UNIVERSITY OF MAURITIUS

FACULTY OF ENGINEERING

FIRST SEMESTER EXAMINATIONS

NOVEMBER/DECEMBER 2010

PROGRAMME	**BSc (Hons) Computer Science** **BSc (Hons) Information Systems**		
MODULE NAME	**Web Technologies II**		
DATE	**Tuesday** **30 November 2010**	**MODULE CODE**	**CSE 2041(3)**
TIME	**09.30 – 11.30 Hrs**	**DURATION**	**2 hours**
NO. OF QUESTIONS SET	**4**	**NO. OF QUESTIONS TO BE ATTEMPTED**	**4**

INSTRUCTIONS TO CANDIDATES

Answer All Questions

Read the entire set of questions before attempting the questions

Questions 1 – 3 relate to the feedback case study

WEB TECHNOLOGIES II – CSE 2041(3)

Consider a feedback application that is being developed to collect feedback from students about different modules. The following schema has been defined for the feedback database, implemented in MySQL.

classsizes (classsize, classsizedesc)
feedbacks (email, modulecode, moduleyear, classsize, delivery, labs, programme othercomments, moderation)
modules(modulecode, moduledesc)
users(username, pass)

Notes:

- *classsize* and *modulecode* in the *feedbacks* table are foreign keys from the classsizes and *modules* tables respectively.
- The *moderation* field (of type enum) contains values 'a' for approved, 'p' for pending, and 'r' for rejected. The default value is 'p'.
- The values for the *classsizes* table are '-1', '1' , '2' , '3' in the *classsize* field for 'poor', 'adequate', 'good' and 'excellent' respectively in the *classsizedesc* field.
- Delivery and labs fields are boolean and indicate whether improvements are required in the delivery and labs (a value of 1 indicates that improvement is required, 0 otherwise)

Question 1 [25 marks]

Management would like to know the average class size ratings, for each student and module. Therefore, a form was created to build the (email, modulecode) list (*Figure 1a*), which was then constructed into an XML format as shown in *Figure 1b*.

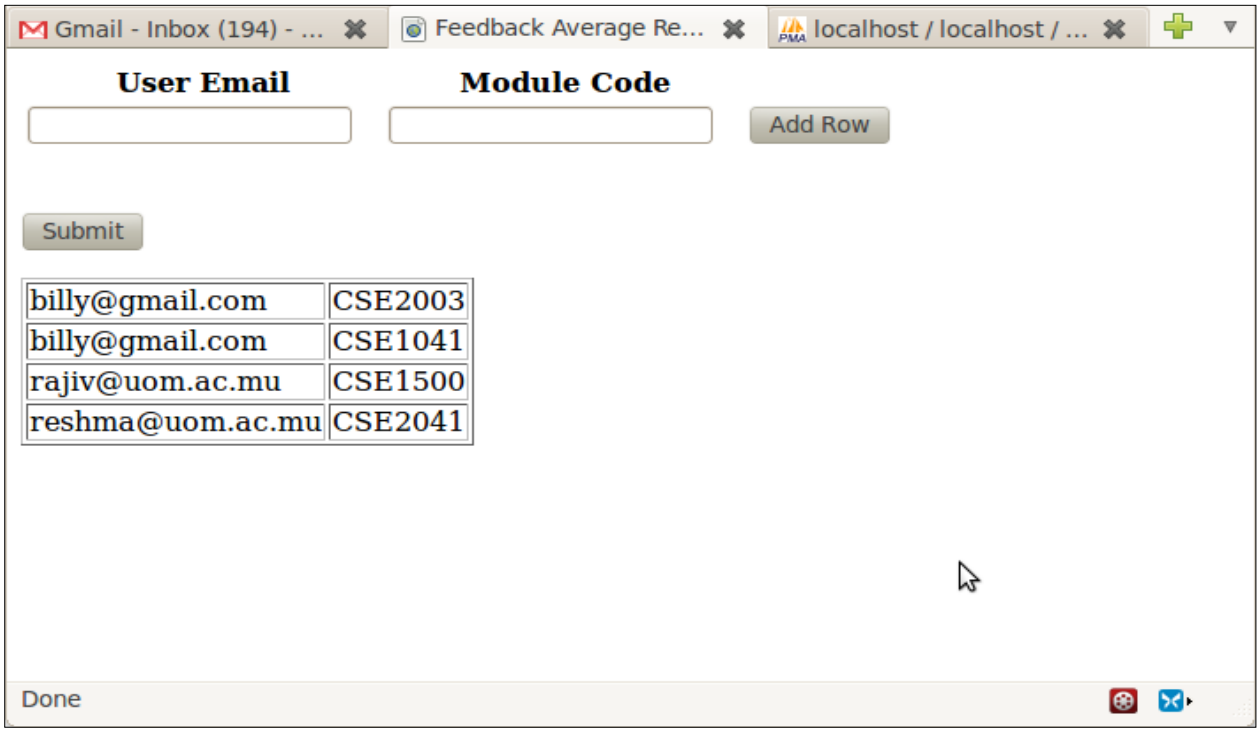

Figure 1a: Feeback.html

```
<modules>
      <query qdate="2010-01-01">
            <user>billy@gmail.com</user>
            <module>CSE2003</module>
      </query>
      <query qdate="2010-01-01">
            <user>billy@gmail.com</user>
            <module>CSE1041</module>
      </query>
      <query qdate="2010-01-01">
            <user>rajiv@uom.ac.mu</user>
            <module>CSE1500</module>
      </query>
      <query qdate="2010-01-01">
            <user>reshma@uom.ac.mu</user>
            <module>CSE2041</module>
      </query>
</modules>
```

Figure 1b

The XML tree in *Figure 1b* is stored in a hidden field, *txt_xml_queries*, and sent to the page *computeavg_pro.php* through the *POST* method.

.../Cont'd next page

Question 1 ***[Cont'd]***

Part of the codes for the page, *computeavg_pro.php,* is as follows:

```
<?php
 header('Content-Type: text/xml');
 $xmlstring = $_POST["txt_xml_queries"];

 //1 - missing codes for loading the xml string

 if(//2 - missing codes for validating against the xsd schema definition  )
   echo "Your xml is NOT valid";
 else
 {
  include("db_connect.php");

  //3 - missing codes for
  // - using the xml string and query the database for average feedbacks
  //4 - construct another xml string for displaying average feedbacks

   $xml_output = "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>";
   //5 - missing codes for
   //  - adding the stylesheet
   //  - adding the feedback xml string
  echo $xml_output;

mysql_close($con);
}//end else

?>
```

(a) You are required to fill in the ***missing codes*** for the page, *computeavg_pro.php*, ensuring the following:

(i) To load the XML string into a DOM Object

(ii) To validate the XML string above against the xsd schema definition defined in *moduleslist.xsd* (found in the same directory as the current page)

(iii) To query the feedback database, in order to get the email, module code and average classsize for each (email, module) from the xml tree.

You may find the following codes useful:

```
$sql_select="SELECT email, modulecode, avg(classsize) as avgsize FROM
feedbacks WHERE email='$user' and modulecode='$modulecode' GROUP BY email,
modulecode";

$Rs = mysql_query($sql_select);

$rows = mysql_fetch_array($Rs);
```

(iv) To construct the feedback xml tree in the following format:

.../Cont'd next page

Question 1 ***[Cont'd]***

```
<feedback>
<classsize>
      <email>billy@gmail.com</email>
      <modulecode>CSE2003</modulecode>
      <average>1.5</average>
</classsize>
<classsize>
      <email>billy@gmail.com</email>
      <modulecode>CSE1041</modulecode>
      <average>2.5</average>
</classsize>
<classsize>
      <email>rajiv@uom.ac.mu</email>
      <modulecode>CSE1500</modulecode>
      <average>-1</average>
</classsize>
<classsize>
      <email>reshma@uom.ac.mu</email>
      <modulecode>CSE2041</modulecode>
      <average>3</average>
</classsize>
</feedback>
```

Figure 1c: Feedback XML

(v) Finally to link the *feedback.xsl* (found in the same directory as the current page) stylesheet definition to transform the feedback XML. The default namespace for the stylesheet is http://www.w3.org/1999/XSL/Transform

[2+2+6+6+2 marks]

(b) Another way to transform an XML is to use an XSLT Processor. Write the codes for transforming the feedback XML string using the XSLT Processor and *feedback.xsl* file. **[7 marks]**

Question 2 [25 marks]

(a) You are required to write the schema definition, *moduleslist.xsd*, for the XML tree in *Figure 1b* from **Question 1(a)**, ensuring the following:

(i) there should be at least one *query* element.
(ii) the attribute *qdate* is of type date and should be explicitly defined as optional.
(iii) each *query* element can consist of only one *user* and one *module*.

You may provide any other assumption you feel applicable. The default namespace for the schema is http://www.w3.org/2001/XMLSchema

[10 marks]

.../Cont'd next page

Question 2 [*Cont'd*]

(b) The following page (Figure 2) is obtained after transforming the feedback XML from Question 1, ***Figure 1c*** above:

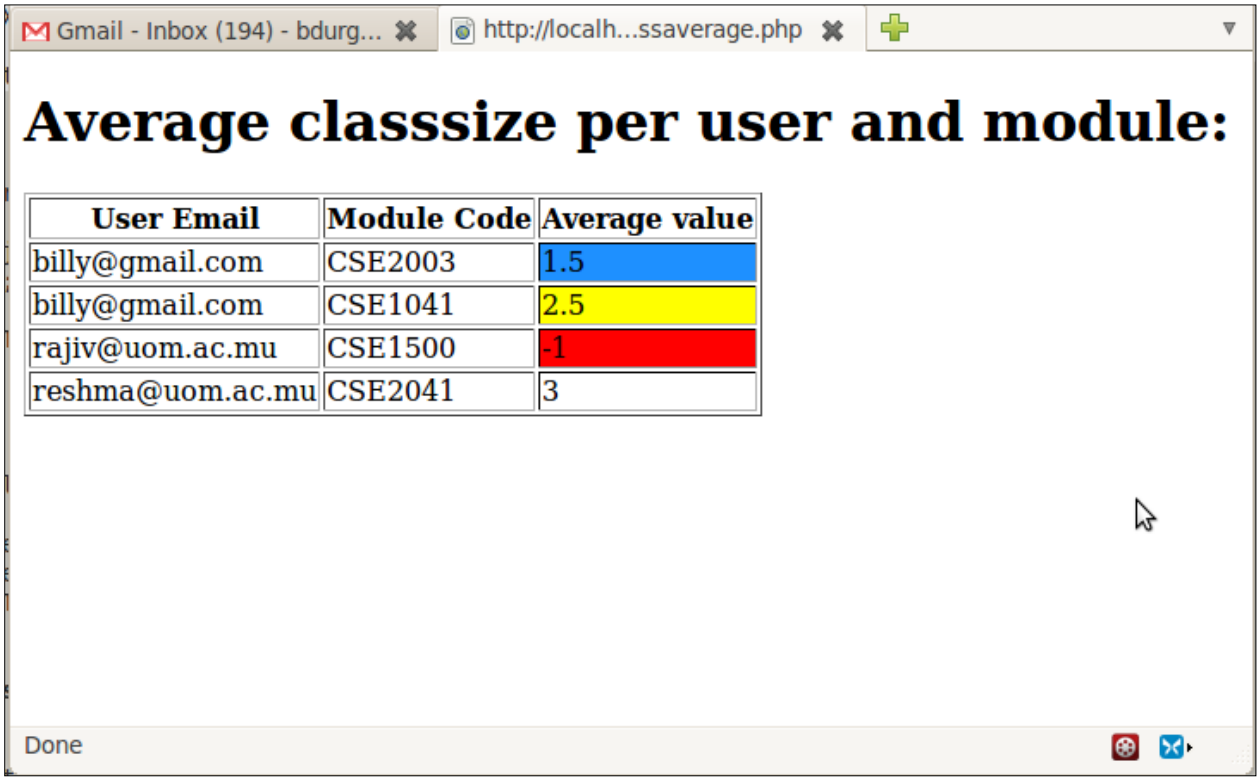

Figure 2: computeavg_pro.php

(i) You are required to give the codes for *feedback.xsl*, ensuring the following transformation:

Condition	Set Background of Average cell to
average < 1	red
1 ≤ average < 2	blue
2 ≤ average < 3	yellow
average ≥ 3	white

The default namespace for the stylesheet is http://www.w3.org/1999/XSL/Transform

[15 marks]

Question 3 [25 marks]

Consider the following page, *addmodule_xml3.html* (Figure 3a), augmented from the one that was discussed in class, that allowed users to add a series of *modulecode* and respective *moduledesc* to an XML, display the content of the XML into a table, then submit the xml as a string on clicking on *Submit*. On clicking the *Delete* button, the specific *modulecode* and *moduledesc* are deleted from the xml tree.

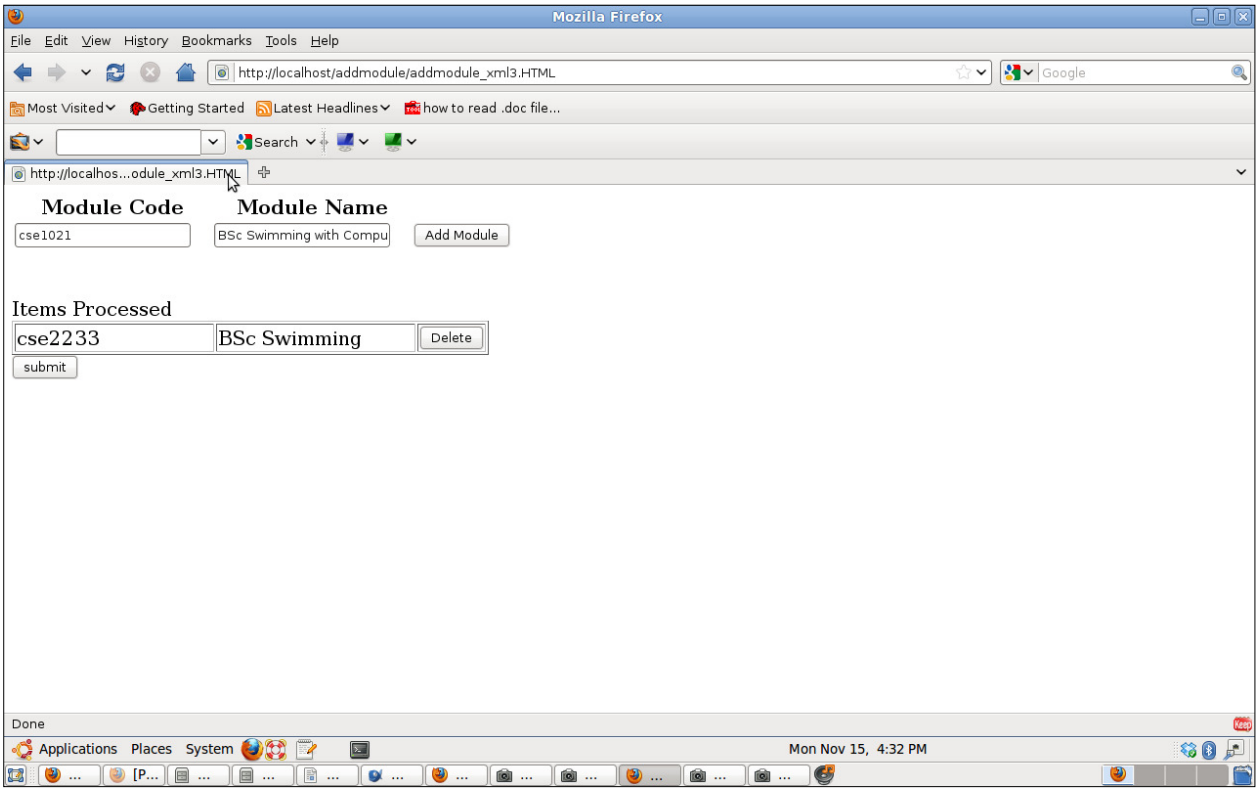

Figure 3a: addmodule_xml3.html

The codes for *addmodule_xml3.html* is as follows:

```
<html>
<head>
<script type="text/javascript">
<!--
  var count_modules=0;
  var xml_modules="";
  xml_modules= "<modules></modules>";

  var xmlDoc_modules = loadXMLString(xml_modules);

  function loadXMLString(txt)
  {
      //function that returns an DOM object instance (for both IE and
Firefox)

  }//end function loadXMLString

  function insertModule(modulecode, moduledesc)
 {
     modules_node=xmlDoc_modules.childNodes[0];
     module_node = xmlDoc_modules.createElement("module");
```

```
    modulecode_node =xmlDoc_modules.createElement("modulecode");
    moduledesc_node =xmlDoc_modules.createElement("moduledesc");

    module_node.appendChild(modulecode_node);
    module_node.appendChild(moduledesc_node);
    modules_node.appendChild(module_node);

    len = modules_node.childNodes.length;
    modulecode_node = modules_node.childNodes[len-1].childNodes[0];
    modulecode_node_val = xmlDoc_modules.createTextNode(modulecode);
    modulecode_node.appendChild(modulecode_node_val);

    moduledesc_node = modules_node.childNodes[len-1].childNodes[1];
    moduledesc_node_val = xmlDoc_modules.createTextNode(moduledesc);
    moduledesc_node.appendChild(moduledesc_node_val);
    return true;
}

function addtablerow()
{
  var modulecode=document.forms[0].txt_modulecode.value;
  var moduledesc = document.forms[0].txt_moduledesc.value;
  insertModule(modulecode, moduledesc);
  document.forms[0].txt_modulecode.value="";
  document.forms[0].txt_moduledesc.value="";
  //recompute display
  displayItems()
  return false; //to prevent the data from being submitted to the server
}//end function addtablerow()

function displayItems()
{
   table_str= "Items Processed <br/><table border=1>";
   modules_div = document.getElementById("div_modules");
   modules_node=xmlDoc_modules.childNodes[0];
   len = modules_node.childNodes.length;

   for (i=0; i<len ;i++)
   {
     table_str = table_str + "<tr><td width=200>";
     module_code =
    modules_node.childNodes[i].childNodes[0].firstChild.nodeValue;
     table_str = table_str + module_code +"</td><td width=200>";
     module_desc =
    modules_node.childNodes[i].childNodes[1].firstChild.nodeValue;
     table_str = table_str + module_desc +"</td><td><button id="+i+"
    onclick=deleteNode(this)>Delete</button></td></tr>";
   }
   table_str = table_str + "</table>";
   modules_div.innerHTML = table_str;
}//end function displayItems()

function deleteNode(obj)
{
   modules_node=xmlDoc_modules.childNodes[0];
   //get node to be deleted
   i= obj.id;
   module_node = modules_node.childNodes[i];
   modules_node.removeChild(module_node);
   displayItems();
}

function encode_Items()
```

```
  {
      //adds the content of the DOM object instance into the hidden field

      //txt_xml_modules

  }//end function encode_Items()

  function getBrowser()
  {
    //code to determine browser version
  }//end function getBrowser()

-->
</script>
</head>
 <body>
 <form action=addmodulexml_pro.php method=post onsubmit="return
encode_Items()">
  <table id=table_modules>
    <tr>
     <th width=200>Module Code</th>
     <th width=200>Module Name</th>
     <th>
    </tr>
    <tr>
     <td><input type=text name=txt_modulecode></td>
     <td><input type=text name=txt_moduledesc></td>
     <td><button onclick="return addtablerow()">Add Module</button></td>
    </tr>
  </table>
  <br><br>
  <div id="div_modules"></div>
  <input type=submit value=submit onclick='return encode_Items()'>
 <input type=hidden name=txt_xml_modules id=txt_xml_modules>
 </form>
</body>
</html>
```

Briefly explain why the lines in the function *insertModule()* highlighted in bold are redundant.

```
     len = modules_node.childNodes.length;
     modulecode_node = modules_node.childNodes[len-1].childNodes[0];
..
     moduledesc_node = modules_node.childNodes[len-1].childNodes[1];
```

[5 marks]

You want to now change the functionality of the page, such that users can update the xml tree in case they made errors. ***Figure 3b*** below illustrates the desired behavior.

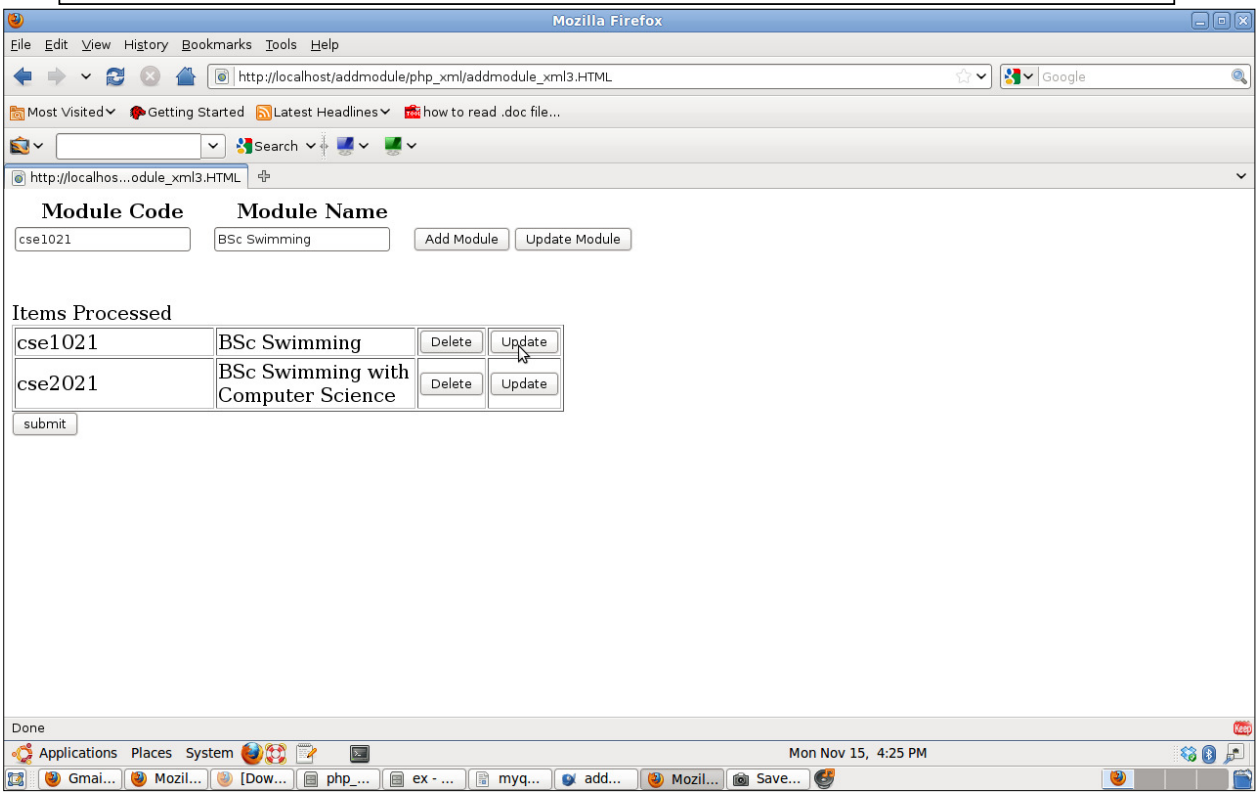

Figure 3b: addmodule_xml3.html with enhanced behavior

You want the following properties for the form:
On loading the page, the user can only add a module and clicking on *UpdateModule* button should have no effect. Once a module has been added, the user can now click on the *Update* button for that respective module. This will load the *modulecode* and *moduledesc* into the text fields above. Then, the *UpdateModule* button is active and the *Add Module* button becomes inactive. After the user has made necessary changes, she can click on *UpdateModule* so that the XML tree is updated and the table displaying the modules is refreshed. The text fields are also cleared. The *Add Module* button becomes active while *UpdateModule* becomes inactive.

Write the codes for the above enhanced behavior. You need not rewrite the whole set of codes, but indicate where your changes fit. Make sure that:

(i) You add the codes to include the *Update* buttons to your page and associate the appropriate events.
(ii) Ensure that the selected *modulecode* and *moduledesc* to modify are displayed in the text fields.
(iii) Make sure that when you click on *UpdateModule*, the **relevant** node is updated (keep track of which node was chosen), the text fields cleared and the display refreshed.
(iv) Make sure that only either "*add*" mode is enabled or "*update*" mode is enabled. *Hint*: You may want to define a global boolean variable, *update_mode*

[4 x 5 marks]

Question 4 [25 marks]

(a) Explain the following terms by emphasizing on their importance in the area of web service technology:

i. WSDL
ii. UDDI
iii. SOAP

[3 x 2 marks]

(b) Consider the following WSDL file, available at "http://lyrics.wikia.com/server.php?wsdl" which is used to look for lyrics of songs.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd" xmlns:tns="urn:LyricWiki"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:LyricWiki">
<types><xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:LyricWiki">

 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/"/>

 <xsd:complexType name="LyricsResult">
  <xsd:all>
   <xsd:element name="artist" type="xsd:string"/>
   <xsd:element name="song" type="xsd:string"/>
   <xsd:element name="lyrics" type="xsd:string"/>
   <xsd:element name="url" type="xsd:string"/>
  </xsd:all>
 </xsd:complexType>
</xsd:schema>
</types>

<message name="checkSongExistsRequest">
 <part name="artist" type="xsd:string"/>
 <part name="song" type="xsd:string"/>
</message>

<message name="checkSongExistsResponse">
 <part name="return" type="xsd:boolean"/>
</message>

<message name="getSongRequest">
 <part name="artist" type="xsd:string"/>
 <part name="song" type="xsd:string"/>
</message>
```

```
<message name="getSongResponse">
 <part name="return" type="tns:LyricsResult"/>
</message>

<portType name="LyricWikiPortType">
 <operation name="checkSongExists">
  <documentation>Check if a song exists in the LyricWiki database yet
  </documentation>
  <input message="tns:checkSongExistsRequest"/>
  <output message="tns:checkSongExistsResponse"/>
 </operation>

 <operation name="getSong">
  <documentation>Get the lyrics for a LyricWiki song with the exact
artist and song match
  </documentation>
  <input message="tns:getSongRequest"/>
  <output message="tns:getSongResponse"/>
 </operation>

</portType>

<binding name="LyricWikiBinding" type="tns:LyricWikiPortType">
 <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="checkSongExists">
   <soap:operation soapAction="urn:LyricWiki#checkSongExists"
style="rpc"/>
     <input>
      <soap:body   use="encoded" namespace="urn:LyricWiki"
encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/"/>
     </input>
     <output>
      <soap:body use="encoded" namespace="urn:LyricWiki" encodingStyle=

      "http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
  </operation>

 <operation name="getSong">
  <soap:operation soapAction="urn:LyricWiki#getSong" style="rpc"/>
   <input><soap:body use="encoded" namespace="urn:LyricWiki"
encodingStyle=    "http://schemas.xmlsoap.org/soap/encoding/"/>
   </input>
   <output>
    <soap:body use="encoded" namespace="urn:LyricWiki" encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/"/>
   </output>
  </operation>
 </binding>
 <service name="LyricWiki">
  <port name="LyricWikiPort" binding="tns:LyricWikiBinding">
   <soap:address location="http://lyrics.wikia.com/server.php"/>
  </port></service>
</definitions>
```

You want to call the Web Service for the following using SOAP:

.../Cont'd next page

Question 4 ***[Cont'd]***

Assume that the user submits the artist name and song to the page *music.php*. *music.php* **first** calls the web service to determine whether the artist and song combination exists. If the latter does not exist, an appropriate error message displayed. If the latter exists, then it calls the web service to get the lyrics of the song. The details are displayed as shown in figure 4 (which contains the results for artist "Eagles" and song "Hotel of California"). The result also contains the hyperlink to the page containing the lyrics.

Write the codes for the above specs.

[19 marks]

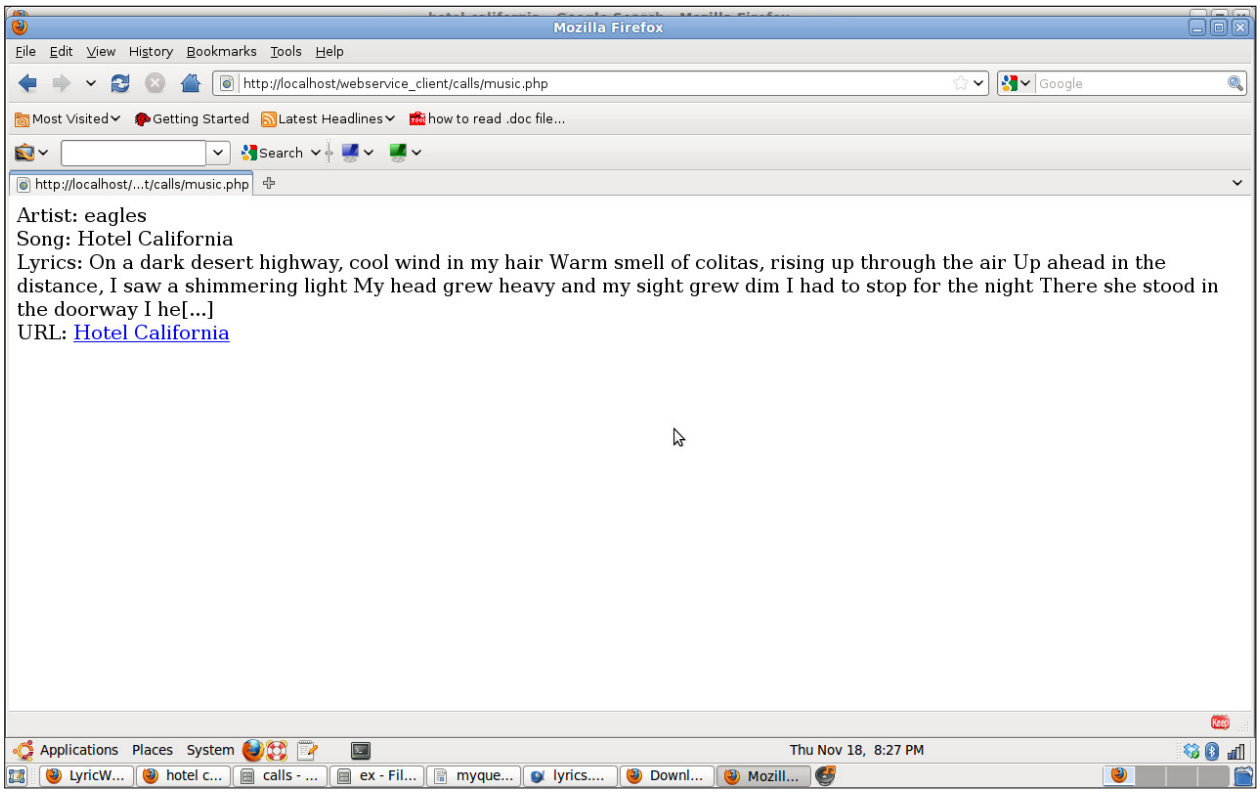

Figure 4 : Results to calling music.php

You may find these codes, that had been defined for using the GoogleRandomSearch WS useful.

```
<?php
  require_once('nusoap/nusoap.php');
  $key = $_POST['txt_key'];

  $wsdl = "http://ghettodriveby.com/soap/?wsdl";
  $client = new nusoap_client($wsdl, 'wsdl');
  //input requires only one parameter, called word, type string


  $param = array('word'=>$key);
  $result = $client->call('getRandomGoogleSearch', $param);
```

```
      if ($client->fault) {
    echo '<p><b>Fault: ';
    print_r($result);
    echo '</b></p>';
   } else {
    // Check for errors
    $err = $client->getError();
    if ($err) {
        // Display the error
        echo '<p><b>Error: ' . $err . '</b></p>';
    } else {
        // Display the result
        //output consists of 2 elements, one of which is called image

         $img_src=$result['image'];
        echo "<img src= '$img_src' />";
    }
}
?>
```

END OF QUESTION PAPER

/ph