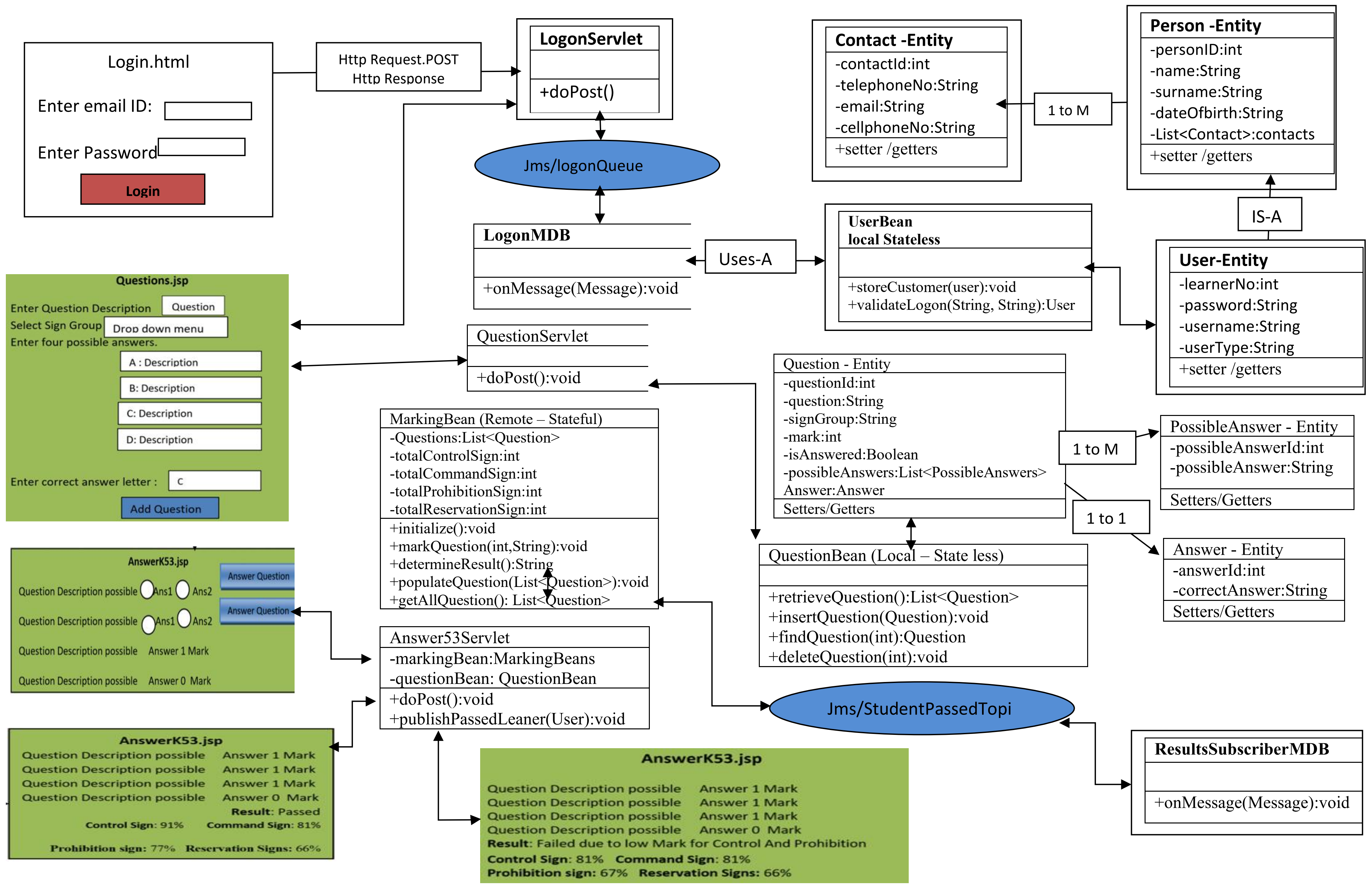
Department of Transport in the Maluti district require the online K53 exam enterprise application which includes the web application. The web application should allow the Maluti examiner to set the K53 questions online, by entering questions with their four possible answers and a correct answer. Also, the web application should only allow registered applicant to login and write a K53 exam depending on different group signs, then the application should give the user the immediate mark for each question he/she had answered. The web application should allocate marks depending on the sign group (control signs, command signs, prohibition signs, and reservation signs) average mark. For the incumbent learner to pass a learner's licence at the end of the session he/she must obtain the following average percentage mark for each sign group:

**Sign Group Table 1**

| **Sign Group** | **Pass Mark** |
|---|---|
| control signs | >=90 |
| command signs | >=80 |
| prohibition signs | >=70 |
| reservation signs | >=70 |

After all questions are answered the application should be terminated and the result should be given as pass or fail.


Login.html
Enter email ID:
Enter Password
Login
Http Request.POST
Http Response
LogonServlet
+doPost()
Jms/logonQueue
LogonMDB
+onMessage(Message):void
Uses-A
UserBean
local Stateless
+storeCustomer(user):void
+validateLogon(String, String):User
Contact -Entity
-contactId:int
-telephoneNo:String
-email:String
-cellphoneNo:String
+setter /getters
1 to M
Person -Entity
-personID:int
-name:String
-surname:String
-dateOfbirth:String
-List<Contact>:contacts
+setter /getters
IS-A
User-Entity
-learnerNo:int
-password:String
-username:String
-userType:String
+setter /getters
Questions.jsp
Enter Question Description
Question
Select Sign Group
Drop down menu
Enter four possible answers.
A : Description
B: Description
C: Description
D: Description
Enter correct answer letter :
C
Add Question
QuestionServlet
+doPost():void
Question - Entity
-questionId:int
-question:String
-signGroup:String
-mark:int
-isAnswered:Boolean
-possibleAnswers:List<PossibleAnswers>
Answer:Answer
Setters/Getters
1 to M
PossibleAnswer - Entity
-possibleAnswerId:int
-possibleAnswer:String
Setters/Getters
1 to 1
Answer - Entity
-answerId:int
-correctAnswer:String
Setters/Getters
MarkingBean (Remote – Stateful)
-Questions:List<Question>
-totalControlSign:int
-totalCommandSign:int
-totalProhibitionSign:int
-totalReservationSign:int
+initialize():void
+markQuestion(int,String):void
+determineResult():String
+populateQuestion(List<Question>):void
+getAllQuestion(): List<Question>
QuestionBean (Local – State less)
+retrieveQuestion():List<Question>
+insertQuestion(Question):void
+findQuestion(int):Question
+deleteQuestion(int):void
AnswerK53.jsp
Answer Question
Question Description possible
Ans1
Ans2
Answer Question
Question Description possible
Ans1
Ans2
Question Description possible
Answer 1 Mark
Question Description possible
Answer 0 Mark
Answer53Servlet
-markingBean:MarkingBeans
-questionBean: QuestionBean
+doPost():void
+publishPassedLeaner(User):void
Jms/StudentPassedTopi
ResultsSubscriberMDB
+onMessage(Message):void
AnswerK53.jsp
Question Description possible
Answer 1 Mark
Question Description possible
Answer 1 Mark
Question Description possible
Answer 1 Mark
Question Description possible
Answer 0 Mark
Result: Passed
Control Sign: 91%
Command Sign: 81%
Prohibition sign: 77%
Reservation Signs: 66%
AnswerK53.jsp
Question Description possible
Answer 1 Mark
Question Description possible
Answer 1 Mark
Question Description possible
Answer 1 Mark
Question Description possible
Answer 0 Mark
Result: Failed due to low Mark for Control And Prohibition
Control Sign: 81%
Command Sign: 81%
Prohibition sign: 67%
Reservation Signs: 66%

## Question 1 **(30)**

1. Create a persistence.xml that will allow a user to interact with malutiDbase hosted on a derby database. //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="malutiPU" transaction-type="JTA">//2 Marks
    <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 three entities (Person, Contact and User) with their existing relationships as shown on the system architecture. Map each entity to tblPerson, tblContact and tblUser respectively and implement inheritance type as "table per class". //7/

```
@Entity1 Mark
@Table(name="tblPerson") 1 Mark
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS) 1 Mark
public class Person implements Serializable
{
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
   private int id;
   private String name;
   private String surname;
   private String dateOfBirth;
   @OneToMany (cascade = CascadeType.ALL)
```

```
@Entity
@Table(name="tblUser")1 Mark
public class User extends Person 1 Mark
{
   private int learnerNo;
   private String userName;
   private String password;
   private String userType;
}
```

```
   private List<Contact> contacts;
}
```

3.

```
@Entity
@Table(name="tblContact")1 Mark
public class Contact implements Serializable1 Mark
{
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
   private int contactId;
   private String telephoneNo;
   private String cellphoneNo;
   private String email;
   @ManyToOne
   private Person person;
}
```

4. Create another three entities (Question, Answer, PossibleAnswers) with their existing relationships as shown on the system architecture. //7/

```
@Entity//1 Mark
@Table(name="tblQuestion")
public class Question implements Serializable//1 Mark
{
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO) //1 Mark
   private int questionID;
   private String question;
   private boolean isAnswered;
   private int mark;
   private String signGroup;
   @OneToMany//1 Mark (cascade=CascadeType.PERSIST)
   private List<PossibleAnswer> possibleAnswers;
   @OneToOne//1 Mark (cascade=CascadeType.PERSIST)
   private Answer answer;
```

```
        @Entity
        public class PossibleAnswer implements Serializable {

           @Id
           @GeneratedValue(strategy = GenerationType.AUTO)
           private int possibleAnswerID;
           private String possibleAnswer;
           @ManyToOne//1 Mark (fetch = FetchType.LAZY)
            private Question question;
}
```

```
}
Answer.java
        @Entity//1 Mark
        @Table(name="tblAnswer")
        public class Answer implements Serializable
        {
          @Id
          @GeneratedValue(strategy=GenerationType.AUTO)
          private int answerID;
          private int fkQuestionID;
          private String correctAnswer;

          public Answer() {
  }
```

5. Create a local session bean called UserBean that will store user object and to validate user's logons using username and password and return an object of type User if the logons are valid otherwise return null. //7/

```
public class UserBean implements UserBeanService
{

  @PersistenceContext(unitName="examPU")1 Mark
  private EntityManager entityManager;

  @Override
  public void storeUser(User user) {1 Mark
    entityManager.persist(user);
  }

  @Override
  public User validateLogon(String username, String password)
  {
    String sql = "select user from User user where user.username like '"+username+"' and user.password '"+password+"'";1 Mark
```

```
        User user = null; 1 Mark
        Query query = entityManager.createQuery(sql); 1 Mark
        try
        {
          user = (User) query.getSingleResult();1 Mark
        }
        catch(Exception er)
        {
            user = null;
        }
        return user; 1 Mark
    }
}
```

6. Create a local session bean called QuestionBean that will retrieves all questions. The session bean deletes and find a question by a question id. Also, the session bean inserts a new question to the database. //14/

```
@Stateless
public class QuestionBean implements QuestionService
{

    @PersistenceContext(unitName="Assignment1Project-ejbPU")//1 Mark
    private EntityManager manager;
    @Override
    public List<Question> retrieveQuestion()
    {
          Query executeQuery = manager.createQuery("Select question from Question question");//1 Mark
             List<Question> list;
          list = (List<Question>)executeQuery.getResultList();//1 Mark
          for(Question question: list)
          {
               question.getPossibleAnswers().size();//1 Mark
          }
          return list;
    }
    @Override
    @TransactionAttribute(TransactionAttributeType.REQUIRED) //1 Mark
    public void insertQuestion(Question question) {//1 Mark
        manager.persist(question); //1 Mark
    }



    @Override
    @TransactionAttribute(TransactionAttributeType.REQUIRED) //1 Mark
    public void deleteQuestion(Question question) {
        Question questionFind = manager.find(Question.class, question.getQuestionID());//1 Mark

        if (questionFind != null) //1 Mark
        {
           manager.remove(question); //1 Mark
        }
    }

    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")
    @Override
    public Question FindQuestions(int questionID) {//1 Mark
        Question question = manager.find(Question.class, question.getQuestionID());//1 Mark
          return question;//1 Mark
    }
}
```

**Question 2** **(36)**

1. Create a message driven bean called LogonMDB that reads a text message 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 a session bean named UserBean to validate the logons. If the user is found the LogonMDB will publish an object of type

User otherwise it will publish “the user is not found” message to a queue named “jms/logonQueue” as a confirmation.
//11/

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

  @Override
  public void onMessage(Message message)
  {
    if (message instanceof TextMessage)
    {
      Connection connect = null;
      try {
        TextMessage txtMsg = (TextMessage) message;
        String[] logons = txtMsg.getText().split("#");1 Mark
        //Validate logons
        User user = userBean.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); 1 Mark
         //determine

         if (user!= null)
         {
          ObjectMessage objMsg = session.createObjectMessage();1 Mark
```

```
                objMsg.setObject(user); 1 Mark
                publish.send(objMsg);
              }
              else
              {
                txtMsg.setText("user is not found");1 Mark
                publish.send(txtMsg); 1 Mark
              }

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

}
```

2. Create another message driven bean called ResultsSubscriberMDB that subscribes to a topic named "jms/leanersPassedTopic". The message driven bean will read any changes made on the "jms/leanersPassedTopic" topic and display all leaners who passed on a console screen. //7/

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "clientId", propertyValue = " jms/leanersPassedTopic"),
    @ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = " jms/leanersPassedTopic "),
    @ActivationConfigProperty(propertyName = "subscriptionDurability", propertyValue = "Durable"),
    @ActivationConfigProperty(propertyName = "subscriptionName", propertyValue = " jms/leanersPassedTopic "),
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Topic")
})
public class ResultsSubscriberMDB implements MessageListener {1 Mark


    @Override
    public void onMessage(Message message) 1 Mark
    {
        if(message instanceof ObjectMessage) 1 Mark
        {
          try {
            ObjectMessage objMsg = (ObjectMessage) message; 1 Mark
            User user = (User) objMsg.getObject();1 Mark
            if (user != null) 1 Mark
            {
                System.out.println(user.toString());1 Mark
            }
          } catch (JMSException ex) {
            Logger.getLogger(ResultsSubscriberMDB.class.getName()).log(Level.SEVERE, null, ex);
          }

        }
    }

}
```

3. Create a stateful session bean named MarkingBean that will initialise the questions and sign totals data members and allow a user to mark a question and determine results based on sign group category condition given in table 1. The session beans will mark each question using the markQuestion() method which accepts the questionID and possible answer. If the possible answer is correct, the markQuestion() method will set the "mark" data member of Question object to "1" otherwise it set it as "0" and it must set the isAswered to "true", finally it will increment the sign group total by "1" according to sign group category. The session bean should determine whether a leaner "passed" or "failed" by calculating sign group averages for each sign group totals" //18/

```
@Stateful
public class MarkingBean implements MarkingService
{

   private List<Question> questions;
   private int totalControlSign;
   private int totalCommandSign;
   private int totalProhibitionSign;
   private int totalReservationSign;


   @Override
   @PostConstruct
   public void initialize()
   {
     this.questions = new ArrayList<Question>();
     totalControlSign = 0;
     totalCommandSign = 0;
     totalProhibitionSign = 0;
     totalReservationSign = 0;
   }

   @Override
   public void markAquestion(int questionID, String givenAns)
   {
       int count=0;
        for (Question question : questions) //1 Mark
        {
           if (questionID == question.getQuestionID())
           {
              if(question.getAnswer().getCorrectAnswer().equals(givenAns)) //1 Mark
              {
                 if (question.getSignGroup().equals("control"))//1 Mark
                 {
                   totalControlSign++;//1 Mark
                 }
                 else if (question.getSignGroup().equals("command"))
                 {
                  totalCommandSign++;
                 }
                 else if (question.getSignGroup().equals("prohibition"))//1 Mark
```

```
                    {
                     totalProhibitionSign++;//1 Mark
                   }
                    else if (question.getSignGroup().equals("reservation"))//1 Mark
                    {
                     totalReservationSign++;//1 Mark
                    }
                     question.setMark(1);                    //1 Mark

                }
                else
                {
                    question.setMark(0);                 //1 Mark
                }
                question.setIsAnswered(true); //1 Mark
                questions.set(count, question);
            }
            count++;
        }

    }

    @Override
    public String determineResult()
    {
        double totalControl=0, totalCommand=0, totalProhibition=0, totalReservation=0;

        for(Question question :questions) //1 Mark
        {
         if (question.getSignGroup().equals("control"))//1 Mark
            {
              ++totalControl;
            }
            else if (question.getSignGroup().equals("command"))//1 Mark
            {
```

```
            ++totalCommand;
          }
          else if (question.getSignGroup().equals("prohibition"))//1 Mark
          {
            ++totalProhibition;
         }
          else if (question.getSignGroup().equals("reservation"))
          {
            ++totalReservation;
          }
      }

      if ((totalControlSign / totalControl * 100) >= 90 && (totalCommandSign / totalCommand * 100) >= 90  && (totalProhibitionSign /
totalProhibition * 100) >= 70 && (totalReservationSign /totalReservation * 100)  >= 70) //1 Mark
      {
          return "Passed";//1 Mark
      }
      else
      {
          return "Failed";//1 Mark
      }
  }

  @Override
  public void destroyObjects()
  {
    questions.clear();
  }

  @Override
  public void populateList(List<Question> questions) {
      this.questions = questions;
  }

  @Override
```

```
    public List<Question> getQuestions() {
        return questions;
    }

    @Override
    public int getTotalControlSign() {
        return totalControlSign;
    }

    @Override
    public int getTotalCommandSign() {
        return totalCommandSign;
    }

    @Override
    public int getTotalProhibitionSign() {
        return totalProhibitionSign;
    }

    @Override
    public int getTotalReservationSign() {
        return totalReservationSign;
    }
}
```

## Question 3 (46)

1. Create a controller class called LogonServlet that override the doPost method to do the following: //20/
   a. The doPost method will accept the username and password to validate a user. The method will publish the combined logons "username#password" to a queue named "jms/logonQueue". Then it will wait for confirmation. If the confirmation is of type User and the user is of type "admin" redirect a user question.jsp to add questions, if the user is of type "leaner", the servlet should read all question using the QuestionBean session bean and

populate the question list of the MarkingBean session bean with values and store it on a “session” and redirect the user to “answeK53.jsp”, otherwise redirect a user to login.html.

```
@WebServlet(urlPatterns = {"/LoginServlet"})
public class LoginServlet extends HttpServlet {
@Resource(mappedName="jms/logonQueueFactory")
ConnectionFactory connectFactory;
@Resource(mappedName="jms/logonQueue")
Queue queue;
@EJB
  private QuestionBeanService questionBean;

     @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException
   {
      String username, password, logons;
      username = request.getParameter("username");
      password = request.getParameter("password");
      logons = username+"#"+password; 1 Mark

      //Connection
      Connection connection = null;
      Connection connection2 = null;
   try {
       connection = connectFactory.createConnection();
       Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); 1 Mark
       TextMessage txtMsg = session.createTextMessage();1 Mark
       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();
       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()) {

            if (message instanceof TextMessage) 1 Mark
            {
               TextMessage txtMsgOut = (TextMessage) message;
               response.sendRedirect("login.html");1 Mark

            }
            else if (message instanceof ObjectMessage)
            {
               ObjectMessage objMsg = (ObjectMessage) message;
               User user = (User) objMsg.getObject();
               request.getSession().setAttribute("user", user); 1 Mark
                 if (user.getUserType().equals("admin"))
                     {
                       response.sendRedirect("question.jsp");              1 Mark
                      }
                  if (user.getUserType().equals("admin"))1 Mark
                 {
                     response.sendRedirect("question.jsp");              1 Mark
                     List<Question> questions = questionBean. retrieveQuestion();1 Mark
                     InitialContext context = new InitialContext();
                     MarkingBeanService markBean = (MarkingBeanService) context.lookUp("za.ac.tut.marking. MarkingBeanService");1 Mark
                    markBean.populateQuestion(questions); 1 Mark
                    request.getSession().setAttribute("markingBeaning",markBean); 1 Mark
                   response.sendRedirect("answerK53.jsp");
            }


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

    }
```

```
}
```

b. Create the answerK53.jsp to display all the questions, each with their questionID, question, possible answers using a radio button and submit button named "answer question". Each time a question is answered it will be marked and displayed as either as 1 or 0 based on the given answer. All answered question must not display the radio button of possible answers and "answer question" button. //10/

Answer.jsp

```
<%@page import="za.ac.tut.dao.PossibleAnswer"%>
<%@page import="za.ac.tut.dao.Question"%>
<%@page import="za.ac.tut.session.MarkingService"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Answer the question</h1>
    <table>


    <%
        MarkingService serviceBean = (MarkingService) request.getSession().getAttribute("markingContext");//1 Mark

      if (serviceBean != null) //1 Mark
      {
```

```
                    for(Question question:serviceBean.getQuestions())//1 Mark
                    {
               %>
               <form action="AnswerK53" method="POST">
                 <input type="hidden" name="questionID" value="<%=question.getQuestionID()%>">//1 Mark
                 <tr>
                   <td><%=question.getQuestion()%></td>//1 Mark
                   <td> Select the possible Answer <br/>

                     <%
                       for (PossibleAnswer answer :question.getPossibleAnswers())//1 Mark
                       {
                         if (!question.isIsAnswered())//1 Mark
                         {
                      %>

                       <input              type="radio"              name="givenAns"              value="<%=answer.getPossibleAnswer().charAt(0)%>">
          <%=answer.getPossibleAnswer()%>//1 Mark <br/>
                      <%
                           }
                       }

                     %>
                      Your Mark :<%=question.getMark()%>//1 Mark
                     <input type="submit" value="Answer Question" name="select" />
                   </td>
                 </tr>
               </form>

               <%
                    }


                  }

              %>
              <tr>
                <td>
               <p> Total Command Marks is :<%=serviceBean.getTotalCommandSign() %></p>//1 Mark
              <p> Total Control Marks is :<%=serviceBean.getTotalControlSign() %></p>
```

```
                    <p> Total Prohibition Marks is :<%=serviceBean.getTotalProhibitionSign() %></p>
                    <p> Total Command Marks is :<%=serviceBean.getTotalReservationSign() %></p>
                    <P> Result:<%=serviceBean.determineResult() %> </p>
                        </td>
                    </tr>

                    </table>
                </body>
            </html>
```

2. Create another controller class called Answer53Servlet that override the doPost method to do the following: //16/
   - The doPost method will accept given answer and the question id from answerK53.jsp, the doPost() method will determine whether the provided answer is correct or not, using the MarkingBean session bean. Then, the dePost() method will store the updated MarkingBean objects to the "session" again, and redirect a learner back to the answerK53.jsp.
   - When all the questions are answered, the doPost() method will publish each leaner/user who passed the learners to the "Jms/leanersPassedTopic" topic use the publishPassedLeaner(). The method receives an object of type User and publish it on the "jms/leanersPassedTopic" topic.

```
        @WebServlet(name = "AnswerK53", urlPatterns = {"/AnswerK53"})
        public class AnswerK53 extends HttpServlet {

            @EJB
            QuestionService questionBean;
            @Resource(mappedName="jms/ leanersPassedTopicFactory ")
             private ConnectionFactory factory;
             @Resource(mappedName="jms/leanersPassedTopic ")
              private Topic topic;


```

```
/**
 * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
 * methods.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    try {
        response.setContentType("text/html;charset=UTF-8");
        InitialContext initialContext = new InitialContext();
        MarkingService serviceBean = (MarkingService) request.getSession().getAttribute("markingContext");//1 Mark

        if (serviceBean == null)
        {
            serviceBean = (MarkingService) initialContext.lookup("za.ac.tut.session.MarkingService");//1 Mark
            //populate the session bean
            serviceBean.populateList(questionBean.retrieveAllQuestions());//1 Mark
        }

        String select = request.getParameter("select");
        if (select != null)
        {
            if (select.equals("Answer Question"))
            {
                int questionID = Integer.parseInt(request.getParameter("questionID"));//1 Mark
                String answer = request.getParameter("givenAns");
                serviceBean.markAquestion(questionID, answer);          //1 Mark
                if (serviceBean. determineResult().equals("Passed"))//1 Mark
                  {
                      User user = request.getSession().getAttribute("user");//1 Mark
                       publishPassedLeaner(user); //1 Mark
                  }
```

```
          }

         request.getSession().setAttribute("markingContext", serviceBean);     //1 Mark
          response.sendRedirect("answerK53.jsp");//1 Mark

        } catch (NamingException ex) {
          Logger.getLogger(AnswerK53.class.getName()).log(Level.SEVERE, null, ex);
        }



      }
public void publishPassedLeaner (User user)
  {
    Connection connection = null;
    try
    {
      connection = factory.createConnection();//1 Mark
      Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); 1 Mark
      ObjectMessage objMsg = session.createObjectMessage();1 Mark
      objMsg.setObject(user); 1 Mark
      MessageProducer msgProducer = session.createProducer(topic); 1 Mark
      msgProducer.send(objMsg); 1 Mark

    } catch (JMSException ex) {
      Logger.getLogger(ProductBean.class.getName()).log(Level.SEVERE, null, ex);
    }
    finally
    {
      try {
        connection.close();
      } catch (JMSException ex) {
        Logger.getLogger(ProductBean.class.getName()).log(Level.SEVERE, null, ex);
      }
    }
  }
```