Name : Nisarg .k. Amlani
Roll : Ce001
Lab : 07
Id : 22ceueg082

WCF Message Contract Code (Service)

=> App.Config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>


 <appSettings>
   <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
 </appSettings>
 <system.web>
   <compilation debug="true" />
 </system.web>
 <!-- When deploying the service library project, the content of the config file must be
added to the host's
 app.config file. System.Configuration does not support config files for libraries. -->
 <system.serviceModel>
   <services>
     <service name="Message_Contract.Service1">
       <host>
         <baseAddresses>
           <add baseAddress =
"http://localhost:8733/Design_Time_Addresses/Message_Contract/Service1/" />
         </baseAddresses>
       </host>
       <!-- Service Endpoints -->
```

```
      <!-- Unless fully qualified, address is relative to base address supplied above -->
      <endpoint address="" binding="basicHttpBinding"
contract="Message_Contract.IService1">
        <!--
            Upon deployment, the following identity element should be removed or replaced
to reflect the
            identity under which the deployed service runs.  If removed, WCF will infer an
appropriate identity
            automatically.
        -->
        <identity>
          <dns value="localhost"/>
        </identity>
      </endpoint>
      <!-- Metadata Endpoints -->
      <!-- The Metadata Exchange endpoint is used by the service to describe itself to
clients. -->
      <!-- This endpoint does not use a secure binding and should be secured or
removed before deployment -->
      <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange"/>
    </service>
  </services>
  <behaviors>
    <serviceBehaviors>
      <behavior>
        <!-- To avoid disclosing metadata information,
        set the values below to false before deployment -->
        <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True"/>
        <!-- To receive exception details in faults for debugging purposes,
        set the value below to true.  Set to false before deployment
        to avoid disclosing exception information -->
        <serviceDebug includeExceptionDetailInFaults="False" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
 </system.serviceModel>


</configuration>
```

=> IService.cs Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;


namespace Message_Contract
{
  // NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.
  [ServiceContract]
  public interface IService1
  {
     [OperationContract]
     string InitiateOrder();


     [OperationContract]
     BookOrder PlaceOrder(BookOrder request);


     [OperationContract]
     string FinalizeOrder();


     // TODO: Add your service operations here
  }


  [MessageContract]
  public class BookOrder
  {
     private string isbn;
     private int quantity;
     private string firstname;
     private string lastname;
```

```
private string address;
private string ordernumber;


public BookOrder()
{
}


public BookOrder(BookOrder message)
{
    this.isbn = message.isbn;
    this.quantity = message.quantity;
    this.firstname = message.firstname;
    this.lastname = message.lastname;
    this.address = message.address;
    this.ordernumber = message.ordernumber;
}


[MessageHeader]
public string ISBN
{
    get { return isbn; }
    set { isbn = value; }
}


[MessageBodyMember]
public int Quantity
{
    get { return quantity; }
    set { quantity = value; }
}

[MessageBodyMember]
public string Firstname
{
    get { return firstname; }
    set { firstname = value; }
```

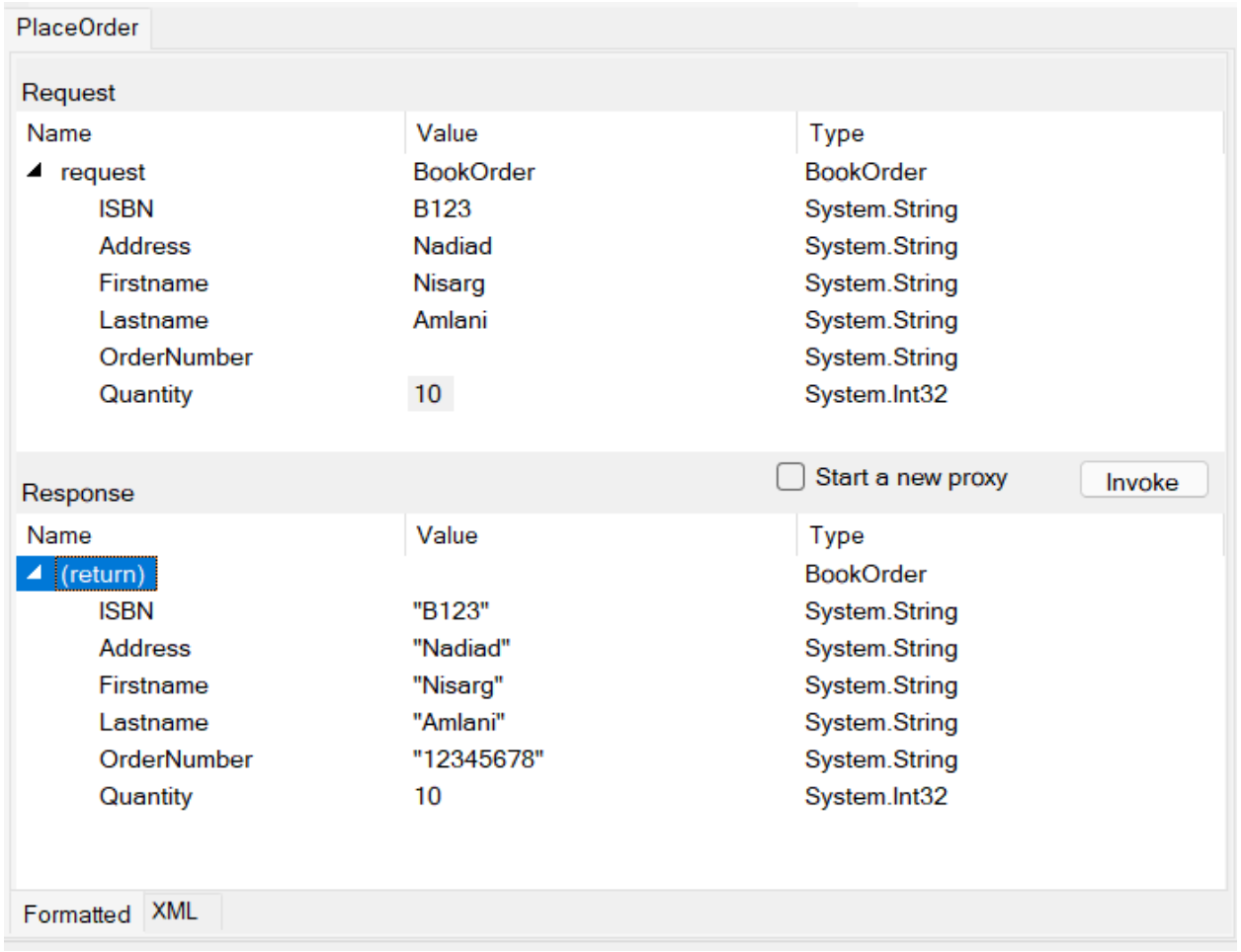
```
        }


        [MessageBodyMember]
        public string Lastname
        {
            get { return lastname; }
            set { lastname = value; }
        }


        [MessageBodyMember]
        public string Address
        {
            get { return address; }
            set { address = value; }
        }


        [MessageBodyMember]
        public string OrderNumber
        {
            get { return ordernumber; }
            set { ordernumber = value; }
        }
    }
}
```

=> Service.cs Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;


namespace Message_Contract
```

```
{
   // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
class name "Service1" in both code and config file together.
   public class Service1 : IService1
   {
      string IService1.InitiateOrder()
      {
         return "Initiating Order...";
      }


       BookOrder IService1.PlaceOrder(BookOrder request)
      {
         BookOrder response = new BookOrder(request);
         response.OrderNumber = "12345678";
         return response;
      }


      string IService1.FinalizeOrder()
      {
         return "Order placed sucessfully.";
      }
   }
}
```

=> Output (Service)



PlaceOrder

Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Header>
  <Action s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
  <h:ISBN xmlns:h="http://tempuri.org/">B123</h:ISBN>
 </s:Header>
 <s:Body>
  <BookOrder xmlns="http://tempuri.org/">
   <Address>Nadiad</Address>
   <Firstname>Nisarg</Firstname>
   <Lastname>Amlani</Lastname>
   <OrderNumber />
   <Quantity>10</Quantity>
  </BookOrder>
 </s:Body>
```

Response

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Header>
  <h:ISBN xmlns:h="http://tempuri.org/">B123</h:ISBN>
 </s:Header>
 <s:Body>
  <BookOrder xmlns="http://tempuri.org/">
   <Address>Nadiad</Address>
   <Firstname>Nisarg</Firstname>
   <Lastname>Amlani</Lastname>
   <OrderNumber>12345678</OrderNumber>
   <Quantity>10</Quantity>
  </BookOrder>
 </s:Body>
</s:Envelope>
```

=> Windows Form App (Host)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
     <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <system.serviceModel>

 <services>
       <service name="Message_Contract.Service1"
behaviorConfiguration="metadataSupport">
         <host>
           <baseAddresses>
             <add baseAddress="http://localhost:8733/Design_Time_Addresses/" />
             <add baseAddress="net.pipe://localhost/Message_Contract" />
             <add baseAddress="net.tcp://localhost:8000/Message_Contract" />
           </baseAddresses>
         </host>


         <endpoint address="" binding="wsHttpBinding"
contract="Message_Contract.IService1" />


         <endpoint address="tcpmex" binding="mexTcpBinding"
contract="IMetadataExchange" />
         <endpoint address="namedpipemex" binding="mexNamedPipeBinding"
contract="IMetadataExchange" />


       </service>
     </services>
     <behaviors>
       <serviceBehaviors>
         <behavior name="metadataSupport">
           <serviceMetadata httpGetEnabled="false" httpGetUrl="" />
         </behavior>
       </serviceBehaviors>
     </behaviors>
```

```
  </system.serviceModel>
</configuration>
```

=> App.config (host)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
     <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <system.serviceModel>

 <services>
        <service name="Message_Contract.Service1"
behaviorConfiguration="metadataSupport">
          <host>
            <baseAddresses>
              <add baseAddress="http://localhost:8733/Design_Time_Addresses/" />
              <add baseAddress="net.pipe://localhost/Message_Contract" />
              <add baseAddress="net.tcp://localhost:8000/Message_Contract" />
            </baseAddresses>
          </host>


          <endpoint address="" binding="wsHttpBinding"
contract="Message_Contract.IService1" />


          <endpoint address="tcpmex" binding="mexTcpBinding"
contract="IMetadataExchange" />
          <endpoint address="namedpipemex" binding="mexNamedPipeBinding"
contract="IMetadataExchange" />


        </service>
      </services>
      <behaviors>
        <serviceBehaviors>
```

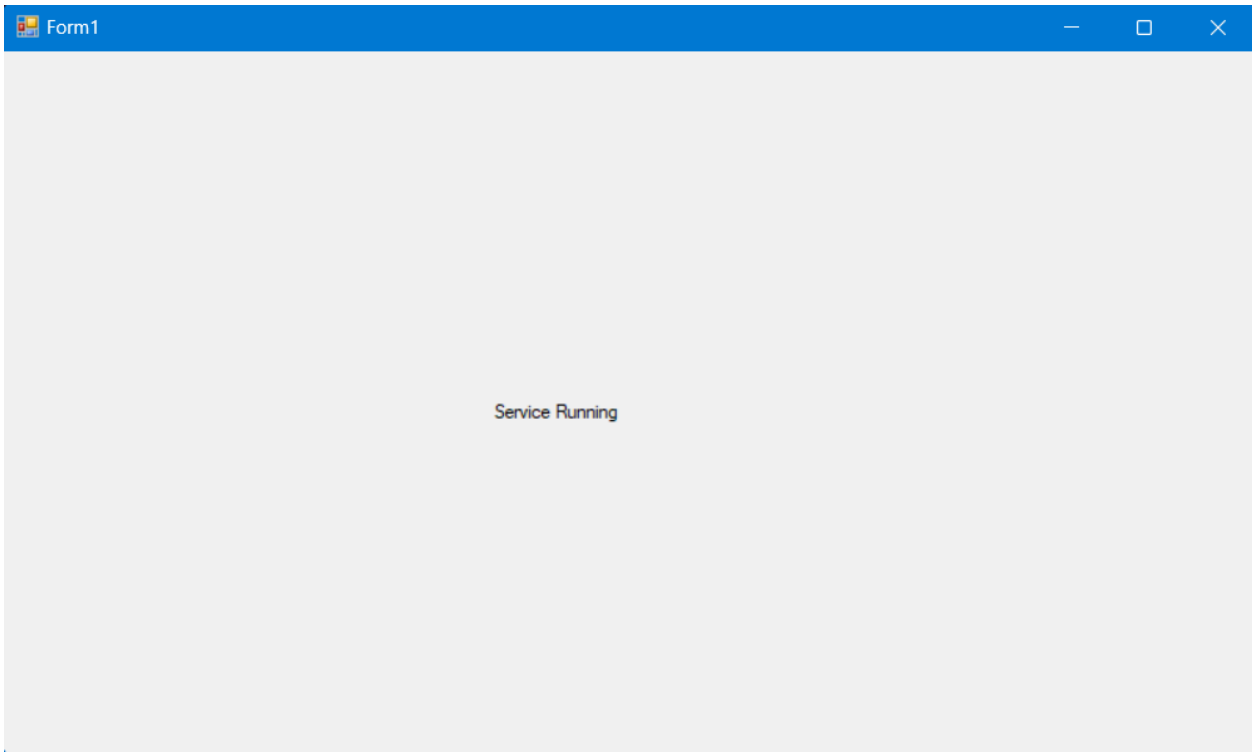
```
        <behavior name="metadataSupport">
          <serviceMetadata httpGetEnabled="false" httpGetUrl="" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

=> Form.cs (Host)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.ServiceModel;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace WindowsFormsAppHost
{
  public partial class Form1 : Form
  {
    ServiceHost sh = null;
    public Form1()
    {
      InitializeComponent();
    }

    private void Form1_FormClosing(object sender ,
FormClosedEventArgs e)
    {
      sh.Close();
    }
```

```
        private void Form1_Load_1(object sender, EventArgs e)
        {
            sh = new ServiceHost(typeof(Message_Contract.Service1));
            sh.Open();
            lbl1.Text = "Service Running";
        }
    }
}
```

=> Output (Host)



=> Windows Form App (Client)

=> App.Config Code (Client)

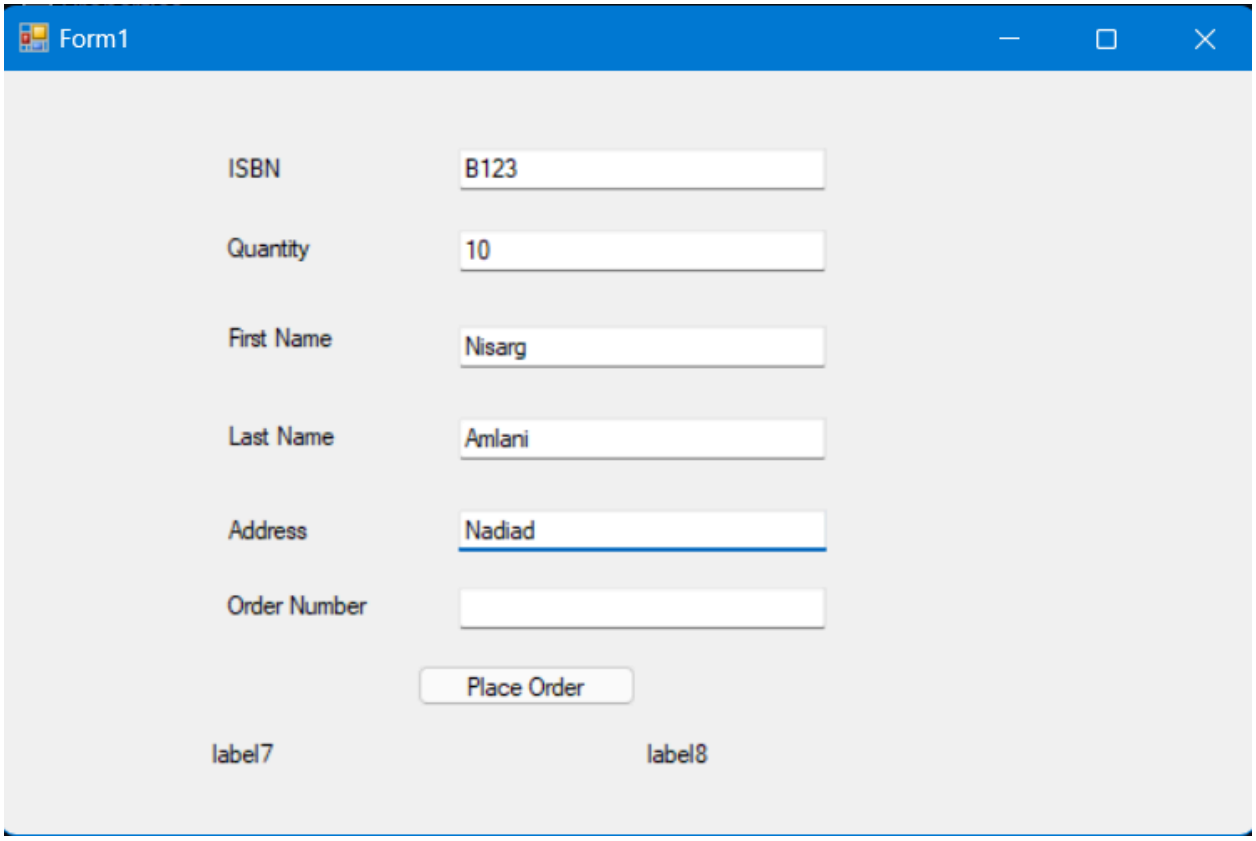
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8.1" />
  </startup>
  <system.serviceModel>
    <bindings>
      <wsHttpBinding>
        <binding name="WSHttpBinding_IService" />
        <binding name="WSHttpBinding_IService1" />
      </wsHttpBinding>
    </bindings>
    <client>


      <endpoint address="http://localhost:8733/Design_Time_Addresses/"
        binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_IService1"
        contract="TCP.IService1" name="WSHttpBinding_IService1">
        <identity>
          <userPrincipalName value="NISARGSLIG3\LENOVO" />
        </identity>
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

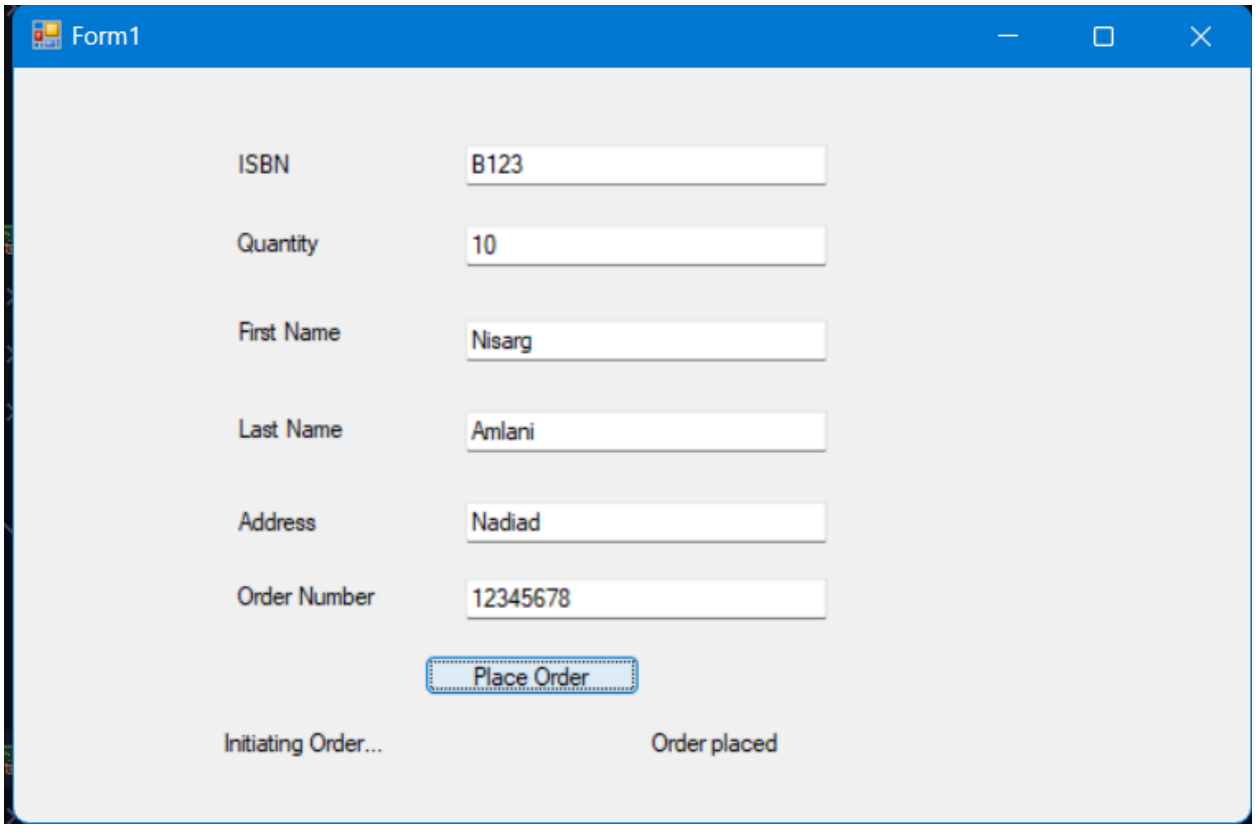
=> Form.cs (Client)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;



namespace WindowsFormsAppClient
{
  public partial class Form1 : Form
  {
     public Form1()
     {
        InitializeComponent();
     }


     private void pl_order_Click_1(object sender, EventArgs e)
     {
        TCP.Service1Client client = new TCP.Service1Client();
        label7.Text = client.InitiateOrder();
        TCP.BookOrder order = new TCP.BookOrder();
        order.ISBN = textBox1.Text;
        order.Quantity = int.Parse(textBox2.Text);
        order.Firstname = textBox3.Text;
        order.Lastname = textBox4.Text;
        order.Address = textBox5.Text;


        TCP.BookOrder reply =          ((TCP.IService1)client).PlaceOrder(order);
        textBox6.Text = reply.OrderNumber;
        label8.Text = client.FinalizeOrder();
        client.Close();
     }
  }
}
```

=> Output (Client)


Form1
ISBN
B123
Quantity
10
First Name
Nisarg
Last Name
Amlani
Address
Nadiad
Order Number
Place Order
label7
label8


Form1
ISBN
B123
Quantity
10
First Name
Nisarg
Last Name
Amlani
Address
Nadiad
Order Number
12345678
Place Order
Initiating Order...
Order placed