|  |  |
| --- | --- |
| Assessment Title | Test |

|  |  |
| --- | --- |
| Competency Details | |
| Unit code/s and title/s | ICTPRG556 Implement and use a model view controller framework |
| Qualification code/s and title/s | ICT50120 Diploma of Information Technology (Advanced Programming) |
| Business unit/Work group | Business and Arts/ IT Studies |

|  |  |
| --- | --- |
| Complete for eachstudent | |
| Student Details | |
| Student name |  |
| Student SIS ID |  |
| Assessment date |  |
| Reassessment | This is a re-attempt |

|  |
| --- |
| Guidelines |
| Ensure you have read the Student Assessment Instructions related to this assessment task before completing the task described in the Task Details section below.  The task details are provided for you to use during the assessment. This document must be returned to your assessor at the end of the assessment. **You are not permitted to keep this document or make a copy.**  All project/Document files once uploaded must be deleted from the computer you used to complete this assessment. |

**Online Auction System - Prototype**

IT Works has hired you as a senior software engineer to refactor/reengineer an online auction prototype application for their client, The Acme Auctions. The client is currently operating a web based JavaEE application that offers its customers the following online services as shown in the Use Case model below:



The existing application was created using NetBeans and the required project files are available to you in a repository titled *Acme Auction Resources*. The existing application prototype has some major short comings as all the web pages were static HTML pages with hard coded data which made it difficult to maintain.

It was decided to re-engineer this prototype based on the following requirements

The analyst/designers at IT Works have come up with the following requirements which you need to interpret and implement according to the Requirements below.

The DAO and model classes have been fully implemented for you by the previous team

**Requirements**

**Task 1: Creating the Database/Tables and Connection Pool**

**NOTE: If your using Glassfish version 4.1.1, IGNORE step (1) and create the tables using the sample database that comes with the JavaDB database server in Netbeans.**

1. Open the existing prototype project in Netbeans Select the *JavaDB* database and create a new database *AuctionDB* with *user -> user1* and *password -> password* ***(IGNORE THIS IF USING THE sample DB)***
2. Use the *TBids, TItems and TMembers.grab* files found in the *ICTPRG556 - ASDS - Test Resources* folder on Learn to create the tables using the above database.
3. Create a *jdbc* connection pool and name it ‘*jdbc/Auction\_DB’* (***NOTE: use ‘jdbc/sample’ if you created the tables in the sample database***) Use this connection pool to access the database.
4. Populate the *TMembers* and *TItems* tables with values seen in the *CustomerBids.html* and *ItemBids.html* pages.

**Task 2 – Run the existing prototype and fix any existing errors/resource**

**dependency issues**

Before you run the application, compile the project and fix any compile time errors and resource dependency issues. Once fixed, run the application from the *Main.html* to make sure that the services all work.

**Task 3 – Re-engineer the existing application using MVC Framework**

Re-engineer static HTML Pages that have hard coded data into dynamic pages as JSP’s.

For example, the *Bids.html* page has *Member* and *Item* information hard coded into the page. The same is observed with the *ItemBids.html* and *CustomerBids.html* pages.

This architecture has some obvious drawbacks.

Your task is to re-engineer this application based using an MVC approach by following the recommendations below

Convert the following HTML files to JSPs: (**NOTE – If you complete just the Bids.jsp that would be enough the pass this assessment)**

1. *ItemBids.html* -> *ItemBids.jsp* (populated with Items dynamically)
2. *CustomerBids.html* -> *CustomerBids.jsp* (populated with Customer)
3. *Bids.html* -> *Bids.jsp* (populated with Members and Items dynamically
4. Use the Java Bean class models to bind to the relevant JSP views

Use the MVC project template available in Netbeans and put the different MVC components into their relevant packages. Modify the Servlet to redirect control to the relevant JSP pages as views

**The analyst design team has outlined the following workflow(s) to re-engineer the prototype application**

1. Create the relevant JSP’s that and dynamically populate them with relevant data from the database created earlier. For example, the Bids.jsp will be populated with *Member* and *Item* data from the *TMembers* and *TItems* tables respectively.
2. Modify your *AuctionController* Servlet to provide the required data to each of the JSPs. The Servlet should use the appropriate methods in the *DAO* to get the required data from the database. Use the comments in the Servlet as a guide to make the required changes.
3. You will need to modify the *Main.html* page to access the Servlet, through each of the hyperlinked services. Modify the servlet to handle GET methods from this page using the doGet() method of the Servlet. All requests from the JSPs to Servlet will be POST methods that would be handled by the Servlets doPost() method**. NOTE: Use the servlets processRequest( ) to process both the doGet and doPost methods**
4. Run the modified application and demonstrate to the project lead that the application works after the re-engineering process.
5. After the demo and if required the fixing of any issues, upload your re-engineered Netbeans project to Learn using the Test link provided.