<table>
<tr><td rowspan="3"> 

_____

Signature

I hereby subject myself to the examination rules and regulations of Tshwane University of Technology</td><td colspan="2">Faculty of Information And Communication Technology (ICT)
Department of Computer Science
Subject: Distributed Programming
DSD117V

Semester Test 2</td></tr>
<tr><td>Date 18 October 2025
Number of Pages: 15</td><td>1st Examiner:
Dr. ML. Gadebe</td></tr>
<tr><td colspan="2">Total Marks 65</td></tr>
<tr><td>Group Number:</td><td colspan="2">Student Number:</td></tr>
</table>

Question 1 (22)

1. Create a persistence.xml that will interact with the pjpDbase that are mapped with entity managers. //2/

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="SemesterTestProject-ejbPU" transaction-type="JTA"> //2 mark
    <jta-data-source>jdbc/sample</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.schema-generation.database.action" value="create"/>
    </properties>
  </persistence-unit>
</persistence>
```

2. Create four entities (***User, Contact, Customer and Product***) as shown on the system architecture and link them to ***tblUser, tblContact, tblCustomer and tblProduct*** respectively. //6/

```
@Entity
@Table(name=" tblUser ")//1 mark
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS)
public class User implements Serializable
{
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
   private int id;
   private String name;
   private String surname;
   private String address;
  @OneToMany(cascade=CascadeType.ALL) //1 mark
   private List<Contact> contacts;
   private String username;
   private String password;

>>>>>>
}
```

```
Entity
@Table(name="tblCustomer")//1 mark
public class Customer extends User
{
   private int customerNo;
   private double creditBalance;
…….
}
```

```
Entity
@Table(name="tblContacts")//1 mark
public class Contact
{
   private int contactNO;
   private String telephone;
   private String cellphoneNo;
   private String email;
…….
}
```

```
@Entity//1 mark
public class Product implements Serializable//1 mark
{
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
private int productID;
   private String name;
   private String productType;
   private int qty;
   private double price
```

3. Create a local session bean called ***CustomerBean*** that will store, delete and validate a customer. The *validateLogin()* method validates login using username and password and return an object of type Customer if the logons are valid otherwise return null. //6/

```
package za.ac.tut.person;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;


public class CustomerBean implements CustomerBeanService
{

   @PersistenceContext(unitName="SemesterTestProject-ejbPU")//1 mark
   private EntityManager entityManager;

   @Override
   public void storeCustomer(Customer customer) {
      entityManager.persist(customer); //1 mark
   }

   public Customer getCustomer(int id) {
      return entityManager.find(Customer.java, id);
   }

@Override
   public void deleteCustomer(int id) {
      entityManager.remove(getCustomer(id)); //1 mark
   }

   @Override
   public Customer validateLogon(String username, String password)
   {
```

```
        String sql = "select customer from Customer customer where customer.username like '"+username+"' and customer.password
    '"+password+"'";//1 mark
        Customer cust = null;
        Query query = entityManager.createQuery(sql); //1 mark
        try
        {
          cust = (Customer) query.getSingleResult();//1 mark
        }
        catch(Exception er)
        {
           cust = null;
        }
        return cust;
      }
}
```

4. Create a local stateless session bean called ***ProductBean*** with *storeProduct(), updateProduct(), getProduct() and getAllProduct()* methods as shown in system architecture. //8/

```
@Stateless
public class ProductBean implements ProductBeanLocal
{
   @PersistenceContext(unitName="SemesterTestProject-ejbPU")

   private EntityManager entityManager;
   @Override
   public List<Product> getAllProducts()
   {
     String sql = "select product from Product product";//1 mark
```

```
    List<Product> products;
    Query query = entityManager.createQuery(sql); //1 mark

    try
    {
      products = (List<Product>) query.getResultList();//1 mark

    }
    catch(Exception er)
    {
       products = null;
    }
    return products;
}

@Override
public void storeProduct(Product product) //1 mark
{
   entityManager.persist(product); //1 mark

}

@Override
public Product getProduct(int productId)
{
   return entityManager.find(Product.class, productId); //1 mark

}

@Override
public void updateProduct(Product product)
{
   Product productGet = getProduct(product.getProductID());//1 mark

   if (productGet != null)
   {
      entityManager.merge(product); //1 mark
   }
```

```
    }

}
```

Question 2 (28)

1. Create a message driven bean called LogonMDB reads text messaged from queue named “jms/logonQueue”. The text contains username and password separated by a # key. Then, the message driven bean will pass the logons to the CustomerBean to validate the logons. If the Customer is found the LogonMDB will publish an object of type Customer otherwise it will publish “customer not found” message to a queue named “jms/logonQueue” as a confirmation.

//12/

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "jms/logonQueues"),
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue")
})
public class LogonMDB implements MessageListener {//1 mark
    @Resource(mappedName="jms/logonQueueFactory")
    ConnectionFactory connectFactory;
    @EJB
    CustomerBeanService customerBean; //1 mark
    public LogonMDB() {
    }

    @Override
```

```
public void onMessage(Message message)
{
   if (message instanceof TextMessage)
   {
      Connection connect = null;
      try {
         TextMessage txtMsg = (TextMessage) message; //1 mark
         String[] logons = txtMsg.getText().split("#");//1 mark
         //Validate logons
         Customer cust = customerBean.validateLogon(logons[0], logons[1]); //1 mark
         //Send the feedback
          connect = connectFactory.createConnection();//1 mark
          //Create session
          Session session = connect.createSession(false, Session.AUTO_ACKNOWLEDGE); //1 mark
          //Publisher
          Queue queue = (Queue) message.getJMSDestination();//1 mark
          MessageProducer publish = session.createProducer(queue);
          //determine

          if (cust != null)
          {
            ObjectMessage objMsg = session.createObjectMessage();
            objMsg.setObject(cust); //1 mark
            publish.send(objMsg); //1 mark
          }
          else
          {
```

```
                    txtMsg.setText("customer not found");//1 mark
                    publish.send(txtMsg); //1 mark
                }

            } catch (JMSException ex) {
                Logger.getLogger(LogonMDB.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

}
```

2. Create a stateful session bean named ShoppingCartBean that will initialise the shopping cart data member, allow a user to add and remove products to/from a shopping cart data member. Also the stateful session bean will return all products added to a shopping cart. //8/

```
import java.util.ArrayList;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.ejb.Stateful;
import za.ac.tut.product.Product;

/**
 *
 * @author MosesGadebe
```

```
 */
@Stateful
public class ShoppingCart implements ShoppingCartBeanService {
    private List<Product> products; 1 Mark
    @Override
    @PostConstruct1 Mark
    public void initialliseShopCart() {
        products = new ArrayList<Item>();1 Mark
    }

    @Override
   @Interceptors(ShoppingInterceptor.class) //1 mark
   public void addToCart(Product product) {
      items.add(product); 1 Mark
    }

    @Override
    public void removeItem(int itemId)
    {
        for(Product product: products)        {
            if (product.getProductsID() == itemId) 1 Mark
            {
                products.remove(product); 1 Mark
                break;
            }
        }
    }

    @Override
    public List<Product> getAllItem()
    {
        return products; 1 Mark
    }
    @Override

}
```

3. Create a reusable interceptor class named ***ShoppingInterceptor*** that intercepts *addToCart*() method to add a VAT of 14% to a parameter of type Product. //8/

```
@Interceptor//1 mark
public class ShoppingInterceptor {
   @AroundInvoke
   public Object addToCartInterrceptor(InvocationContext cnt) throws Exception//1 mark

   {
      Object[] parameters = cnt.getParameters();//1 mark
     if (parameters != null)
     {
        for(Object parameter: parameters ) //1 mark
         {
            if (parameter instanceof Product)
            {
               Product product = (Product) parameter; //1 mark

                     //Add Levy
                      item.setPrice(item.getPrice() + item.getPrice() * 0.14); //2 mark

                  }

         }
     }
      return cnt.proceed();//1 mark
```

```
        }
}
```

Question 3 (15)

1. Create a controller class called ***ShoppingServlet*** that override the doPost method to do the following:

- The doPost method will accept the username and password to validate a customer. The method will publish the combined logons "username#password" to a queue named "jms/logonQueue" and wait for confirmation. If the confirmed message is of type Customer redirect the user to ***shoppingCart.jsp*** and display a list of products as shown on the system architecture. //15/

```
@WebServlet(urlPatterns = {"/ ShoppingServlet "})
public class ShoppingServlet extends HttpServlet {
@Resource(mappedName="jms/logonQueueFactory")//1 mark
ConnectionFactory connectFactory;
@Resource(mappedName="jms/logonQueue")//1 mark
Queue queue;
@EJB
ProductBeanLocal productBean;
    @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
  {
     String custNo, password, logons;
     username = request.getParameter("usernme");
```

```
        password = request.getParameter("password");
        logons = username +"#"+password;

        //Connection
        Connection connection = null;
        Connection connection2 = null;
    try {
        connection = connectFactory.createConnection();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        TextMessage txtMsg = session.createTextMessage();
        txtMsg.setText(logons); //1 mark
        txtMsg.setJMSReplyTo(queue); //1 mark
        MessageProducer publish = session.createProducer(queue); //1 mark
        publish.send(txtMsg); //1 mark
        //Consume the returned message
        connection2 = connectFactory.createConnection();//1 mark
        Session session2 = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageConsumer consumer = session2.createConsumer(queue); //1 mark
        connection2.start();//1 mark
        Message message = consumer.receive(0); //1 mark
         try (PrintWriter out = response.getWriter()) {
           /* TODO output your page here. You may use following sample code. */
           out.println("<!DOCTYPE html>");
           out.println("<html>");
           out.println("<head>");
           out.println("<title>Servlet NewServlet</title>");
           out.println("</head>");
```

```
            out.println("<body>");


                if (message instanceof TextMessage) //1 mark
                {
                    TextMessage txtMsgOut = (TextMessage) message;
                    out.println("<h1> " + txtMsgOut.getText() + "</h1>");

                }
                else if (message instanceof ObjectMessage) //1 mark
                {
                    ObjectMessage objMsg = (ObjectMessage) message; //1 mark
                    Customer cust = (Customer) objMsg.getObject();//1 mark
                    Request.setAttribute("products", productBean.getAllProducts());
                  request.getRequestDispatcher("shoppingCart.jsp").forward(request, response); //1 mark

                }
              out.println("</body>");
               out.println("</html>");
             }
               } catch (JMSException ex) {
          Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
      }
      }
  }
```