Tshwane University of Technology We empower people ________________ Signature **I hereby subject myself to the examination rules and regulations of Tshwane University of Technology**	**Faculty of Information And Communication Technology (ICT)** **Department of Computer Science** DSD117V	
	Semester Tutorial 1 **August 2025** **Number of Pages: 6**	1st Examiner: Dr. M. L. Gadebe
	Duration: 3 hours **Total Mark: 70**	
Group Number:	**Student Number:**	

Question 1 **//45/**

1. Create a database called NokoDbase using Derby database management service and create the JDBC Pool and resource using glassfish.
2. Crete a persistence.xml to persist the entities and to configure the database connection. [2]

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="abcResource" transaction-type="JTA">
    <jta-data-source>jdbc/empResource</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>
```

3. Create an entity called Book map it to tblBook also map author, title, and publisher fields to book_author, book_title and book_publisher respectively. Ensure that the bookID is entered automatically. [10]

```
@Entity//1 mark
@Table//1 mark (name=”tblBook”) // 1 Mark
public class Book implements Serializable // 1 Mark
{
@Id //1 mark
@GeneratedValue(strategy=GenerationType.AUTO)//2 mark
    private int bookID;
   private String bookType;
    private String ISBN;
@Column(name=”book_publisher”) //1 mark
    private String publisher;
@Column(name=”book_author”) //1 mark
    private String author;
@Column(name=”book_title”) //1 mark
    private String title;
    private String edition;
    private double price;
```

4. Create a local stateless session bean called BookServiceBean and mapped it to the persistence.xml to do the following: [6]
    a. addBook the method receives a composite value object Book and persist it to the database [5]
    b. getBook the method receives the author name and book title, then retrieve and return Book. [5]
    c. getAllBooks the method retrieves all Books and return a List of Book [5]
    d. updateBook the method receives the Book and update the instance of the Book to the database. [5]
    e. deleteBook the method receives the book primary key to the delete a LiteratureBook instance from the database [5]

```
@Remote//1 mark
public interface BookService//1 mark
{
   public void addBook(Book); //1 mark
public Book getBook (String author, String title);
public List< Book > getAllBooks ();//1 mark
public void updateBook(Book book):void
public String deleteLiteratureBook(int booki);
}
```

```
public class BookServiceBean implements BookService//2 mark
{
@PersistenceContext("abcResource")
private EntityManager manager;
@TransactionAttribute//1 mark (TransactionAttributeType.REQUIRED) //1 mark
  public void addBook(Book book) //1 mark
  {
     manager.persist(book);//2 mark
  }
 public Book getBook(String eauthor, String etitle)
 {
    String sql = "select objBook from Book objBook where objBook.author Like : author and
objBook.title Like: title";//1 mark
     Query query = manager.createQuery(sql);//1 mark
        query.setParameter("author", eauthor); //1 mark
       query.setParameter("title", etitle); //1 mark
     try
       {
          book = query. getSingleResult();//1 mark

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

 public List<Book> getAllBook()//1 mark
 {
    String sql = "select objBook from Book objBook";//1 mark
     Query query = manager.createQuery(sql);//1 mark
      List<Book> ejbList = query.getResultList();//2 mark
      return ejbList;
   }
   return list;
 }

@TransactionAttribute(TransactionAttributeType.REQUIRED) //1 mark
  public void updateBook( Book book) //1 mark
  {
     Book ejbLit =  getBook( book.getbookID());/1 mark

            if(ejbLit!=null)/1 mark
            {
                     manager.merge(book); //1 mark
            }

  }

  @TransactionAttribute(TransactionAttributeType.REQUIRED) //1 mark
   public void deleteBook( int bookID) //1 mark
  {
    Book ejbBook = manager getBook(bookID);/1 mark
            if(ejbBook!=null)/1 mark
            {

                     manager.remove(ejbBook);/1 mark

            }
  }

}
```

5. Create a remote singleton session bean called CounterServiceBean that will keep the count of all clients calls [5]

```
package za.ac.tut.session;
import javax.ejb.Remote;
@Remote1 mark
public interface SingletonSessionBeanLocal 1 mark
{
   public int counter();
   public void initialise()
}
```

```
package za.ac.tut.session;
import javax.ejb.Singleton;
@Singleton1 mark
public class SingletonSessionBean implements SingletonSessionBeanLocal
{
   private int count;
  @PostConstruct1 mark
public void initialise()
{
  count=0;
}
   @Override
   public int counter()
   {
       return count++;1 mark
   }
   }
```

6. Create an index,jsp page as shown in system architecture and keep track of all clients calls logon on the system [5]

Question 2 **//25/**

Create a controller class called BookServlet that overrides the doPost method to do the following:

1. The doPost method will accept a request to add a new LiteratureBook to the database and display the message “Record Added” using results.jsp
/5/
2. The doPost method will accept a request to search for a LiteratureBook using book id and display the LiteratureBook details on the form input text fields using updateBook.jsp
/7/
3. The doPost method will accept a request to delete the instance of LiteratureBook from the database table using book id and display the message “Book Deleted” use the results.jsp
/3/
4. The doPost method will accept a request to update the instances of LiteratureBook object to the database table and display the message “Book update” using result.jsp
/5/
5. Create the web deployment descriptor to configure the servlet.
/5/

```
public class BookServlet
{
@EJB/1 mark
BookService bookBean; /1 mark
public void doPost(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
{

     String choice = request.getParameter("decision");
                PrintWriter out = response.getWriter();
                RequestDispatcher dispatcher =null;
                 ServiceBook dao = new ServiceBook();
                 try{

              if(choice.equals("Add"))/1 mark
                 {
                     Book book = new Book(request.getParameter("authors")
                                                         request.getParameter("title"), request.getParameter("edition"),
                                                            0, request.getParameter("ISBN"),
                                                            request.getParameter("publisher"),
                                                            Double.parseDouble(request.getParameter("price")));/1 mark


                                      bookBean.addBook(book); /1 mark
                                      request.setAttribute("result","Record Added");/1 mark
                                      dispatcher =request.getRequestDispatcher("result.jsp");/1 mark
                                      dispatcher.foward(request,response);
                 }
              else if(choice.equals("search"))/1 mark
                 {
                    String author = Integer.parseInt(request.getParameter("author "));/1 mark
                   String author = Integer.parseInt(request.getParameter("title"));
                     Book book = bookBean.getBook(author, title);           /1 mark
```

```
                        request.setAttribute("book",book); /1 mark
                        dispatcher =request.getRequestDispatcher("results.jsp");/1 mark
                        dispatcher.foward(request,response); /1 mark
                    }else if(choice.equals("delete"))/1 mark
                    {

                       int id = Integer.parseInt(request.getParameter("bookID"));/1 mark
                      bookBean.deleteBook(id);              /1 mark
                       request.setAttribute("result","Book deleted");/1 mark
                      dispatcher =request.getRequestDispatcher("result.jsp");/1 mark
                      dispatcher.foward(request,response);

                    }else if(choice.equals("update"))
                    {
                      Book  book = new Book(request.getParameter("authors")/1 mark
                                                    request.getParameter("title"),
                                                    request.getParameter("edition"),
                                                    Integer.parseInt(request.getParameter("bookID")),/1 mark
                                                    request.getParameter("ISBN"),
                                                    request.getParameter("publisher"),/1 mark
                                                   Double.parseDouble(request.getParameter("price")));

                     bookBean.updateBook(lbook); /1 mark
                     request.setAttribute("result","Book updated");
                     dispatcher =request.getRequestDispatcher("result.jsp");/1 mark
                     dispatcher.foward(request,response);
                    }

                    }catch(Exception e)
                    {
                     out.println("error " + e.getMessage());
                    }


  dao.close();
}
}
```

****
****

--- result.jsp

```
<head>
 <title>Results</title>
<head>
<body>
<%@page import =”za.ac.tut.book.Book%”>
<%
    String result = (String)request.getAttribute("result");

              if(result!=null)

              {
%>
                        <h4><%=result%></h3>
<%
                      }
%>

<%
    Book book = (Book) resquest.getAttribute("book");
              if(book!=null)
              {
```

```
%>

   <form action ="book.do" method ="post">
    <p>Enter ISBN:<input type="text" name ="ISBN" value ="<%=book.getISBN()%>"></p>
              <p>Enter Publisher:<input type="text" name ="publisher" value
="<%=book.getPublisher()%>"></p>
              <p>Enter Price:<input type="text" name ="price" value ="<%=book.getPrice()%>"></p>
              <p>Enter Authors:<input type="text" name ="authors" value ="<%=book.getAuthors()%>"></p>
              <p>Enter Title:<input type="text" name ="title" value ="<%=book.getTitle%>"></p>
              <p>Enter Edition:<input type="text" name ="edition" value ="<%=book.getEdition()%>"></p>
               <p>Enter bookID:<input type="text" name ="bookID" value ="<%=book.getBookID()%>"></p>
              <p> <input type="submit" name="decision" value ="search">
               <input type="submit" name="decision" value ="Add"><p>
              </form>
<%
   }
%>
</body>
```

```
--- index.jsp
<head>
 <title>Results</title>
<head>
<body>
<%@page import="za.ac.tut.session.SingletonSessionBeanLocal"%>
<%@page import="za.ac.tut.session.YearMarkService"%>
<%@page import="javax.naming.InitialContext"%>
<%
            //Create a stateless session
            InitialContext ic = new InitialContext(); /1 mark
            //Connect to the session bean
            SingletonSessionBeanLocal serviceCounter = (SingletonSessionBeanLocal)/1 mark
ic.lookup("za.ac.tut.session.SingletonSessionBeanLocal");/1 mark
%>
   <form action ="book.do" method ="post">
    <p>Enter ISBN:<input type="text" name ="ISBN" value ="">
              <p>Enter Publisher:<input type="text" name ="publisher" value =""></p>
              <p>Enter Price:<input type="text" name ="price" value =""></p>
              <p>Enter Authors:<input type="text" name ="authors" value =""></p>/
              <p>Enter Title:<input type="text" name ="title" value =""></p>
              <p>Enter Edition:<input type="text" name ="edition" value =""></p>
               <p>Enter bookID:<input type="text" name ="bookID" value =""></p>
              <p> <input type="submit" name="decision" value ="search">
               <input type="submit" name="decision" value ="Add"><p>
              </form>
<%
   }
%>
   <p> User number : <%=serviceCounter.counter() %>/2 mark
</body>
```