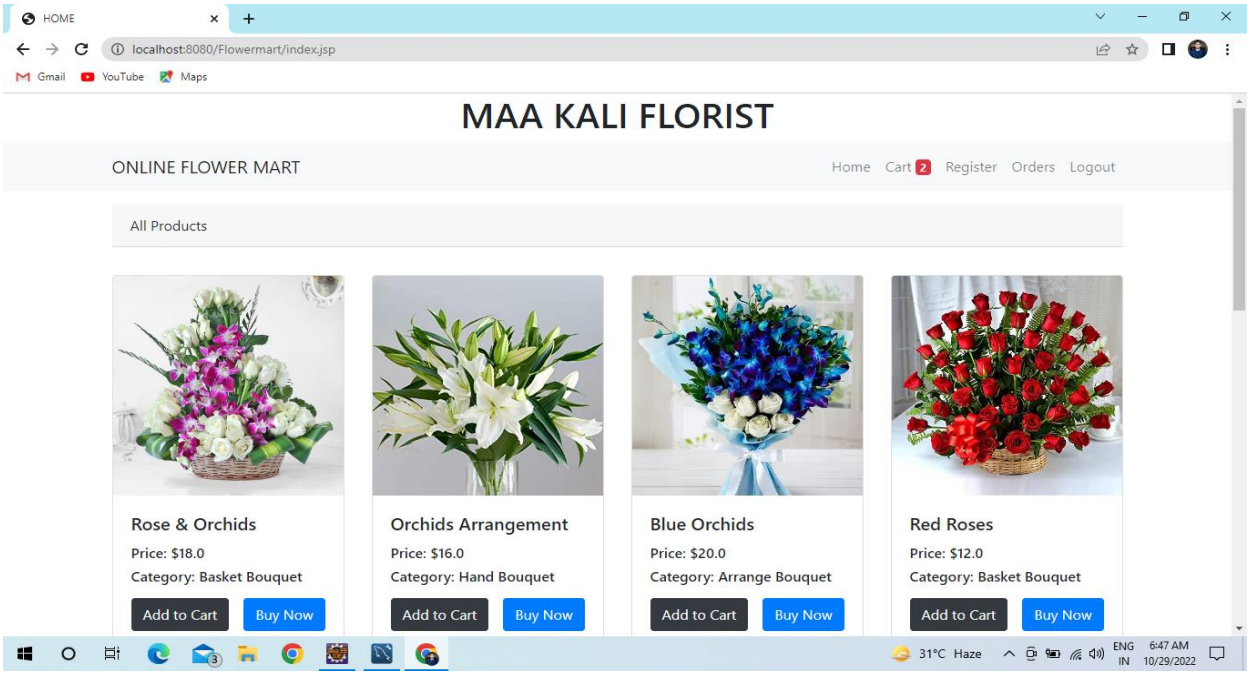
3. Register



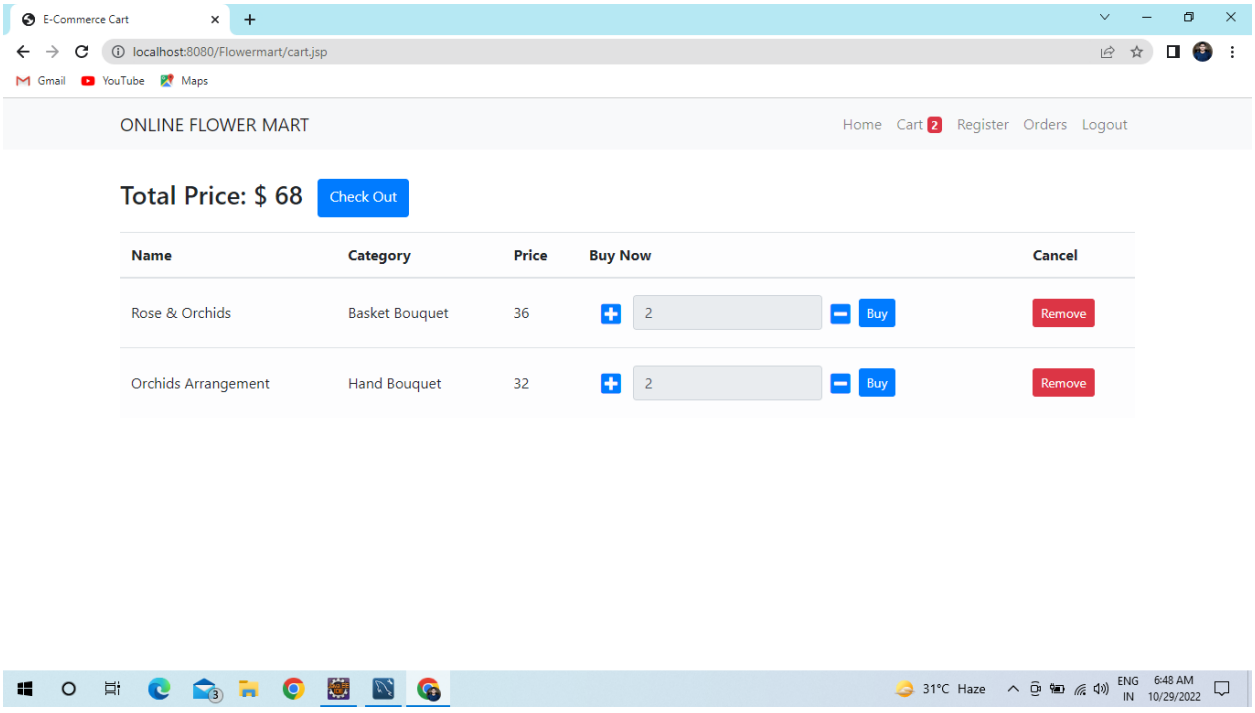
4. Login



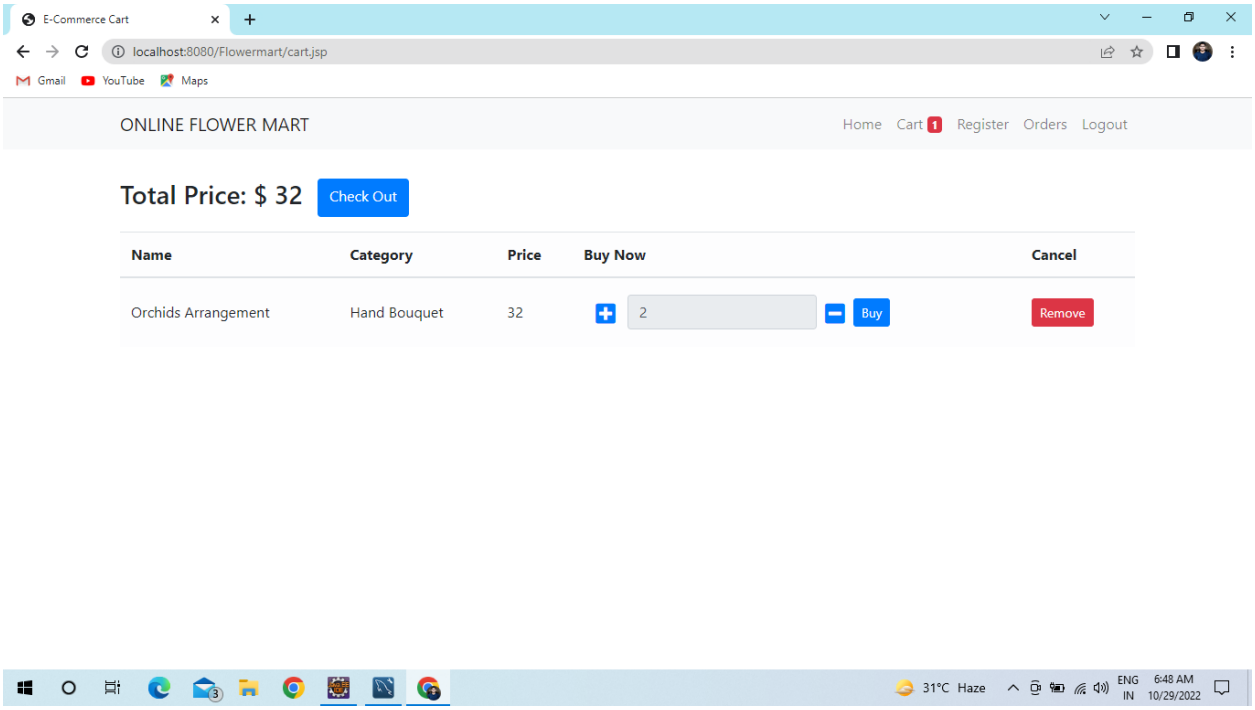
5. Add Product In Cart



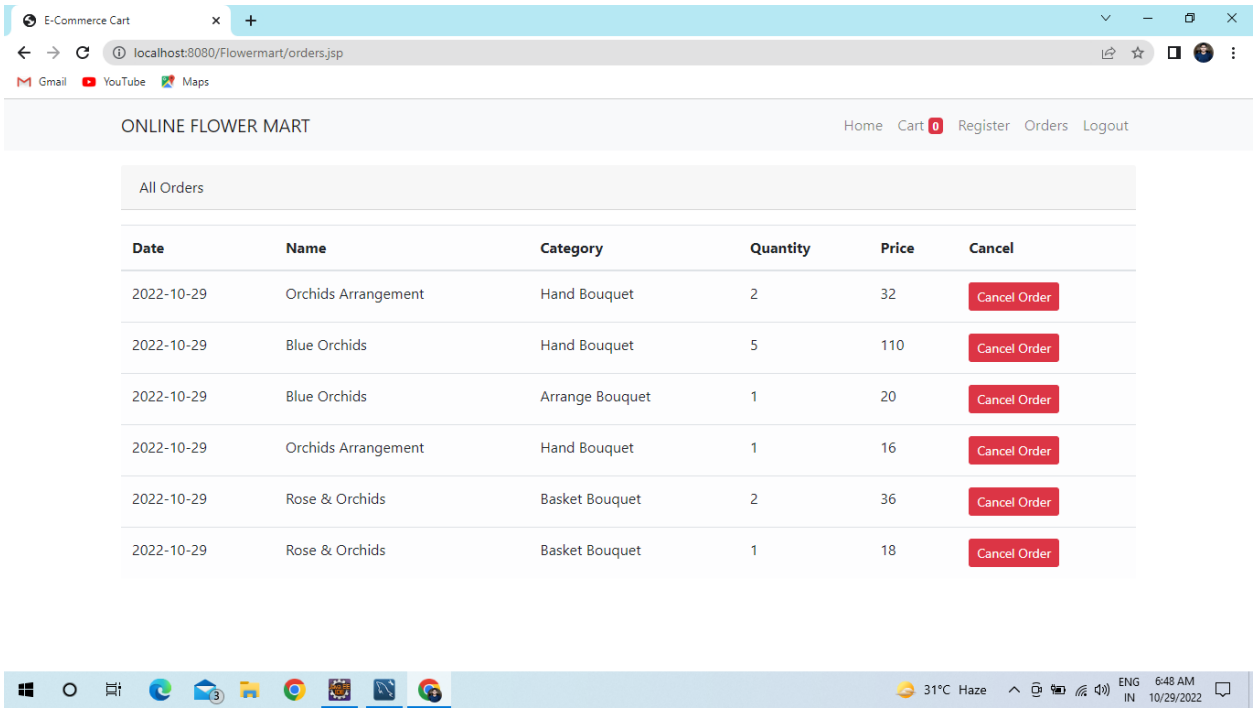
6. Cart



5. Remove Product from Cart



6. Order



CODE

Login.Jsp –

```
<%@page import="maakaliflorist.model.*"%>
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
	<%
	User auth = (User) request.getSession().getAttribute("auth");
	if (auth != null) {
		response.sendRedirect("index.jsp");
	}
	ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
	if (cart_list != null) {
		request.setAttribute("cart_list", cart_list);
	}
	%>
<!DOCTYPE html>
<html>
<head>
<%@include file="/includes/header.jsp"%>
<title>MAAKALI FLOWER MART</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@500&family=Ubuntu:wght@700&display=swa
p" rel="stylesheet">
</head>
<style>
	body {
	    background: linear-gradient(to right, #d53369, #cbad6d);
	}

	.card {
	    margin-top: 9rem;
	    width: 50rem;
	    height: 30rem;
	    padding: 1rem;
	    margin-left: 35rem;
	    font-family: 'Poppins', sans-serif;
	}

	.card-title {
	    font-weight: 700;
	    font-size: 2.5rem;
	}

	#emaillabel {
	    margin-top: 1rem;
	    font-weight: 500;
	    font-size: 1.5rem;
	}

	#exampleInputEmail1 {
	    margin-top: 5px;

	}
```

```
        #exampleInputPassword1 {
            margin-top: 5px;
        }

        #plabel {
            margin-top: 1rem;
            font-weight: 500;
            font-size: 1.5rem;
        }

        .btn-dark {
            font-weight: 500;
            margin-top: 1.5rem;
            margin-bottom: 10px;
        }

        .btn-success {
            font-weight: 500;
        }
    </style>
    <body>
        <%@include file="/includes/navbar.jsp"%>

        <div class="container">
            <div class="card w-50 mx-auto my-5">
                <div class="card-header text-center">User Login</div>
                <div class="card-body">
                    <form action="user-login" method="post">
                        <div class="form-group">
                            <label>Email address</label>
                            <input type="email" name="login-email" class="form-control"
placeholder="Enter email">
                        </div>
                        <div class="form-group">
                            <label>Password</label>
                            <input type="password" name="login-password" class="form-
control" placeholder="Password">
                        </div>
                        <div class="text-center">
                            <button type="submit" class="btn btn-
primary">Login</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>

        <%@include file="/includes/footer.jsp"%>
    </body>
    </html>
```

Index.jsp–

```
<%@page import="maakaliflorist.connection.DBcon"%>
<%@page import="maakaliflorist.bt.Productbt"%>
<%@page import="maakaliflorist.model.*"%>
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>

<%
User auth = (User) request.getSession().getAttribute("auth");
if (auth != null) {
    request.setAttribute("person", auth);
}
Productbt pd = new Productbt(DBcon.getConnection());
List<Product> product = pd.getAllProducts();
ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
if (cart_list != null) {
      request.setAttribute("cart_list", cart_list);
}
%>
<!DOCTYPE html>
<html>
<head>
<%@include file="/includes/header.jsp"%>
<title>HOME</title>
<center><h1>MAA KALI FLORIST</h1></center>
</head>
<body>
      <%@include file="/includes/navbar.jsp"%>

      <div class="container">
            <div class="card-header my-3">All Products</div>
            <div class="row">
                  <%
                  if (!product.isEmpty()) {
                        for (Product p : product) {
                  %>
                  <div class="col-md-3 my-3">
                        <div class="card w-100">
                              <img class="card-img-top" src="product-image/product-
image/<%=p.getImage() %>" alt="Card image cap">
                              <div class="card-body">
                                    <h5 class="card-title"><%=p.getName() %></h5>
                                    <h6 class="price">Price: $<%=p.getPrice() %></h6>
                                    <h6 class="category">Category: <%=p.getCategory() %></h6>
                                    <div class="mt-3 d-flex justify-content-between">
                                          <a class="btn btn-dark" href="add-to-
cart?id=<%=p.getId()%>">Add to Cart</a>
                                           <a class="btn btn-primary" href="order-
now?quantity=1&id=<%=p.getId()%>">Buy Now</a>
                                    </div>
                              </div>
                        </div>
                  </div>
                  <%
                  }
                  } else {
                  out.println("There is no proucts");
                  }
```

```
                %>

            </div>
        </div>

        <%@include file="/includes/footer.jsp"%>
</body>
</html>
```

Orders.jsp –

```
<%@page import="java.text.DecimalFormat"%>
<%@page import="maakaliflorist.bt.Orderbt"%>
<%@page import="maakaliflorist.connection.DBcon"%>
<%@page import="maakaliflorist.bt.Productbt"%>
<%@page import="maakaliflorist.model.*"%>
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
        <%
        DecimalFormat dcf = new DecimalFormat("#.##");
        request.setAttribute("dcf", dcf);
        User auth = (User) request.getSession().getAttribute("auth");
        List<Order> orders = null;
        if (auth != null) {
            request.setAttribute("person", auth);
            Orderbt orderDao = new Orderbt(DBcon.getConnection());
               orders = orderDao.userOrders(auth.getId());
        }else{
               response.sendRedirect("login.jsp");
        }
        ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
        if (cart_list != null) {
               request.setAttribute("cart_list", cart_list);
        }

        %>
<!DOCTYPE html>
<html>
<head>
<%@include file="/includes/header.jsp"%>
<title>E-Commerce Cart</title>
</head>
<body>
        <%@include file="/includes/navbar.jsp"%>
        <div class="container">
               <div class="card-header my-3">All Orders</div>
               <table class="table table-light">
                      <thead>
                             <tr>
                                    <th scope="col">Date</th>
                                    <th scope="col">Name</th>
                                    <th scope="col">Category</th>
                                    <th scope="col">Quantity</th>
                                    <th scope="col">Price</th>
                                    <th scope="col">Cancel</th>
                             </tr>
                      </thead>
                      <tbody>
```

```
                    <%
                    if(orders != null){
                           for(Order o:orders){%>
                                  <tr>
                                         <td><%=o.getDate() %></td>
                                         <td><%=o.getName() %></td>
                                         <td><%=o.getCategory() %></td>
                                         <td><%=o.getQunatity() %></td>
                                         <td><%=dcf.format(o.getPrice()) %></td>
                                         <td><a class="btn btn-sm btn-danger" href="cancel-
order?id=<%=o.getOrderId()%>">Cancel Order</a></td>
                                  </tr>
                           <%}
                    }
                    %>

                    </tbody>
             </table>
       </div>
       <%@include file="/includes/footer.jsp"%>
</body>
</html>
```

Cart.jsp -

```
<%@page import="maakaliflorist.connection.DBcon"%>
<%@page import="maakaliflorist.bt.Productbt"%>
<%@page import="maakaliflorist.model.*"%>
<%@page import="java.util.*"%>
<%@page import="java.text.DecimalFormat"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
       pageEncoding="ISO-8859-1"%>
<%
DecimalFormat dcf = new DecimalFormat("#.##");
request.setAttribute("dcf", dcf);
User auth = (User) request.getSession().getAttribute("auth");
if (auth != null) {
    request.setAttribute("person", auth);
}
ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
List<Cart> cartProduct = null;
if (cart_list != null) {
       Productbt pDao = new Productbt(DBcon.getConnection());
       cartProduct = pDao.getCartProducts(cart_list);
       double total = pDao.getTotalCartPrice(cart_list);
       request.setAttribute("total", total);
       request.setAttribute("cart_list", cart_list);
}
%>
<!DOCTYPE html>
<html>
<head>
<%@include file="/includes/header.jsp"%>
<title>E-Commerce Cart</title>
<style type="text/css">

.table tbody td{
vertical-align: middle;
}
```

```
box-shadow: none;
font-size: 25px;
}
</style>
</head>
<body>
	<%@include file="/includes/navbar.jsp"%>

	<div class="container my-3">
		<div class="d-flex py-3"><h3>Total Price: $ ${(total>0)?dcf.format(total):0} </h3> <a
class="mx-3 btn btn-primary" href="cart-check-out">Check Out</a></div>
		<table class="table table-light">
			<thead>
				<tr>
					<th scope="col">Name</th>
					<th scope="col">Category</th>
					<th scope="col">Price</th>
					<th scope="col">Buy Now</th>
					<th scope="col">Cancel</th>
				</tr>
			</thead>
			<tbody>
				<%
				if (cart_list != null) {
					for (Cart c : cartProduct) {
				%>
				<tr>
					<td><%=c.getName()%></td>
					<td><%=c.getCategory()%></td>
					<td><%= dcf.format(c.getPrice())%></td>
					<td>
						<form action="order-now" method="post" class="form-inline">
						<input type="hidden" name="id" value="<%= c.getId()%>"
class="form-input">
							<div class="form-group d-flex justify-content-
between">
								<a class="btn bnt-sm btn-incre"
href="quantity-inc-dec?action=inc&id=<%=c.getId()%>"><i class="fas fa-plus-square"></i></a>
								<input type="text" name="quantity"
class="form-control"  value="<%=c.getQuantity()%>" readonly>
								<a class="btn btn-sm btn-decre"
href="quantity-inc-dec?action=dec&id=<%=c.getId()%>"><i class="fas fa-minus-square"></i></a>
							</div>
							<button type="submit" class="btn btn-primary btn-
sm">Buy</button>
						</form>
					</td>
					<td><a href="remove-from-cart?id=<%=c.getId() %>" class="btn btn-
sm btn-danger">Remove</a></td>
				</tr>
				<%
				}}%>
			</tbody>
		</table>
	</div>
	<%@include file="/includes/footer.jsp"%>
</body>
</html>
```

Conclusion-

The florist's shop online system was an important application of e-commerce sales model. It could realize online trading of flowers. In the rapid development of the internet today, the florist shop online system would have a great market potential because the flower had certain refreshing time.
Based on elaborating the necessary of development of florist's shop online system, the technology to develop the system was introduce at first in this paper. However, it was a complicated to project to develop the florist shop online system, the further improvement of system must be done combined the problems appeared in the course of the actual use.

Future Work Enhancement:

- In the future the website will enhance.
- Website will more user-friendly.
- In future we will provide Direct UPI payments.
- The website will be more secure and updated.
- In future we will add the voice search option.
- In future we will host the website in the cloud.

THANK YOU!