Online Shopping

Assignment #2

LONDON SOUTH BANK UNIVERSITY

Author: Georgi Butev 3024421

Lecturer: Dr Perry Xiao

Class: Advanced Network Programming

Course: MS Computer Systems and Networking (1024 FT)

Faculty: ESBE

University: London South Bank University, London, SE1 6LN.

Email: butevg@lsbu.ac.uk

Submission Deadline: 11/05/2012

Table of Contents

1. Introduction

This project is similar to two categories in web development known as Content Management Systems and Electronic Commerce. The first category of development aims to efficiently store and manage data in a relational database rather than a flat file database (e.g. text file, excel table). The second category aims to provide the buyer with all that is necessary to place and order and checkout using his or her online bank account (e.g. PayPal) or credit card. Most notable such web sites are Amazon which stores books information in a database and allows users to buy a book of their choice and iTunes which stores songs information in a database and allows users to buy a song of their choice. These platforms all have a common infrastructure – server side logic written in PHP, JSP, ASP, database storage engine such as MySQL, PostgreSQL, DB2, Oracle, web server such as Apache, Internet Information Server, and user side frontend to the CMS written in HTML, JavaScript, and Flash according to [1]. There are two extremely popular web development stacks based on the previously mentioned infrastructure – Linux Apache MySQL PHP and Windows, IIS, Microsoft SQL Server, ASPNET. The first instance is popular because it is readily available, free of charge and well documented across forums and blogs. The second instance is popular because of Microsoft's monopoly over the IT market and slight benefits in terms of professional support/documentation and performance.

Naturally there are other web sites which use Sun Solaris UNIX, Apache, MySQL/Oracle, and Java JSP/Servlets. For example BlackBoard, Flickr, Lloyds TSB, and other companies which require scripting language such as JSP coupled with fast performing Java beans code to suite their corporate business needs as suggested by [1].

1.1 Servlets

They are Java based technology used to construct dynamic web pages. They are dynamic because unlike HTML pages which contents do not change based on user input, Servlets respond to user requests. The response is made with the Common Gateway Interface. The client does not need to have the latest Java Runtime installed to receive CGI output. The reason for that is Servlets are Java source code parsed by the web server and translated into HTML syntax according to [2].

The user interaction is done using HTML forms. These form fields could be text, password, select option lists, radio buttons, and checkboxes. The form is sent for processing using the POST and GET methods. The difference between the two is that the GET method displays all carried data in the address bar while POST hides the data from the user. Hiding sensible data like passwords is always recommended.

One of the major concerns with CGI programming is how slow it could become serving lots of clients. The reason for this is because CGI spawns a new process for each new request which means that the system resources will be overwhelmed. Servlets have a solution to this problem. All Servlets are run by a single instance of the JVM. Furthermore servlets use light-weight processes (threads) which have minimal impact on memory and processing power while at the same time sharing commonly used resources from all other threads according to [2].

In order to bypass CGI, Servlets use a container which runs alongside the web server. There are three popular server software solutions – Apache, Tomcat, and GlassFish.

The Java Servlets API states that all Servlets are extended from HttpServlet which can request with ServletRequest library or respond with ServletResponse library. When working with HTML forms such as text fields, text areas, and select options the servlet gets the data using request.getParameter("surname") for example. Java Servlets can be used to collect data from submit forms, processes them with platform methods, and store them in a database. To do this the servlet needs a DB connector/driver and special prepare statement for SQL queries. The connector requires the address of the DB server, valid username, valid password, and selection of existing database table.

The user session can be tracked with HTML form hidden values and cookies which store user information useful for web site usability. However their usage is not recommended because they are stored in plain text and can be easily stolen by malicious users on the network.

1.2 Java Server Pages

They were introduced to deal with some of the Servlets drawbacks and ultimately replace this technology with a more compact and script-like version, similar to the market leaders – PHP, Perl, JavaScript. The biggest shortcoming of Servlets is that the source code has to be a compiled .java file which has to include cumbersome HTML tags. In contrast JSP pages can be pure Java, pure HTML, or a reasonable mixture of the two. For example to display a variable in the HTML code, only the following is needed **<%= java.util.Date( ) %>**. Rather than being compiled manually by the developer, JSPs are parsed by the web server running Tomcat according to [2]. The syntax is also slightly different from Servlets. For example the following independent Java tags:

```
<%= Java expression or source code variable %>
<% Java statement %>
<% Java declaration or source code method %>
<%-- Java comment %>
```

Since JSPs are modified Servlets they too benefit from pre-defined variables, ready to be used:

request	cookies
response	response type
out	print writer
session	Single client sessions.
application	All clients' sessions.
pageContext	Access to all of a page's attributes.

The usage of JSP directives is mainly for inclusion of other JSP pages or Java libraries, for example <%@ import java.util.*; %> and <%@ page import Calculate %>.

JSP uses a special statement to perform SQL queries. This is the prepare statement which prepares variables to be part of an SQL statement, for example:

```
Class.forName("com.mysql.jdbc.Driver");
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost", "username", "password");

psmt = connection.prepateStatement("INSERT INTO students (firstname, lastname, studentId) VALUES (?,?,?);

psmt.setString(1, firstname);
psmt.setString(2, lastname);
psmt.setString(3, studentId);
```

Due to the evolving complexity of web sites, it is recommended to use redirects to reduce the number of pages in total as such **<%: forward page="Register.jsp" %>**.

The following table lists the benefits and drawbacks of JSP and Servlets.

Servlets	
Pros	**Cons**
Security – separation between the site's business logic from the web script.	Unpopular – modern web applications rarely use Servlets with Tomcat or GlassFish.
Performance – faster and more efficient than CGI.	Convoluted – testing manually or creating automated cases is a difficult task.
Stability – upon an error or exception, only the web page will be corrupted.	Maintainability – because HTML and Java source code are mixed into one entity, the application is difficult to support, scale, and be improved.
Java Server Pages	
Efficiency – the scripting nature of JSP and separation between programming logic and HTML design.	Project Size – JSP is very useful for small dynamic web sites but it becomes difficult to maintain if the web site if very complicated.
Compatibility and Availability – Tomcat and the Java platform runs of Windows, Linux, Mac OS X.	Features and Functionality – even though JSP uses Java beans for complex algorithms, Servlets offer a lot more to the developer who is building a complicated web application.
Convenient – no need to compile manually the source code after each iteration.	

The following figure illustrates the initial web site hierarchy.

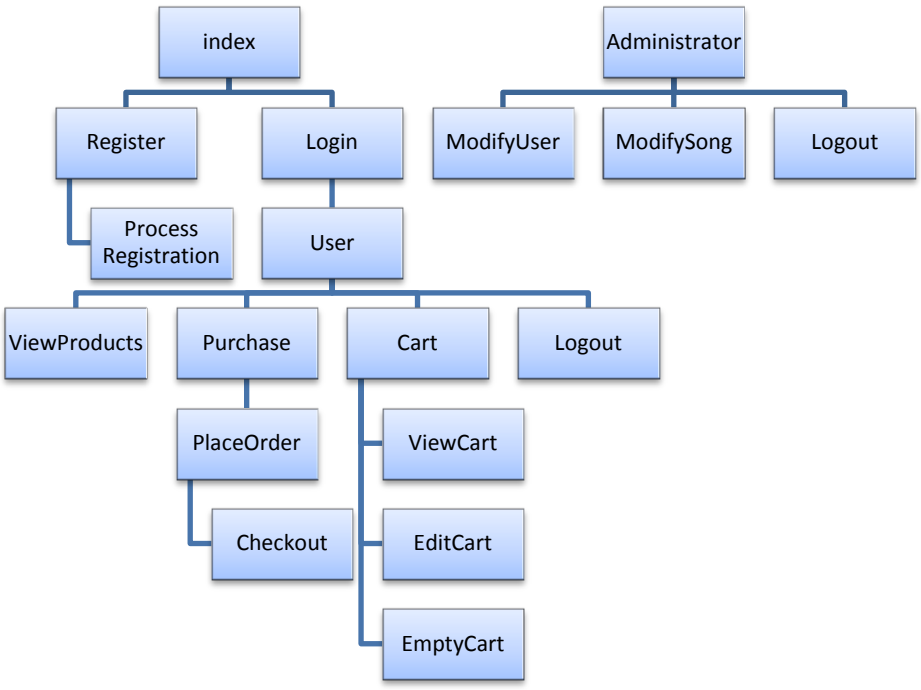


Figure 1: Web site hierarchy.

The following figure illustrates the initial web application flowchart.

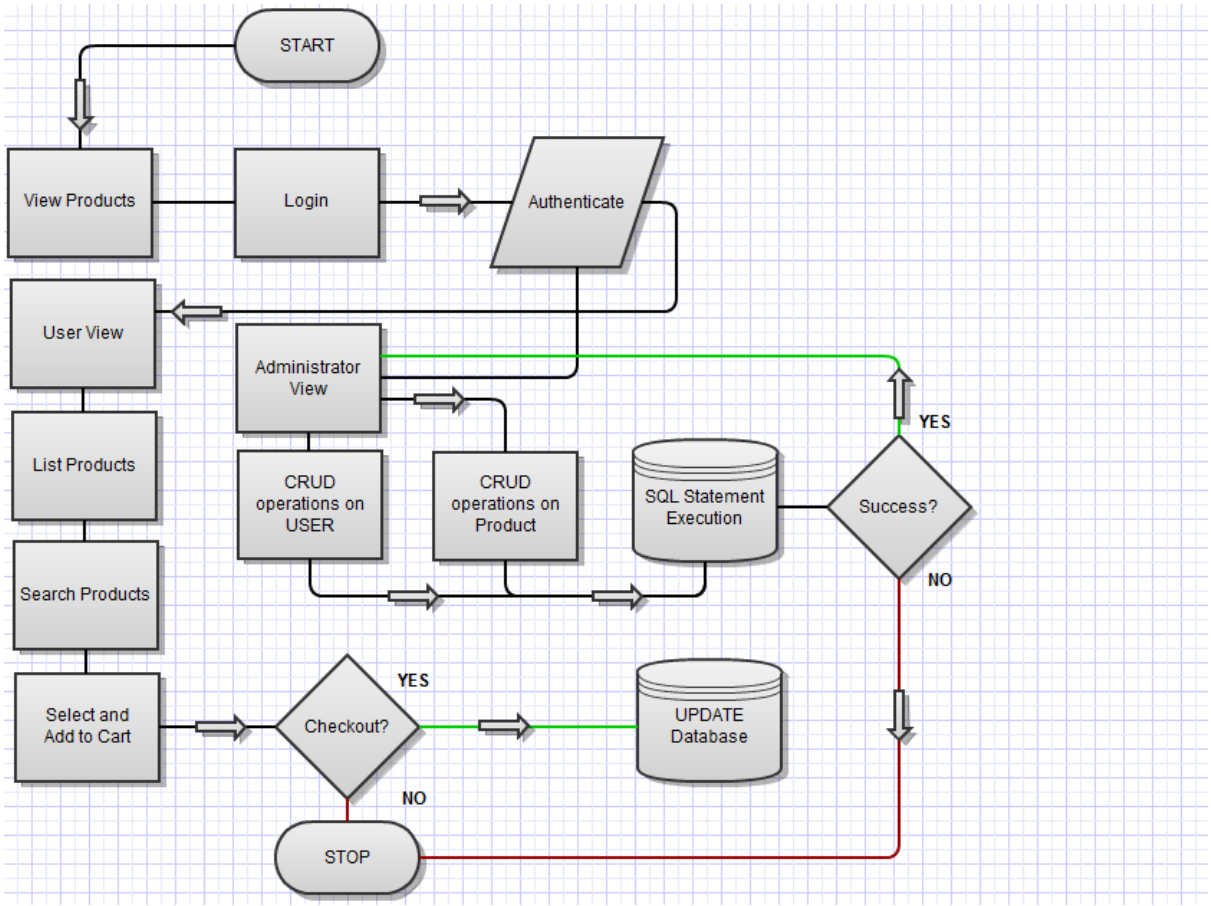


Figure 2: Web application flowchart.

2. Aims, Objectives, and Deliverables

The **aim** of this project is the creation of an online shopping web site similar to ecommerce web sites. As such there are several requirements which must be fulfilled. These are – the usage of scripting/programming language, relational database, user privileges, secure user authorisation, convenient and useful user interface. In addition the user should be able to perform common ecommerce tasks such as browse available products, search products by keyword, add selected products to the shopping cart, and finally checkout paying the right amount of money. The author's personal aim is to gain knowledge into electronic commerce web site structure and practise with popular web development tools such as JSP, JavaScript, PHP, MySQL, and HTML.

To achieve this aim the following **objectives** have to be satisfied in order to create a functional simple ecommerce web site:

- Build ecommerce logic with scripting / programming language – Java Servlet Pages.
- Store and manipulate data (Create, Read, Update, and Delete) in a relational database (i.e. MySQL) via the ODBC (Open Database Connectivity) driver from MySQL for JSP/Servlets/Java SQL and perform database administration using the DBMS (Database Management System) with PHPMyAdmin.
- Organise levels of user privileges which can access the web site into guest, user, and administrator. The guest would only be able to view the web site's front page. Registered

users would be able to browse, search, and select items to purchase. Administrators would be able to perform CRUD operations on all registered users and database records from a single unified interface.

- Appropriate user interface marked up in XHTML rather than HTML4, HTML5, or Flash. The usage of section headings, data tables containing database records, site href links, and standard HTML form fields would be essential elements of the human-computer interaction.

The resulting **deliverables** of this assignment are – project report discussing important aspects of the author's work, logbook detailing the author's progress in time, commented source code, and the original JSP work which includes source code, JSP/MySQL/Tomcat libraries, exported SQL dump file, image library, and readme file instructions.

3. Technical Approach

This section of the report discusses the web application's **algorithms**, **class diagrams**, and **application improvements**.

3.1 Algorithms

3.1.1 Connect to the MySQL database

The following block of code will only work if the Java connector driver is placed in the Tomcat server hierarchy. It is recommended that the password is automatically generated, for example by PHPMyAdmin itself for additional security.

```
String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
String usernameForDB = "adminaudiofarm";
String passwordForDB = "b9YMpCFtm9QPxCjv";
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
Statement statement = connection.createStatement();
```

3.1.2 Retrieve all song entries from the database

It is recommended to test each new SQL statement via the console or PHPMyAdmin. The latter one auto-generates simple SQL, based on user actions with the convention and format used in the official MySQL 5.5 manual. An SQL statement query returns a result set which is useless unless some form of loop is used. In this case the while loop is apt along with the next( ) method. It will return false if no more results are sent. Adding the results to String[ ] arrays is possible but potentially problematic. That is why **ArrayList<String>** type was used which offers flexible. This type also supports methods such as isEmpty( ), add( ), remove( ), indexOf( ).

```
String query = "SELECT* FROM`artist` LIMIT 0 , 30";
ResultSet resultSet = statement.executeQuery(query);
while(resultSet.next()){
        id.add(resultSet.getString(1));
        name.add(resultSet.getString(2));
        artist.add(resultSet.getString(3));
        album.add(resultSet.getString(4));
}
```

3.1.3 Displaying the result to the user

Displaying the resulting data as plain text/paragraph is not user friendly. Displaying the results as ordered or unordered list was inefficient. That is why the table model was used. In order to show all songs rather than just one, we need an iterator – 'for loop' restricted by the size of the id ArrayList. The get( ) method retrieves the data much like an ordinary array with an index.

```
<table border=1 color="black">
<th>ID</th><th>Name</th><th>Artist</th><th>Album</th>
<%
for (int i = 0; i<id.size(); i++){
		out.println("<tr>");
		out.println("<td>" + id.get(i) + "</td>");
		out.println("<td>" + name.get(i) + "</td>");
		out.println("<td>" + artist.get(i) + "</td>");
		out.println("<td>" + album.get(i) + "</td>");
		out.println("</tr>");} %></table>
```

Figure 3: User Home Page

3.1.4 Login → Process Login

The form submitted with username and password is accessible through the request.getParameter( ) method. As a precautionary measure they are trimmed from involuntary and unnecessary white spaces. Only if the username and password are not empty, the SQL statement to receive all user table data will be executed. If the SQL username and password match those submitted from the user, the user privileges will be assigned. Furthermore if the admin field is marked as one for the user, his/her privileges are evaluated to administrator. Even though a user can register to browse, select, and buy songs, only the system administrator of the web site and database can mark a user as admin who can manage other users and songs.

```
String formUsername = request.getParameter("username"); formUsername.trim();
String formPassword = request.getParameter("password"); formPassword.trim();

if(formUsername.isEmpty()){
		out.println("<p>Username field was empty.</p>");
} else if (formPassword != null && formPassword.isEmpty()){
		out.println("<p>Password field was empty.</p>");
} else {

if(queryUsername.matches(formUsername) && queryPassword.matches(Password)){
		userVerified = true;
		userId = Integer.parseInt(queryUserId);
		if(Integer.parseInt(queryAdmin) == 1){
			adminVerified = true;
			privileges = true;
		}
}
```

3.1.5 Logout → Terminate Session

In order to prevent unauthorised access to the web site after the user is no longer active, we need to destroy all user associated session variables. This is done by either the removeAttribute( ) method or clearAll( ) method. If we catch null pointer exception this means that either the session has timed out or someone else is trying to logout who is not an active user.

```
String username = (String) session.getAttribute("name");
try{
        session.removeAttribute("userId");
        session.removeAttribute("name");
        session.removeAttribute("privileges");
        message = username;
} catch(NullPointerException e){
        message = "Logout problem" + e.toString();
}
```

3.1.6 Register → Process Registration

The first code block gets the register form values. The second code block checks whether the values from the form are empty which would produce inaccurate database record. Also the password has to be at least 6 characters long for additional security. The third code block is an INSERT SQL statement into the user specific table fields. The same procedure is used to add songs into the database records.

```
String forname = request.getParameter("forname");
String surname = request.getParameter("surname");
String username = request.getParameter("username");
String password = request.getParameter("password");
String password2 = request.getParameter("password2");
```

```
if(forname.isEmpty()){
        out.println("<p>Forname field is empty!</p>");
} else if(surname.isEmpty()){
        out.println("<p>Surname field is empty!</p>");
} else if(password.isEmpty()){
        out.println("<p>Password field is empty!</p>");
} else if(password.length() < 6){
        out.println("<p>Your password should be more than 6 characters long!</p>");
} else if(password2.isEmpty()){
        out.println("<p>Retype Password field is empty!</p>");
        password2.trim();
} else if(!password2.matches(password)){
        out.println("<p>The two passwords do not match!</p>");
}
```

```
String query = "INSERT INTO user (id, forname, surname, username, password) VALUES (NULL,'"
                + forname + "', '"
                + surname + "', '"
                + username + "',  '"
                + password + "',  '"
```

3.1.7 Admin → Modify Artist Details

Depending on the received action type from the HTML form, the algorithm either updates artist details or removes an artist/song identified by its unique id, also sent through the HTML form. The same procedure is used to modify user details.

```
if(action.startsWith("edit")){
        String query = "UPDATE artist SET name='"
        + name + "',                                    artist='"
        + artist + "',                                  album='"
        + album + "',                                   "' WHERE id='" + id + "'";
        statement.executeUpdate(query);
} else if {
        String queryDelete = "DELETE FROM artist WHERE id='" + id + "'";
        statement.executeUpdate(queryDelete);
}
```

3.1.8 User

The procedure is similar to getting all song details as in the index.jsp. However we get the number of the songs resident in the basket first. If the session variable is not empty this means that the user has purchased some songs. If that is true we get the actual number of items/songs with the size( ) method. The basket image is either full or empty based on the 'basket' variable which is either zero or a positive integer number.

```
if(session.getAttribute("songsId") != null){
        cartItemsArrayList = (ArrayList<String>) session.getAttribute("songsId");
        cartItems = cartItemsArrayList.size();
} else {
        cartItems = 0;
}

if(cartItems == 0){
        basket = "img/basket-empty.png";
} else {
        basket = "img/basket-full.png";
}
```

Figure 4: Number of cart items and image.

3.1.9 Cart

This is an efficient loop to retrieve only already purchased songs from the database records. The loop iterations are limited by the number of song ids in the session. Then the SQL statement retrieves only those ids from the database corresponding to the ones in session. As usual the results are placed in and ArrayList. It is important to catch null pointer exception in order to notify the user that the basket is empty.

```
try{
        for(int j = 0; j<songsId.size(); j++){

                String query = "SELECT * FROM artist WHERE id ='" + songsId.get(j) + "'";
                ResultSet result = statement.executeQuery(query);

                while(result.next()){
                        id.add(result.getString(1));
                        name.add(result.getString(2));
                        artist.add(result.getString(3));
```

```
                                album.add(result.getString(4));
                        }
                }

} catch(NullPointerException exception){
                message = "The basket is empty.";
} catch(Exception exception){
                out.println("Could not retrieve data from the database.");
}
```

3.1.10 Budget

We need to notify the user of his/her budget. The SQL statement selects only the id and credit card according to the session id of the user. Afterwards only the integer value representing the budget is assigned.

```
String userBudget = "SELECT id, card FROM user WHERE id=" + session.getAttribute("userId") + "";
ResultSet resultBudget = statement.executeQuery(userBudget);

while(resultBudget.next()){
                card.add(resultBudget.getString(2));
}
```

3.1.11 Remove from Cart

The loop limited by the number of elements in the session array. All HTML form elements selected for removal are retrieved based on their sequential value of 1, 2, 3 etc. Afterwards a loop limited by the number of values received, checks whether they are contained in the session array. If that is so, the element is removed. Next the session array containing the purchased songs is erased. The same name array is loaded into the session environment with modified (removed) ids of the songs selected by the user.

```
songsId = (ArrayList<String>) session.getAttribute("songsId");
for(int i = 0; i<songsId.size(); i++){
                id.add(request.getParameter(String.valueOf(i)));
}
out.println(request.getParameter(String.valueOf(1)));

for(int j = 0; j<id.size(); j++){
                if(songsId.contains(id.get(j))){
                                index = songsId.indexOf(id.get(j));
                                songsId.remove(index);
                } else {
                                out.println("Nothing to remove");
                }
}
session.removeAttribute("songsId");
session.setAttribute("songsId", songsId);
```

3.1.12 Search

To search for records by the user input keyword we need to select all records in the artist tables, match a particular field (e.g. album) against the inclusive (e.g. +) keyword. Afterwards the results are recorded in song related arrays.

```
query = "SELECT * FROM artist WHERE MATCH (" + searchType + ") AGAINST ('+" + keyword + "' IN BOOLEAN MODE)";
result = statement.executeQuery(query);
                while(result.next()){
                                id.add(result.getString(1));
                                name.add(result.getString(2));
                                artist.add(result.getString(3));
                                album.add(result.getString(4));
                }
} catch (NullPointerException exception){     out.println("Nothing was found!"); }
```

3.1.13 Checkout

The first condition decreases the song quantity unless 0 which means – out of stock. The following SQL statement updates the artist table database records. The budget is also decreased depending on the user's credit card amount and the price of the song. The following SQL statement updates the user table at the id specified by the user id session variable.

```
if(songQuantity == 0){
        warning = "<h1>" + name + " by " + artist + " is temporary out of stock. </h1>";
} else {
        songQuantity -= 1;
}

budget = Double.parseDouble(card.get(0)) - Double.parseDouble(price);
String queryUpdateSongQuantity = "UPDATE artist SET quantity='" + songQuantity + "' WHERE id ='" + songsId.get(j) + "'";
statementSongGet.executeUpdate(queryUpdateSongQuantity);
String queryUpdateUserBudget = "UPDATE user SET card=" + budget + " WHERE id =" + session.getAttribute("userId") + "";
```

3.2 Class Diagrams

This section of the report contains several figures which represent the web application classes using UML diagrams. It is worth to mention that none of the variables are private. This is because the compiled code environment is the Tomcat server. Thus it would be difficult for a malicious software hacker to exploit the program. That is only possible if the web server's security was compromised.

AddArtist	
message: String	name: String
artist: String	album: String
year: String	duration: String
price: String	quantityInt: Integer
quantity: String	priceDouble: Double

AddSongsToCart	
id: ArrayList<String>	songsId: ArrayList<String>
name: ArrayList<String>	artist: ArrayList<String>
album: ArrayList<String>	year: ArrayList<String>
duration: ArrayList<String>	price: ArrayList<String>
message: String	query: String
resultSet: ResultSet	

Adduser	
message: String	forname: String
surname: String	gender: String
country: String	city: String
email: String	username: String
password: String	card: Integer
admin: Integer	query: String

Admin	
id: ArrayList<String>	name: ArrayList<String>
artist: ArrayList<String>	album: ArrayList<String>
year: ArrayList<String>	duration: ArrayList<String>
price: ArrayList<String>	quantity: ArrayList<String>
queryUserId: ArrayList<String>	forname: ArrayList<String>
surname: ArrayList<String>	gender: ArrayList<String>
country: ArrayList<String>	city: ArrayList<String>
email: ArrayList<String>	username: ArrayList<String>
password: ArrayList<String>	card: Integer
admin: Integer	query: String
result: ResultSet	

Cart	
songsId: ArrayList<String>	id: ArrayList<String>
name: ArrayList<String>	artist: ArrayList<String>
album: ArrayList<String>	year: ArrayList<String>
duration: ArrayList<String>	price: ArrayList<String>
quantity: ArrayList<String>	card: ArrayList<String>
message: String	budget: Integer
query: String	result: ResultSet

Checkout	
card: ArrayList<String>	songsId: ArrayList<String>
message: String	name: String
artist: String	price: String
warning: String	songQuantity: Integer
budget: Double	getUserBudget: String
budgetResult: ResultSet	getSongQuantity: String
quantityResult: ResultSet	Update: String

index	
id: ArrayList<String>	name: ArrayList<String>
artist: ArrayList<String>	album: ArrayList<String>
year: ArrayList<String>	duration: ArrayList<String>
price: ArrayList<String>	quantity: ArrayList<String>
query: String	result: ResultSet

Logout	
warning: String	username: String
lastVisit: Date	query: String

ModifyArtistDetails	
message: String	id: String
name: String	artist: String
album: String	year: String
duration: String	price: String
quantity: String	priceDouble:Double
quantityInt: Integer	action: String
queryUpdate: String	queryDelete: String

ModifyUserDetails	
message: String	queryUserId: String
forname: String	surname: String
gender: String	country: String
city: String	email: String
username: String	password: String
action: String	card: String
cardInt: Integer	queryUpdate: String
queryDelete: String	

ProcessLogin	
queryUserId: String	userId: Integer
queryForname: String	querySurname: String
queryUsername: String	queryPassword: String
queryAdmin: String	userVerified: Boolean
privileges: Boolean	verifiedForname: String
message: String	link: String
formUserName: String	formPassword: String

RemoveFromCart	
index: Integer	songsId: ArrayList<String>
id: ArrayList<String>	

ProcessRegistration	
forname: String	surname: String
gender: String	country: String
city: String	email: String
username: String	password: String
password2: String	card: String
cardInt: Integer	queryUpdate: String

Search	
id: ArrayList<String>	name: ArrayList<String>
artist: ArrayList<String>	album: ArrayList<String>
year: ArrayList<String>	duration: ArrayList<String>
price: ArrayList<String>	quantity: ArrayList<String>
keyword: String	searchType: String
query: String	result: ResultSet

3.3 Web Application Improvements

This section of the report lists and discusses web application improvements. In addition the reason behind the improvement is also provided. The following is the list of improvements with brief explanation of the problem or the upgrade itself:

- The administrator should be able to upload pictures to the site. This would be the song's album art cover.
- The user should be able to browse the website and see album art cover together with song details.
- Security feature which would prevent arbitrary users from accessing web pages that should be only used by the administrator/system.
- Security feature which would encrypt the password supplied by the user during the registration process. The password is stored in plain text in the database together with the user details. However if someone (malicious user) was to steal the user's details from the database, they would not be able to log in as neither user nor administrator. This is because the password is based on the username two characters long salt and needs to be decrypted.
- During the login and register procedures there was a software bug due to username duplicates – the query would return the last id of the column that matches against the condition.
- The user should be able to remove recently added items to the cart. Such functionality is similar to popular ecommerce web sites, such as Amazon, iTunes, eBay etc. This gives the user the opportunity to remove unwanted items rather than logout and start out the procedure again. This would increase the users' satisfaction.
- Alter the way the web application performs database SQL queries. Currently before each queryExecute and ResultSet the script has to establish connection to the database – load MySQL connector, create new statement etc. It is recommended to use a Java bean or include JSP with DB settings/connection.
- The user should be able to view his/her history as in purchased songs in the past along with a calculated sum of the purchase. This idea was influenced by popular web sites such as Amazon and eBay which keep track of user activity.
- Based on the previous suggested improvement the user would also be able to view song suggestions. For example if the songs in the past were jazz, rock, pop, the web site interface would suggest other songs of the same style. This would yield more profit for the organisation using this web application.
- As a preventive security measure the web application should be tested for SQL injection. This unfortunately is a problem with many ecommerce web sites where a hacker could view all MySQL table records exploiting SQL statements coupled with HTML forms.

3.3.1 Upload picture to the server

Rather than loading the binary file be it JPEG, GIF, or PNG into the database table using the MySQL format BLOB, it would be better to use VARCHAR which would simply store the link to the image located on the local or remote server. There are two major problems with BLOB. The first one is that it makes the database slower during processing (e.g. SELECT and MATCH AGAINST). This is because

the database is optimised for strings, characters, and integers rather than binary files. The second is that the developer has little control over the image in terms of the HTML <img> tag. For example the following code retrieves an image and then writes out the result:

```
String query = "SELECT id, album_art FROM artist WHERE id=1";
ResultSet result = executeQuery(query);
Response.setContentType("OutputStream outStream = response.getOutputStream();
outStream.write(result.getBytes("album_art");
```

For the above example the SQL statement should be constructed as follows:

```
ALTER TABLE artist ADD album_art BLOB NOT NULL
```

Instead the following SQL statement was used for storage of links.

```
ALTER TABLE artist ADD album_art VARCHAR(25) CHARACTER SET ASCII COLLATE ascii_general_ci NOT NUL
```

In addition the actual HTML form is also different. W3C [3] recommends the usage of either "application/x-www-form-urlencoded" or "multipart/form-data". Whereas the first is not suitable for large files such as media, the second one is more popular amongst web developers because it can be used to load large amounts of non-ASCII text binary data. For example the following HTML form can be used for uploading album art covers in JPEG, PNG, and GIF.

```
<form action="UploadAlbumArt.jsp" method="POST" enctype="multipart/form-data">
        < input type="file" name="file" />
        < input type="submit" name="submit" value="Upload" />
</form>
```

Afterwards the form can be processed – check multipart contents, parse file, iterate over the file to get field name, file name, content type, size in byes, and finally write the bytes to the selected destination on the disk drive of the server. To achieve this, the following libraries had to be loaded onto the Tomcat server in addition to "mysql-connector-java-5.1.18-bin" as suggested by [4]:

- commons-fileupload-1.2.2
- commons-io-2.3
- commons-io-2.3-javadoc
- commons-io-2.3-sources
- commons-io-2.3-tests
- commons-io-2.3-test-sources

Furthermore the actual processing of the image file had to be in conjunction with the ServletContext Interface library as suggested by [5]:

```
item = (FileItem) iterator.next();
fieldName = item.getFieldName();
fileName = item.getName();
contentType = item.getContentType();
fileSize = item.getSize();
file = new File(config.getServletContext().getRealPath("/") + "img/"+ fileName);
item.write(file);
```

3.3.2 User viewing album art while browsing and searching.

The address for the image source is stored in the album_art column, which should be retrieved with other song details, as such:

```
ArrayList<String> album_art = new ArrayList<String>; while (result.next()){album_art.add(result.getString(9));}
out.println("<img src=\" " + album_art.get(i) + "\" "/>
```

3.3.3 Prevent unauthorised access to sensible web resources such as Admin.jsp

To prevent this from happening we can check whether the session has been created by the user/administrator signing in with correct username and password that match the database records. Placing the following code block before the main execution of each JSP code will redirect or rather forward the web browser to a web page notifying the user of unauthorised access:

```
<jsp: forward page="Warning.jsp" />
```

In order to make sure that the browser is forwarded only when the session was not established:

```
If(session.getAttribute("privileges") ==null){

}
```

The privileges attribute is set by ProcessLogin.jsp during successful authentication along with the user id and his/her name. However there is a problem because the **<jsp: forward page=" … " />** cannot be embedded within the Java source code. That is why the following was used – **pageContext.forward("Warning.jsp")**. The following table lists the web pages into which the described script was NOT implemented.

Publically Accessible Web Pages	
ProcessLogin.jsp	Login.jsp
index.jsp	Logout.jsp
Register.jsp	ProcessRegistration.jsp
Warning.jsp	

3.3.4 Encrypt the user supplied password

To prevent account mismanagement by administrators or other privileged users with access to the web application's database and specifically user details, all password are encrypted. This is used by ecommerce web sites to also encrypt other sensible information such as credit card details.

MySQL [6] supports several encryption algorithms such as Advanced Encryption Standard, Data Encryption Standard, and hashing algorithms such as SHA1, SHA2, and MD5. Hash functions provide similar benefits for creating a secret message that would be infeasible to crack in an ordinary amount of time. In addition a hash can be computed faster and supports automatic block padding (unlike AES/DES). In order to obfuscate the user supplied password the SHA2 algorithm was used which according to MySQL [6] is more secure than MD5 and SHA1. Furthermore MySQL's AES/DES require changes to the password column to binary BLOB/String whereas SHA2 uses plain text. According to MySQL [6] there are five options/arguments for the functions that measure the strength in bits of encryption – 224, 256, 384, 512, and default value.

After some testing and experienced difficulties implementing SHA2 it became obvious that the field had to be changed in order to accommodate the big size of 512 bits key encryption, as follows:

```
ALTER TABLE user CHANGE password TEXT CHARACTER SET ascii NOT NULL
```

To generate the SHA2 password the following was used:

```
SELECT SHA2(' " + formPassword + " ', 512)
```

3.3.5 Check for username duplicates during the login procedure

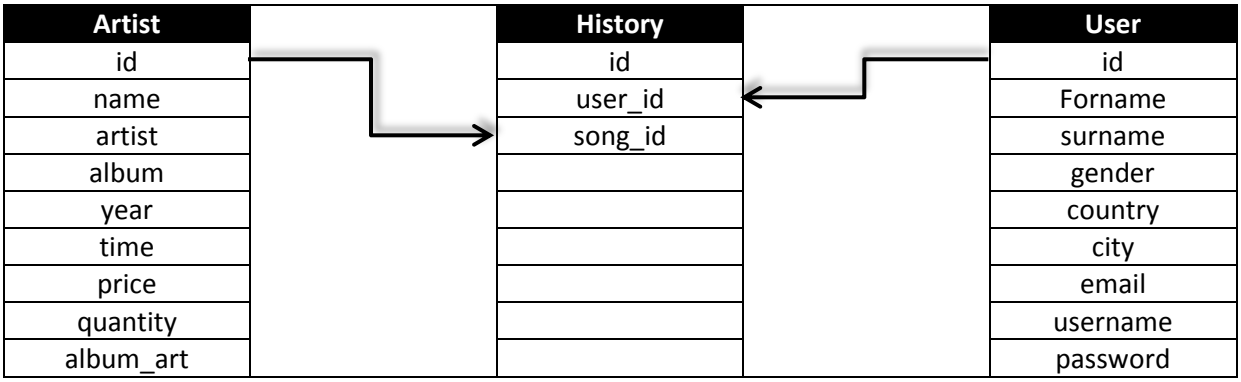
Production ecommerce web sites would not allow a user to login with username and password that are the same as someone else's. The following block of code was used to check for duplicates:

```
String queryForDuplicate = “SELECT username FROM user “;
ResultSet resultFromDuplicate = statement.executeQuery(queryForDuplicate);
while(resultForDuplicate.next()){
        if(resultFromDuplicate.getString(1).startsWith(username)){
                duplicate = true;
                message = “Username already exists!”;
         }
}
If(!duplicate){

}
```

3.3.6 Purchase History

An additional web page was added to the interface where registered users can view past purchases (songs details). This is a common feature amongst ecommerce web sites which could be used as sort of electronic receipt. It is recommended to use a third MySQL table which would store only the user id and purchased songs ids. The last two are also known as foreign id keys to the “history” table, such that:



All three table ids are unique, primary keys and auto increment with each new entry. However the history table user_id and song_id are updated only upon checkout with the following code block:

```
String queryHistory = “INSERT INTO history (id, user_id, song_id) VALUES (NULL, “
+ session.getAttribute(“userId”)
+ “, “
+ songsId.get(i) + “)”;
statement.executeUpdate(queryHistory);
```

Since there is a reference to two different database tables the MySQL JOIN syntax should be used. There are several join expressions such as LEFT, RIGHT, INNER, CROSS etc. This project uses the LEFT join as follows:

```
“SELECT * FROM history
LEFT JOIN artist ON history.song_id = artist.id
LEFT JOIN user ON hisoty.user_id = user.id
WHERE history.user_id = “ + session.getAttribute(“userId”) + LIMIT 0, 30;
```

3.3.7 Songs recommendation

This feature was placed in the History.jsp web page to provide suitable recommendations based on the purchased songs genre (e.g. jazz, rock, pop). First the type of music is acquired from previously purchased songs. Afterwards a query is performed to match all songs from the artist table corresponding to the genre type.

```
SELECT * FROM artist WHERE genre=’ “ + genre.get(0) + “ ‘ “;
```

There was no need to use MySQL LIKE syntax or perform full-text search using MATCH () and AGAINST () SQL syntax. This is because music genres are stored as tiny string values such as pop, rock, and rap. Using the WHERE syntax would not put a heavy strain on the database even if it grows in the future.

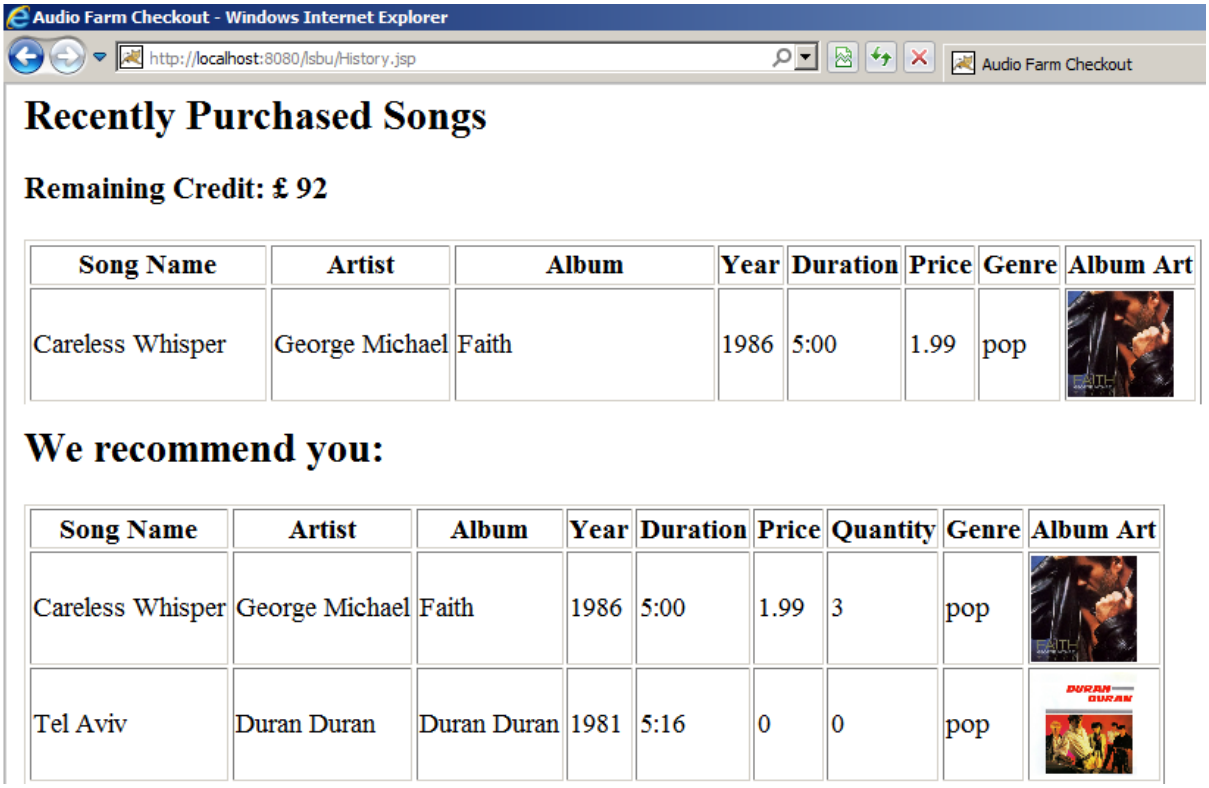


Figure 5: Past song purchases and recommendations.

3.3.8 Database connection Java bean

In order to shorten the source code of the JSP and clear repetitive code the following Java bean can be included into each JSP that requires it, located in Tomcat's lib directory.

ConnectToDB	
driver: String	address: String
username: String	password: String
connection: Connection	result: ResultSet
statement: Statement	success: Boolean
connect( )	disconnect()
insert( )	update()
select()	delete()

3.3.9 SQL Injection

For the past couple of years these SQL attacks have become extremely popular amongst web application exploiting security hackers according to [7]. All they require is a web browser and web pages which contains HTML <form>, which takes arguments from the user (i.e. hacker) and sends them to the Tomcat server with POST or GET methods which trigger an SQL statement. For example:

```
SELECT * FROM user WHERE name=' hack or 1 = 1 --'
```

The ‘hack or 1=1 --‘ is substituted as valid username login query. However regardless of whether ‘hack’ exists or not the query will return all database entries (i.e. 1=1) and omit the rest of the SELECT statement because of (--). Another example is the following where a malicious user tries to delete all entries in the database table:

```
SELECT * FROM user WHERE username=’hack; DELETE * FROM user;
```

The original SQL statement will be interrupted and the malicious SQL query will be executed instead.

To protect against such scenarios the developer should restrict access to the database as such to prevent deletion. For example the following SQL query creates a less privileged database user who can only select, insert, and update data without being able to delete data.

```
CREATE USER ‘user’@’IDENTIFIED BY ‘***’;
GRANT SELECT, INSERT, UPDATE on audiofarm;
```

To prevent hackers from inserting special characters into the HTML forms PHP uses two functions which strip them away – pp_escape_string and mysql_real_escape_string. JSP and Servlets on the other hand make use of ‘prepared’ SQL statement to escape special characters.

4. Results and Discussions

This section of the report discusses the encountered difficulties during the development of the web application and the implementation of vital features and routines.

Inserting variables in SQL statements was problematic. This was so because SQL queries are static when sent to the MySQL server. To fix the problem the query string was constructed by concatenation of static commands (e.g. SELECT, FROM) and actual variables. For example:

```
String query = “SELECT * FROM” + artist + “WHERE id= “ + userId + “ AND admin=’ ” + admin + “ ‘ “;
```

Using char representation for gender (e.g. m, f) in MySQL was problematic. After the statement is executed the following errors were produced:

```
javax.servlet.ServletException: com.mysql.jdbc.Mysql
DataTruncation: DataTruncation: Data too long for column ‘gender’ at row 1
```

To fix this issue the gender field was changed from CHAR type to VARCHAR type of 1 byte length which is sufficient for ‘m’ and ‘f’.

Using MySQL BOOL as a way to store Boolean values from the JSP was problematic. Due to either the driver API associated with the executeQuery( ) method or difference in MySQL and Java Boolean formats, there was a persistent problem. The solution was to use INT type in the MySQL table to represent zero as FALSE and one as TRUE.

There was a software bug discovered during the registration and login procedures. That is if there are two identical usernames in the database table. Even though their id numbers are unique and different, the username remains the same. Either due to the loop in the JSP or loop in MySQL, the problem remained. The solution was to check the database for duplicates before submitting/retrieving any user information, practice commonly used by professional content management systems.

Repetitive use of connection algorithm to the database cluttered the source code. That is because before each data transaction to the MySQL server (e.g. executeQuery, executeUpdate) the script has to connect to the database using address, username, and password. Also additional tasks are required to submit a simple SQL statement. It would be possible to create a connection Java bean which is called by the script when an SQL query is made. This would simplify the source code and make it more modular.

Using Java arrays of String type posed many difficulties during loops and even when retrieving information for HTML submit forms or tables of data presented to the client. The solution was the usage of ArrayList<String> type which automatically expands in size, have useful data manipulation functions (e.g. add, remove, clear, clone), and can be easily iterated with a 'for each' loop. For example this is apt for building an HTML table containing SQL query item details:

```
<td>
    <input type="text" value="<%= artist %>" name="artist" maxlength="25" />
</td>
<%
 for (String item: artist){
    out.println("<td> type=\"text\" value=\"" + item + "\" name=\"artist\" maxlength=\"25\" /></td>");
}
%>
<% for (int i = 0; i< id.size( );  i++){ … }
```

Providing the user with HTML form nested in a table with song details was problematic. For example the form would pick-up only the first entry in the <select><option>. To fix this issue the form was embedded in all table rows. For example:

```
out.println("<tr><form method=\"POST\" action=\"ModifyArtistDetails.jsp\">");
out.println("<td><input readonly=\"readonly\" type=\"text\" value=\"" + id.get(i) + "\" name=\"id\" size=\"2\"/></td>");
out.println("<td><input type=\"text\" value=\"" + name.get(i) + "\" name=\"name\" maxlength=\"25\"/></td>");
out.println("<td><input type=\"text\" value=\"" + artist.get(i) + "\" name=\"artist\" maxlength=\"25\"/></td>");
out.println("<td><input type=\"text\" value=\"" + album.get(i) + "\" name=\"album\" maxlength=\"25\"/></td>");
out.println("<td><select name=\"action\"><option value=\"edit\">Edit</option>
<option value=\"remove\">Remove</option></select>
<br/><input type=\"submit\" value=\"Perform\" name=\"submit\"/></td></tr></tr></form>");
```

Selecting and getting multiple values (of songs for example) using the HTML checkbox with the request.getParameter( ) method was problematic. This was because the JSP has to know exactly what values to expect and checkbox names cannot be duplicated. For example the script would receive only a single, usually the last clicked/selected checkbox. To overcome this issue the names and values were changed to correspond directly to database ids. Thus the processing form script would first connect to the database and then process the HTML form. Alternative solution was to make the input names sequential as in a 'for loop'. Thus the processing form script can use a simple 'for loop' to iterate over the non-null form values.

Deleting a song or user, by song name or user surname was not efficient. Even though it is unlikely for songs to be duplicated and somewhat more likely for surnames, this did not occur during the testing of the application. However as a precautionary measure it would be wise to delete a song/user identified by the unique id.

Processing the session array 'songsId' which contains the ids of the purchased songs was problematic with null values. For example a simple check whether the array is empty, crashed the script. Thus the following block of code was necessary to avoid unexpected behaviour:

```
try {
    if (songsId.isEmpty( )){
        out.println("You have not purchased anything.");
    }
} catch ( NullPointerException exception){
    out.println("Null Values");
}
```

Searching database entries based upon given user criteria. To perform full-text searches in MySQL tables we need the MyISAM engine. The default engine for MySQL Server 5.5 is InnoDB which supports transactions. This means that the artist table containing song details had be changed with the following SQL statement:

```
ALTER TABLE artist ENGINE = MYISAM
```

Also the fields that were going to be searched had to be marked with FULL-TEXT attributes with the following SQL statement:

```
ALTER TABLE artist ADDFULLTEXT (name, album, artist)
```

To test whether the operation was successful the following query was used to look for "George" in the artist column which correctly returned "George Michael".

```
SELECT * FROM artist WHERE MATCH (artist) AGAINST ('George; IN NATURAL LANGUAGE MODE) LIMIT 0, 30
```

During the testing of the application there was a software bug associated with searching, particularly with the 'IN NATURAL LANGUAGE MODE'. That is because if there are two identical entries, the query will return zero results rather than correctly – two or at least one. For example there were two entries for the Duran Duran album Notorious and yet the search query returned nil. To overcome this problem the following SQL mode was implemented as suggested by the MySQL online documentation [8]:

```
MATCH (artist) AGAINST (' + Duran Duran' IN BOOLEAN MODE)
```

The + operator means inclusive and the – operator may be used to exclude a keyword. Unfortunately MySQL FULL-TEXT search does not support wildcards such as % which can find a word that starts or ends with the specified keyword string. However MySQL FULL-TEXT search benefits from optimal performance when searching text according to [8].

5. Conclusion

The aim of the project – to create a simple ecommerce CMS web application which allows users to browse, search, and purchase products was achieved within the given time limit. All of the outlined objectives in previous sections of this report were satisfied as required. In addition several useful and interesting improvements were implemented in the web application such as – user past purchases, song recommendations, username duplicate checks, view album art covers, upload of album art covers, modification of user details, modification of song details, password encryption etc.

Unfortunately not all planned features were implemented in terms of ecommerce functionality. This was partly due to time limitations and partly because there are very few forums, blogs, and books on the topic of Java Server Pages and Content Management Systems. Most of the available resources located in the Perry library were either seriously outdated or unrelated to the assignment as a whole. The few available books on the subject were more confusing than helpful for the development of the project. The most valuable help was Dr. Perry Xiao's instructions and tips in addition to blackboard electronic resources such as tutorials and example JSP source code files.

Despite the author's initial dislike of JSP/Servlets compared to other popular technologies for server side scripting such as PHP, Python, Ruby, at the end of the project development the author's attitude changed to appreciate this Java technology due to the following – simplicity, advance functionality when necessary with Java beans, fast Tomcat performance, readily available jar libraries, no need for invoking the Java Compiler, splendid debugger, and straight-forward integration with HTML without the need for CGI. Nonetheless the lack of relevant online resources and books, Java SE style of documentation, bewildering configuration for WEB-INF, libraries, XML are still very unfortunate aspects of JSP/Servlets programming.

List of Reference

[1] G. Zambon. Beginning JSP, JSF, and Tomcat Web Development: From Novice to Professional. New York: Apress, 2007.

[2] B. Perry. Java Servlet & JSP Cookbook. Sebastopol: O'Reilly, 2004.

[3] W3C. W3C Recommendations - Form Content Type, [no-date]. [Online] Available from: http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2 [Accessed 29 April 2012]

[4]Apache Commons. Commons File Upload, [no-date]. [Online] Available from: http://commons.apache.org/fileupload/using.html [Accessed 01 May 2012]

[5] Oracle Java Documentation. Interface ServletContext, [no-date]. [Online] Available from: http://docs.oracle.com/cd/E17802_01/products/products/servlet/2.3/javadoc/javax/servlet/ServletContext.html [Accessed 02 May 2012]

[6] MySQL Reference Manual. Encryption and Compression Functions, [no-date]. [Online] Available from: http://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_sha2 [Accessed 03 Mat 2012]

[7] BBC News Technology. Nokia's developer network hacked. 29 August 2011. [Online] Available from: http://www.bbc.co.uk/news/technology-14706810 [Accessed 05 May 2012]

[8] MySQL Reference Manual. Full-Text Search Functions, [no-date]. [Online] Available from: http://dev.mysql.com/doc/refman/5.6/en/fulltext-search.html [Accessed 06 May 2012]

Appendix A (JSP code)

AddArtist.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<!-- Prevent unauthorise access to the database -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm Add New User</title>
</head>
<body>
<a id="top" href="Admin.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Get form contents and add artist to the database -->
<%
String message = "";
String name = request.getParameter("name");
name.trim();
String artist = request.getParameter("artist");
artist.trim();
String album = request.getParameter("album");
album.trim();
String year = request.getParameter("year");
year.trim();
String duration = request.getParameter("duration");
duration.trim();
String price = request.getParameter("price");
price.trim();
String quantity = request.getParameter("quantity");
quantity.trim();
String genre = request.getParameter("genre");
genre.trim();
String album_art = "default.jpg";
Double doublePrice = Double.parseDouble(price);
int intQuantity = Integer.parseInt(quantity);

/* Perform simple form validation. Check whether a form is empty. */
if(name.isEmpty()){
		out.println("<p>Name field is empty!</p>");
} else if(artist.isEmpty()){
		out.println("<p>Artist field is empty!</p>");
} else if (album.isEmpty()){
		out.println("<p>Album field is empty!</p>");
} else if (year.isEmpty()){
		out.println("<p>Year field is empty!</p>");
} else if (duration.isEmpty()){
		out.println("<p>Duration field is empty!</p>");
} else if(price.isEmpty()){
		out.println("<p>Price field is empty!</p>");
} else if(quantity.isEmpty()){
		out.println("<p>Quantity field is empty!</p>");
} else {

		/* Establish connection to the database with server address, username and password */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String usernameForDB = "adminaudiofarm";
		String passwordForDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
		Statement statement = connection.createStatement();
```

```
		/* Insert the data supplied by the HTML form to the database. */
		String query = "INSERT INTO artist (id, name, artist, album, year, time, price, quantity, album_art, genre) VALUES (NULL,'"
				+ name + "', '"
				+ artist + "', '"
				+ album + "', '"
				+ year + "', '"
				+ duration + "', '"
				+ doublePrice + "', '"
				+ intQuantity + "', '"
				+ album_art + "', '"
				+ genre + "')";

		statement.executeUpdate(query);

		message = "<h2>Artist details were successfully stored in the database.</h2>";
}

%>
<%= message %>
</body>
</html>
```

AddSongsToCart.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>

<!-- Prevent unauthorise access to the web application -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm add Songs to Cart</title>
</head>
<body>
<a href="User.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Add songs to cart -->

<%
/* Initialise arrays to store songs' details. */
ArrayList<String> id = new ArrayList<String>();
ArrayList<String> songsId = new ArrayList<String>();
String message = "";

ArrayList<String> name = new ArrayList<String>();
ArrayList<String> artist = new ArrayList<String>();
ArrayList<String> album = new ArrayList<String>();
ArrayList<String> year = new ArrayList<String>();
ArrayList<String> duration = new ArrayList<String>();
ArrayList<String> price = new ArrayList<String>();
ArrayList<String> quantity = new ArrayList<String>();
ArrayList<String> genre = new ArrayList<String>();
ArrayList<String> album_art = new ArrayList<String>();

try{

		/* Establish connection to the databse using the server address, username, and password */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String username = "adminaudiofarm";
		String password = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB,username,password);
```

```
		Statement statement = connection.createStatement();

		/* Retriece all data from the artist's table. */
		String query = "SELECT * FROM artist LIMIT 0 , 30";
		ResultSet resultSet = statement.executeQuery(query);

		/* Get the first id from the query. */
		while(resultSet.next()){
			id.add(resultSet.getString(1));
		}

}catch(Exception exception){
		out.println("Could not retrieve data from the database.");
}

/* Get selected songs from the form. Add all non null to an array list. Set session variable using the array list. */
for(int i = 0; i<id.size(); i++){
		if(request.getParameter(id.get(i)) != null ){
			songsId.add(request.getParameter(id.get(i)));
			session.setAttribute("songsId", songsId);
		}
}

/* Iterate limited by the size of the songs array. This is necessary for the SQL statement. */
for(int j = 0; j<songsId.size(); j++){
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String username = "adminaudiofarm";
		String password = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB,username,password);

		Statement statementSong = connection.createStatement();

		/* Retrieve all data from the artist table where there is match against the user id. */
		String querySong = "SELECT * FROM artist WHERE id = " + songsId.get(j) + " ";
		ResultSet resultSetSong = statementSong.executeQuery(querySong);

		/* Iterate until all data is stored into the song related arrays. This is used for the HTML display. */
		while(resultSetSong.next()){
			name.add(resultSetSong.getString(2));
			artist.add(resultSetSong.getString(3));
			album.add(resultSetSong.getString(4));
			year.add(resultSetSong.getString(5));
			duration.add(resultSetSong.getString(6));
			price.add(resultSetSong.getString(7));
			quantity.add(resultSetSong.getString(8));
			album_art.add(resultSetSong.getString(9));
			genre.add(resultSetSong.getString(10));
		}
}

/*
ArrayList<String> result = new ArrayList<String>();
result = (ArrayList<String>) session.getAttribute("songsId");
for(int j = 0; j<result.size(); j++){
		out.println(result.get(j));
}
*/

%>

<h1><%= message %></h1>

<h1>User <%= session.getAttribute("name") %> added to cart!</h1>
<form method="POST" action="Checkout.jsp">
<table border=1 color="black">
<th>Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Time</th><th>Price</th><th>Quantity</th><th>Genre</th><th>Albu
m Art</th>
<%
/* Iterate until all song related array data has been displayed to the user. */
for (int i = 0; i<name.size(); i++){
```

```
                                        out.println("<tr>");
                                        out.println("<td>" + name.get(i) + "</td>");
                                        out.println("<td>" + artist.get(i) + "</td>");
                                        out.println("<td>" + album.get(i) + "</td>");
                                        out.println("<td>" + year.get(i) + "</td>");
                                        out.println("<td>" + duration.get(i) + "</td>");
                                        out.println("<td>" + price.get(i) + "</td>");
                                        out.println("<td>" + quantity.get(i) + "</td>");
                                        out.println("<td>" + genre.get(i) + "</td>");
                                        out.println("<td><img src=\"" + album_art.get(i) + "\" width=\"60\" heigth=\"60\" /></td>");
                                        out.println("</tr>");
}
%>
</table>
<br/><input type="submit" value="Checkout" />

</body>
</html>
```

AddUser.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>

<!-- Prevent unauthorise access to the web application -->
<%
if(session.getAttribute("privileges") == null){
        pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
        <title>Audio Farm Add New User</title>
</head>
<body>
<a id="top" href="Admin.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Get form contents and add user to the database -->
<%
/* Initialise user related Strings with data sent from the form. */
String message = "";
String forname = request.getParameter("forname");
forname.trim();
String surname = request.getParameter("surname");
surname.trim();
String gender = request.getParameter("gender");
surname.trim();
String country = request.getParameter("country");
surname.trim();
String city = request.getParameter("city");
surname.trim();
String email = request.getParameter("email");
surname.trim();
String username = request.getParameter("username");
username.trim();
String password = request.getParameter("password");
password.trim();
String secretQuestion = request.getParameter("secretQuestion");
password.trim();
String secretAnswer = request.getParameter("secretAnswer");
secretAnswer.trim();
String dateofbirth = request.getParameter("dateofbirth");
secretAnswer.trim();

/* Get the amount of resource on the user's credit card for display. If the card is empty notify the user. */
String card = request.getParameter("card");
Integer.parseInt(card);
int intCard = 0;
```

```
if(!card.isEmpty()){
		intCard = Integer.parseInt(card);
}

/* Get the admin parameter. Only administrators can perform advanced operations. They are marked as 1. */
String admin = request.getParameter("admin");
int intAdmin = 0;
if(!admin.isEmpty()){
		intAdmin = Integer.parseInt(admin);
}

/* Perform simple form validation. Check whether the sent HTML forms are empty. */
if(forname.isEmpty()){
		out.println("<p>Forname field is empty!</p>");
} else if(surname.isEmpty()){
		out.println("<p>Surname field is empty!</p>");
} else if (gender.isEmpty()){
		out.println("<p>Gender field is empty!</p>");
} else if (country.isEmpty()){
		out.println("<p>Country field is empty!</p>");
} else if (city.isEmpty()){
		out.println("<p>City field is empty!</p>");
} else if(email.isEmpty()){
		out.println("<p>Email field is empty!</p>");
} else if(username.isEmpty()){
		out.println("<p>Username field is empty!</p>");
} else if(password.isEmpty()){
		out.println("<p>Password field is empty!</p>");
} else if(password.length() < 6){
		out.println("<p>The password should be more than 6 characters long!</p>");
} else if(secretAnswer.isEmpty()){
		out.println("<p>SecretAnswer field is empty!</p>");
} else if(dateofbirth.isEmpty()){
		out.println("<p>Date of Birth field is empty!</p>");
} else {

		/* Establish connection to MySQL with server address, username and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String usernameForDB = "adminaudiofarm";
		String passwordForDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
		Statement statement = connection.createStatement();

		/* Insert the data from the HTML form into the user table. */
		String query = "INSERT INTO user (id, forname, surname, gender, country, city, email, username, password, secretquestion,
secretanswer, dateofbirth, card, admin) VALUES (NULL,'"
				+ forname + "', '"
				+ surname + "', '"
				+ gender + "', '"
				+ country + "', '"
				+ city + "', '"
				+ email + "', '"
				+ username + "', '"
				+ password + "', '"
				+ secretQuestion + "', '"
				+ secretAnswer + "', '"
				+ dateofbirth + "', '"
				+ intCard + "', '"
				+ intAdmin +"')";

		statement.executeUpdate(query);

		message = "<h2>User details were successfully stored in the database.</h2>";
}

%>
<%= message %>
</body>
</html>
```

Admin.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<!-- Prevent unauthorised access to the web application -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm Admin</title>
</head>
<body>
<a id="top" href="Admin.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Show songs and enable admin to add, edit, and remove -->
<%
String welcomeName = "";
/* Variables for artist table */
ArrayList<String> id = new ArrayList<String>();
ArrayList<String> name = new ArrayList<String>();
ArrayList<String> artist = new ArrayList<String>();
ArrayList<String> album = new ArrayList<String>();
ArrayList<String> year = new ArrayList<String>();
ArrayList<String> duration = new ArrayList<String>();
ArrayList<String> price = new ArrayList<String>();
ArrayList<String> quantity = new ArrayList<String>();
ArrayList<String> genre = new ArrayList<String>();
ArrayList<String> album_art = new ArrayList<String>();

/* Variables for user table */
ArrayList<String> queryUserId = new ArrayList<String>();
ArrayList<String> forname = new ArrayList<String>();
ArrayList<String> surname = new ArrayList<String>();
ArrayList<String> gender = new ArrayList<String>();
ArrayList<String> country = new ArrayList<String>();
ArrayList<String> city = new ArrayList<String>();
ArrayList<String> email = new ArrayList<String>();
ArrayList<String> username = new ArrayList<String>();
ArrayList<String> password = new ArrayList<String>();
ArrayList<String> secretQuestion = new ArrayList<String>();
ArrayList<String> secretAnswer = new ArrayList<String>();
ArrayList<String> dateofbirth = new ArrayList<String>();
ArrayList<String> card = new ArrayList<String>();
ArrayList<String> admin = new ArrayList<String>();

/* Database login details */
String linkToDB = null;
String usernameDB = null;
String passwordDB = null;
Connection connection;
Statement statement;
String query = null;
String queryAdminName = null;
ResultSet resultSet;
ResultSet resultAdminName;

try{

		/* Establish connection with server address, username, and password */
		linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		usernameDB = "adminaudiofarm";
		passwordDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		connection = DriverManager.getConnection(linkToDB,usernameDB,passwordDB);
```

```
		statement = connection.createStatement();

		/* Retrieve all data from the artist table. This is displayed to the user. */
		query = "SELECT * FROM artist LIMIT 0 , 30";
		resultSet = statement.executeQuery(query);

		/* Iterate until all artist related data has been store in the arrays. */
		while(resultSet.next()){
			id.add(resultSet.getString(1));
			name.add(resultSet.getString(2));
			artist.add(resultSet.getString(3));
			album.add(resultSet.getString(4));
			year.add(resultSet.getString(5));
			duration.add(resultSet.getString(6));
			price.add(resultSet.getString(7));
			quantity.add(resultSet.getString(8));
			album_art.add(resultSet.getString(9));
			genre.add(resultSet.getString(10));
		}

}catch(Exception exception){
		out.println("Could not retrieve data from the artist database.");
}

try{

		/* Establish connection with server address, username, and password. */
		linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		usernameDB = "adminaudiofarm";
		passwordDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		connection = DriverManager.getConnection(linkToDB,usernameDB,passwordDB);

		statement = connection.createStatement();

		/* Retrieve all data from the user table. This is displayed to the user. */
		query = "SELECT * FROM user LIMIT 0 , 30";
		resultSet = statement.executeQuery(query);
		/* Iterate until all artist related data has been store in the arrays. */
		while(resultSet.next()){
			queryUserId.add(resultSet.getString(1));
			forname.add(resultSet.getString(2));
			surname.add(resultSet.getString(3));
			gender.add(resultSet.getString(4));
			country.add(resultSet.getString(5));
			city.add(resultSet.getString(6));
			email.add(resultSet.getString(7));
			username.add(resultSet.getString(8));
			password.add(resultSet.getString(9));
			secretQuestion.add(resultSet.getString(10));
			secretAnswer.add(resultSet.getString(11));
			dateofbirth.add(resultSet.getString(12));
			card.add(resultSet.getString(13));
			admin.add(resultSet.getString(14));
		}

		/* Retrieve id and name data from the database match against the current user's identification. */
		queryAdminName = "SELECT id, forname FROM user WHERE id ='" + session.getAttribute("userId") + "'";
		resultAdminName = statement.executeQuery(queryAdminName);

		/* Iterate over the results and assign the name to the string variable. This is used to welcome the user by his/her name. */
		while(resultAdminName.next()){
			welcomeName = resultAdminName.getString(2);
		}

} catch(Exception exception) {
		out.println("Could not retrieve data from the user database.");
}

%>
```

```
<h1>Welcome <%= welcomeName %>!</h1>
<a href="#">Manage Songs Details.</a>
<table id="artists" border="1" color="black">
	<th>ID</th><th>Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Time</th><th>Price</th><th>Quantity</th><th
>Genre</th><th>Action</th>
		<%
		/* Display all songs' details using the already assigned arrays. */
		for (int i = 0; i<id.size(); i++){
			out.println("<tr><form method=\"POST\" action=\"ModifyArtistDetails.jsp\">");
			out.println("<td><input readonly=\"readonly\" type=\"text\" value=\"" + id.get(i) + "\" name=\"id\"
size=\"2\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + name.get(i) + "\" name=\"name\"
maxlength=\"25\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + artist.get(i) + "\" name=\"artist\"
maxlength=\"25\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + album.get(i) + "\" name=\"album\"
maxlength=\"25\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + year.get(i) + "\" name=\"year\" size=\"4\"
maxlength=\"4\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + duration.get(i) + "\" name=\"duration\"
size=\"5\"maxlength=\"5\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + price.get(i) + "\" name=\"price\" size=\"4\"
maxlength=\"4\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + quantity.get(i) + "\" name=\"quantity\" size=\"5\"
maxlength=\"5\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + genre.get(i) + "\" name=\"genre\" size=\"5\"
maxlength=\"5\"/></td>");
			out.println("<td><select name=\"action\"><option value=\"edit\">Edit</option><option
value=\"remove\">Remove</option></select><br/><input type=\"submit\" value=\"Perform\"
name=\"submit\"/></td></tr></tr></form>");
		} %>
	<tr>
		<form method="POST" action="AddArtist.jsp">
			<td>
				?
			</td>
			<td><input type="text" name="name" maxlength="25"/></td>
			<td><input type="text" name="artist" maxlength="25"/></td>
			<td><input type="text" name="album" maxlength="25"/></td>
			<td><input type="text" name="year" size="4" maxlength="4"/></td>
			<td><input type="text" name="duration" size="5" maxlength="5"/></td>
			<td><input type="text" name="price" size="4" maxlength="4"/></td>
			<td><input type="text" name="quantity" size="5" maxlength="5"/></td>
			<td><input type="text" name="genre" size="5" maxlength="5"/></td>
			<td>
				<select>
					<option value="Add">Add</option>
				</select>
				<br/>
				<input type="submit" value="Perform" name="submit"/>
			</td>
		</form>
	</tr>
</table>

<a href="#">Upload Album Art</a>
<table id="album" border="1" color="black">
	<th>ID</th><th>Album</th><th>Link Name</th><th>Action</th><th>Album Art</th>
	<%
	/* Display all the album art data using the already assigned arrays. */
	for (int i = 0; i<id.size(); i++){
			out.println("<tr><form method=\"POST\" action=\"LinkAlbumArt.jsp\">");
			out.println("<td><input readonly=\"readonly\" type=\"text\" value=\"" + id.get(i) + "\" name=\"id\"
size=\"2\"/></td>");
			out.println("<td><input type=\"text\" value=\"" + album.get(i) + "\" name=\"album\"
maxlength=\"25\"/></td>");
			out.println("<td><input type=\"text\" name=\"link\" value=\"" + album_art.get(i) + "\" /></td>");
			out.println("<td><input type=\"submit\" name=\"submit\" value=\"Update\" /></td>");
			out.println("<td><img src=\"" + album_art.get(i) + "\" width=\"60\" height=\"60\"
/></td></form></tr>");
```

```
		}
		%>
</table>

<form action="UploadAlbumArt.jsp" method="POST" enctype="multipart/form-data">
		<input type="file" name="file"/>
		<input type="submit" name="submit" value="Upload" />
</form>

<a href="#">Manage Users</a>
<table id="users" border="1" color="black">
		<th>ID</th><th>Forname</th><th>Surname</th><th>Gender</th><th>Country</th><th>City</th><th>Email</th><th>Userna
me</th><th>Password</th><th>Secret Question</th><th>Secret Answer</th><th>Date of
Birth</th><th>Card</th><th>Admin</th><th>Action</th>
			<%
			/* Display all users' details from the already assigned arrays. */
			for (int j=0; j<queryUserId.size(); j++){
				out.println("<tr><form method=\"POST\" action=\"ModifyUserDetails.jsp\">");
				out.println("<td><input readonly=\"readonly\" type=\"text\" value=\"" + queryUserId.get(j) + "\"
name=\"queryUserId\" size=\"2\"</td>");
				out.println("<td><input type=\"text\" value=\"" + forname.get(j) + "\" name=\"forname\"</td>");
				out.println("<td><input type=\"text\" value=\"" + surname.get(j) + "\" name=\"surname\"</td>");
				out.println("<td><input type=\"text\" value=\"" + gender.get(j) + "\" name=\"gender\"</td>");
				out.println("<td><input type=\"text\" value=\"" + country.get(j) + "\" name=\"country\"</td>");
				out.println("<td><input type=\"text\" value=\"" + city.get(j) + "\" name=\"city\"</td>");
				out.println("<td><input type=\"text\" value=\"" + email.get(j) + "\" name=\"email\"</td>");
				out.println("<td><input type=\"text\" value=\"" + username.get(j) + "\" name=\"username\"</td>");
				out.println("<td><input type=\"text\" value=\"" + password.get(j) + "\" name=\"password\"</td>");
				out.println("<td><input type=\"text\" value=\"" + secretQuestion.get(j) + "\"
name=\"secretQuestion\"</td>");
				out.println("<td><input type=\"text\" value=\"" + secretAnswer.get(j) + "\"
name=\"secretAnswer\"</td>");
				out.println("<td><input type=\"text\" value=\"" + dateofbirth.get(j) + "\" name=\"dateofbirth\"</td>");
				out.println("<td><input type=\"text\" value=\"" + card.get(j) + "\" name=\"card\"</td>");
				out.println("<td><input type=\"text\" value=\"" + admin.get(j) + "\" name=\"admin\"</td>");
				out.println("<td><select name=\"action\"><option value=\"edit\">Edit</option><option
value=\"remove\">Remove</option></select><br/><input type=\"submit\" value=\"Perform\" name=\"submit\"/></td></tr></form>");
			} %>
		</form>

		<tr>
			<form method="POST" action="AddUser.jsp">
				<td>
					?
				</td>
				<td><input type="text" name="forname" maxlength="25"/></td>
				<td><input type="text" name="surname" maxlength="25"/></td>
				<td><input type="text" name="gender" maxlength="6"/></td>
				<td><input type="text" name="country" maxlength="25"/></td>
				<td><input type="text" name="city" maxlength="25"/></td>
				<td><input type="text" name="email" maxlength="25"/></td>
				<td><input type="text" name="username" maxlength="25"/></td>
				<td><input type="text" name="password" maxlength="25"/></td>
				<td><input type="text" name="secretQuestion" maxlength="25"/></td>
				<td><input type="text" name="secretAnswer" maxlength="25"/></td>
				<td><input type="text" name="dateofbirth" maxlength="25"/></td>
				<td><input type="text" name="card" maxlength="25"/></td>
				<td><input type="text" name="admin" maxlength="3"/></td>
				<td>
					<select>
						<option value="Add">Add</option>
					</select>
					<br/>
					<input type="submit" value="Perform" name="submit"/>
				</td>
			</form>
		</tr>
</table>
<a href="#top">Back to the top.</a>
</body>
</html>
```

Cart.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<!-- Prevent unauthorise access to the web application -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm add Songs to Cart</title>
</head>
<body>
<a href="User.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Show items in cart -->

<%
/* Initialise songs related arrays. */
String message = "";
int budget = 0;
ArrayList<String> songsId = new ArrayList<String>();
ArrayList<String> id = new ArrayList<String>();
ArrayList<String> name = new ArrayList<String>();
ArrayList<String> artist = new ArrayList<String>();
ArrayList<String> album = new ArrayList<String>();
ArrayList<String> year = new ArrayList<String>();
ArrayList<String> duration = new ArrayList<String>();
ArrayList<String> price = new ArrayList<String>();
ArrayList<String> quantity = new ArrayList<String>();
ArrayList<String> genre = new ArrayList<String>();
ArrayList<String> album_art = new ArrayList<String>();
ArrayList<String> card = new ArrayList<String>();

/* Get the purchased song id from the session variable */
songsId = (ArrayList<String>) session.getAttribute("songsId");

try{
		for(int j = 0; j<songsId.size(); j++){

				/* Establish connection with server address, username, and password. */
				String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
				String username = "adminaudiofarm";
				String password = "b9YMpCFtm9QPxCjv";
				Class.forName("com.mysql.jdbc.Driver").newInstance();
				Connection connection = DriverManager.getConnection(linkToDB,username,password);

				Statement statement = connection.createStatement();

				/* Retrieve all songs details match against the already assigned song id variable. */
				String query = "SELECT * FROM artist WHERE id ='" + songsId.get(j) + "'";
				ResultSet result = statement.executeQuery(query);

				/* Iterate until all song related data is store into the arrays. */
				while(result.next()){
					id.add(result.getString(1));
					name.add(result.getString(2));
					artist.add(result.getString(3));
					album.add(result.getString(4));
					year.add(result.getString(5));
					duration.add(result.getString(6));
					price.add(result.getString(7));
					quantity.add(result.getString(8));
					album_art.add(result.getString(9));
					genre.add(result.getString(10));
```

```
                        }
                }

} catch(NullPointerException exception){
                message = "The basket is empty.";
} catch(Exception exception){
                out.println("Could not retrieve data from the database.");
}

try{
                /* Establish connection with server address, username, and password. */
                String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
                String username = "adminaudiofarm";
                String password = "b9YMpCFtm9QPxCjv";
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                Connection connection = DriverManager.getConnection(linkToDB,username,password);

                Statement statement = connection.createStatement();
                //String index = (String) session.getAttribute("userId");

                /* Retrieve all id and card data from the user table match against the current user id. */
                String userBudget = "SELECT id, card FROM user WHERE id=" + session.getAttribute("userId") + "";
                ResultSet resultBudget = statement.executeQuery(userBudget);

                /* Get the amount of resource from the card for the user */
                while(resultBudget.next()){
                                card.add(resultBudget.getString(2));
                }
} catch(Exception exception){
                out.println("Could not retrieve user data from the database.");
}

%>
<h1><%= message %></h1>
<h2>You have &#163;<% out.println(card.get(0)); %> available in yout bank account.</h2>
<form method="POST" action="RemoveFromCart.jsp">
<table border=1 color="black">
<th>Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Time</th><th>Price</th><th>Quantity</th><th>Genre</th><th>Albu
m Art</th><th>Remove</th>
<%
/* Iterate over the already assigned arrays to display all songs details. */
for (int i = 0; i<id.size(); i++){
                                        out.println("<tr>");
                                        out.println("<td>" + name.get(i) + "</td>");
                                        out.println("<td>" + artist.get(i) + "</td>");
                                        out.println("<td>" + album.get(i) + "</td>");
                                        out.println("<td>" + year.get(i) + "</td>");
                                        out.println("<td>" + duration.get(i) + "</td>");
                                        out.println("<td>" + price.get(i) + "</td>");
                                        out.println("<td>" + quantity.get(i) + "</td>");
                                        out.println("<td>" + genre.get(i) + "</td>");
                                        out.println("<td><img src=\"" + album_art.get(i) + "\" width=\"60\" height=\"60\" /></td>");
                                        out.println("<td><input type=\"checkbox\" name=\""+ i +"\" value=\"" + id.get(i) + "\" /></td>");
                                        out.println("</tr>");
}
%>
</table>
<br/><input type="submit" value="Remove" />
</form>

<form method="POST" action="Checkout.jsp">
<input type="hidden" name="fromCart" value="yes" />
<input type="submit" value="Proceed" />
</form>

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

Checkout.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm Checkout</title>
</head>
<body>
<a href="User.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Checkout Procedure -->

<%
/* Initialise song related variables */
ArrayList<String> card = new ArrayList<String>();
ArrayList<String> songsId = new ArrayList<String>();
String message = "";
String name = "";
String artist = "";
String price = "";
String warning = "";
int songQuantity = 0;
Double budget = 0.0;

/* Get the all purchased song ids from the session variable */
songsId = (ArrayList<String>) session.getAttribute("songsId");

try{
		/* Establish a connection to the database with server address, username, and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String username = "adminaudiofarm";
		String password = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB,username,password);
		Statement statement = connection.createStatement();
		//String index = (String) session.getAttribute("userId");

		/* Select all user columns matching the currently logged in user id. */
		String userBudget = "SELECT id, card FROM user WHERE id=" + session.getAttribute("userId") + "";

		/* Iterate over the array to find the user's credit card amount. */
		ResultSet resultBudget = statement.executeQuery(userBudget);
		while(resultBudget.next()){
			card.add(resultBudget.getString(2));
		}
} catch(Exception exception){
		out.println("Could not retrieve user data from the database.");
}

/* Iterate over the song ids array until all the user and song columns have been updated. */
for(int j = 0; j<songsId.size(); j++){
		//out.println(songsId.get(j));

		try{
			/* Establish a connection to the database with server address, username, and password. */
			String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
			String username = "adminaudiofarm";
			String password = "b9YMpCFtm9QPxCjv";
			Class.forName("com.mysql.jdbc.Driver").newInstance();
			Connection connection = DriverManager.getConnection(linkToDB,username,password);
```

```
                        Statement statementSongGet = connection.createStatement();

                        /* Select all artist details matching the currently logged in user. */
                        String querySongQuantity = "SELECT * FROM artist WHERE id ='" + songsId.get(j) + "'";

                        /* Iterate until all user artist details are received. */
                        ResultSet resultSongQuantity = statementSongGet.executeQuery(querySongQuantity);
                        while(resultSongQuantity.next()){
                                name = resultSongQuantity.getString(2);
                                artist = resultSongQuantity.getString(3);
                                price = resultSongQuantity.getString(7);
                                songQuantity = Integer.parseInt(resultSongQuantity.getString(8));
                        }

                        /* Decrement the purchased song quantity unless it is zero. */
                        if(songQuantity == 0){
                                warning = "<h1>" + name + " by " + artist + " is temporary out of stock. </h1>";
                        } else {
                                songQuantity -= 1;
                        }

                        /* Perform floating point calculation to determine the final user credit card amount. */
                        budget = Double.parseDouble(card.get(0)) - Double.parseDouble(price);
                        //out.println(budget);

                        /* Decrease the purchased song quantity in the database. */
                        String queryUpdateSongQuantity = "UPDATE artist SET quantity='" + songQuantity + "' WHERE id ='" + songsId.get(j)
+ "'";
                        statementSongGet.executeUpdate(queryUpdateSongQuantity);

                        /* Decrease the user credit card amount in the database. */
                        String queryUpdateUserBudget = "UPDATE user SET card=" + budget + " WHERE id =" +
session.getAttribute("userId") + "";
                        statementSongGet.executeUpdate(queryUpdateUserBudget);

                        /* Update the past purchased song database table with the currently logged user id and purchased song ids. */
                        String queryHistory = "INSERT INTO history (id, user_id, song_id) VALUES(NULL, " + session.getAttribute("userId") +
", " + songsId.get(j) + ")";
                        statementSongGet.executeUpdate(queryHistory);

                } catch (Exception exception){
                        out.println("Could not retrieve data from the database.");
                }

                message = "<h1>Your purchase was successful!</h1>";

                /* Remove the already purchased song ids from the session. */
                session.removeAttribute("songsId");
}

%>

<%= warning %>
<%= message %>

</body>
</html>
```

History.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
        pageContext.forward("Unauthorised.jsp");
}
%>
```

```
<html>
<head>
		<title>Audio Farm History</title>
</head>
<body>
<a href="User.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- User History -->

<%
/* Initialise song related variables */
String message = "";
ArrayList<String> user_id = new ArrayList<String>();
ArrayList<String> song_id = new ArrayList<String>();
ArrayList<String> name = new ArrayList<String>();
ArrayList<String> artist = new ArrayList<String>();
ArrayList<String> album = new ArrayList<String>();
ArrayList<String> year = new ArrayList<String>();
ArrayList<String> time = new ArrayList<String>();
ArrayList<String> price = new ArrayList<String>();
ArrayList<String> album_art = new ArrayList<String>();
/* Initialise recommended song related variables */
ArrayList<String> nameRecommend = new ArrayList<String>();
ArrayList<String> artistRecommend = new ArrayList<String>();
ArrayList<String> albumRecommend = new ArrayList<String>();
ArrayList<String> yearRecommend = new ArrayList<String>();
ArrayList<String> timeRecommend = new ArrayList<String>();
ArrayList<String> priceRecommend = new ArrayList<String>();
ArrayList<String> album_artRecommend = new ArrayList<String>();
ArrayList<String> quantityRecommend = new ArrayList<String>();
ArrayList<String> genre = new ArrayList<String>();
ArrayList<String> genreRecommend = new ArrayList<String>();
int card = 0;

try{
		/* Establish a connection to the database with server address, username, and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String username = "adminaudiofarm";
		String password = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB,username,password);
		Statement statement = connection.createStatement();

		/* Join the user, artist, and history tables into one. Then get all entries. */
		String query = "SELECT * FROM history LEFT JOIN artist ON history.song_id=artist.id LEFT JOIN user ON history.user_id=user.id
WHERE history.user_id=" + session.getAttribute("userId") + " LIMIT 0, 30";

		/* Iterate over the results until all song details are assigned. */
		ResultSet result = statement.executeQuery(query);
		while(result.next()){
			name.add(result.getString(5));
			artist.add(result.getString(6));
			album.add(result.getString(7));
			year.add(result.getString(8));
			time.add(result.getString(9));
			price.add(result.getString(10));
			album_art.add(result.getString(12));
			genre.add(result.getString(13));
			card = Integer.parseInt(result.getString(26));
		}

		/* Get all songs from the database that match the purchased song's genre. */
		String queryRecommend = "SELECT * FROM artist WHERE genre='" + genre.get(0) + "'";

		/* Iterate over the result until all recommended song details are assigned. */
		ResultSet resultRecommend = statement.executeQuery(queryRecommend);
		while(resultRecommend.next()){
			nameRecommend.add(resultRecommend.getString(2));
			artistRecommend.add(resultRecommend.getString(3));
			albumRecommend.add(resultRecommend.getString(4));
```

```
                        yearRecommend.add(resultRecommend.getString(5));
                        timeRecommend.add(resultRecommend.getString(6));
                        priceRecommend.add(resultRecommend.getString(7));
                        quantityRecommend.add(resultRecommend.getString(8));
                        album_artRecommend.add(resultRecommend.getString(9));
                        genreRecommend.add(resultRecommend.getString(10));
                }

} catch(Exception exception){
                message = "Could not retrieve data from the database.";
}

%>

<h1><%= message %></h1>
<h2>Recently Purchased Songs</h2>
<h3>Remaining Credit: &#163; <%= card %></h3>
<table border=1 color="black">
                <th>Song
Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Duration</th><th>Price</th><th>Genre</th><th>Album Art</th>
                <%
                /* Iterate over the purchased songs arrays until all data is shown to the user. */
                for(int i = 0; i<name.size(); i++){
                        out.println("<tr><td>" + name.get(i) + "</td>");
                        out.println("<td>" + artist.get(i) + "</td>");
                        out.println("<td>" + album.get(i) + "</td>");
                        out.println("<td>" + year.get(i) + "</td>");
                        out.println("<td>" + time.get(i) + "</td>");
                        out.println("<td>" + price.get(i) + "</td>");
                        out.println("<td>" + genre.get(i) + "</td>");
                        out.println("<td><img src=\"" + album_art.get(i) + "\" width=\"60\" height=\"60\" /></td></tr>");
                }
                %>
</table>
<h2>We recommend you:</h2>
<table border=1 color="black">
                <th>Song
Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Duration</th><th>Price</th><th>Quantity</th><th>Genre</th><th>Album
Art</th>
                <%
                /* Iterate over the recommended songs arrays until all data is shown to the user. */
                for (int j = 0;j<nameRecommend.size(); j++){
                        out.println("<tr><td>" + nameRecommend.get(j) + "</td>");
                        out.println("<td>" + artistRecommend.get(j) + "</td>");
                        out.println("<td>" + albumRecommend.get(j) + "</td>");
                        out.println("<td>" + yearRecommend.get(j) + "</td>");
                        out.println("<td>" + timeRecommend.get(j) + "</td>");
                        out.println("<td>" + priceRecommend.get(j) + "</td>");
                        out.println("<td>" + quantityRecommend.get(j) + "</td>");
                        out.println("<td>" + genreRecommend.get(j) + "</td>");
                        out.println("<td><img src=\"" + album_artRecommend.get(j) + "\" width=\"60\" heigth=\"60\" /></td></tr>");
                }
                %>
</table>
</body>
</html>
```

index.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<html>
<head>
                <title>Audio Farm Home</title>
</head>
<body>
<a href="Login.jsp">Login</a>
<a href="Register.jsp">Register</a>
```

```
<!-- Get all songs for sale -->
<%
/* Initialise song related variables */
ArrayList<String> id = new ArrayList<String>();
ArrayList<String> name = new ArrayList<String>();
ArrayList<String> artist = new ArrayList<String>();
ArrayList<String> album = new ArrayList<String>();
ArrayList<String> year = new ArrayList<String>();
ArrayList<String> duration = new ArrayList<String>();
ArrayList<String> price = new ArrayList<String>();
ArrayList<String> quantity = new ArrayList<String>();
ArrayList<String> album_art = new ArrayList<String>();
ArrayList<String> genre = new ArrayList<String>();

try{
	/* Establish a connection to the database with server address, username, and password. */
	String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
	String username = "adminaudiofarm";
	String password = "b9YMpCFtm9QPxCjv";
	Class.forName("com.mysql.jdbc.Driver").newInstance();
	Connection connection = DriverManager.getConnection(linkToDB,username,password);
	Statement statement = connection.createStatement();

	/* Select all songs from the artist database table */
	String query = "SELECT* FROM`artist` LIMIT 0 , 30";

	/* Iterate over the results until all song details are assigned. */
	ResultSet resultSet = statement.executeQuery(query);
	while(resultSet.next()){
		id.add(resultSet.getString(1));
		name.add(resultSet.getString(2));
		artist.add(resultSet.getString(3));
		album.add(resultSet.getString(4));
		year.add(resultSet.getString(5));
		duration.add(resultSet.getString(6));
		price.add(resultSet.getString(7));
		quantity.add(resultSet.getString(8));
		album_art.add(resultSet.getString(9));
		genre.add(resultSet.getString(10));
	}

}catch(Exception exception){
	out.println("Could not retrieve data from the database.");
}

%>
<a href="#" style="text-decoration: none">Cart: 0<img src="img/basket-empty.png" style="border: none"/></a>
<h1>Welcome to Audio Farm</h1>
<table border=1 color="black">
<th>ID</th><th>Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Time</th><th>Price</th><th>Quantity</th><th>Genre</t
h><th>Album Art</th>
<%
/* Iterate over the database songs arrays until all data is shown to the user. */
for (int i = 0; i<id.size(); i++){
		out.println("<tr>");
		out.println("<td>" + id.get(i) + "</td>");
		out.println("<td>" + name.get(i) + "</td>");
		out.println("<td>" + artist.get(i) + "</td>");
		out.println("<td>" + album.get(i) + "</td>");
		out.println("<td>" + year.get(i) + "</td>");
		out.println("<td>" + duration.get(i) + "</td>");
		out.println("<td>" + price.get(i) + "</td>");
		out.println("<td>" + quantity.get(i) + "</td>");
		out.println("<td>" + genre.get(i) + "</td>");
		out.println("<td><img src=\"" + album_art.get(i) + "\" width=\"60\" heigth=\"60\"
/></td></form></tr>");
		out.println("</tr>");
}
%>
</table>
```

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

LinkAlbumArt.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<%
/* Establish a connection to the database with server address, username, and password. */
String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
String usernameForDB = "adminaudiofarm";
String passwordForDB = "b9YMpCFtm9QPxCjv";
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
Statement statement = connection.createStatement();

/* Update the album art cover address - default img directory on the and user submited file name. */
String query = "UPDATE artist SET album_art='img/" + request.getParameter("link") + "' WHERE id=" + request.getParameter("id") + "";
statement.executeUpdate(query);

%>

<jsp:forward page="Admin.jsp" />
```

Login.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<html>
<head>
		<title>Audio Farm Login</title>
</head>
<body>
<a href="index.jsp">Home</a>
<a href="Register.jsp">Register</a>

<form name="login" id="login" method="POST" action="ProcessLogin.jsp">
		<table>
			<tr>
				<img src="img/checklist.png"/>
			</tr>
			<tr>
				<td>Username: </td>
				<td><input type="text" id="username" name="username"/></td>
			</tr>
			<tr>
				<td>Password: </td>
				<td><input type="password" id="password" name="password"/></td>
			</tr>
			<tr>
				<td><input type="submit" id="submit" name="submit" value="Login"/></td>
				<td><input type="reset" id="reset" name="reset" value="Try Again"></td>
			</tr>
		</table>
</form>

</body>
</html>
```

Logout.jsp

```
<%@ page contentType="text/html" %>
```

```
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>

<html>
<head>
		<title>Audio Farm Logout</title>
</head>
<body>
<a href="index.jsp">Home</a>
<a href="Login.jsp">Login</a>

<!-- Terminate user or admin session -->
<%
/* Initialise session related variables. */
String message = null;
String username = (String) session.getAttribute("name");
Long sessionCreationTime = session.getCreationTime();
Long sessionLastTime = session.getLastAccessedTime();
Date humanTimeCreation = new Date(sessionCreationTime * 1000);
Date humanTimeLast = new Date(sessionLastTime * 1000);

try{
		/* Destroy the current session by removing user related attributes. */
		session.removeAttribute("userId");
		session.removeAttribute("name");
		session.removeAttribute("privileges");
		/* Used to notify which user has been logged out. */
		message = username;
} catch(NullPointerException e){
		message = "Logout problem" + e.toString();
}

if(message == null){
		/* The user session has automatically expired. */
		message = "Please login first.";
} else {
		/* Display user notification message. */
		message = message.concat(" was successfully logged out.<br/>");
		//message = message.concat("You started your session on: " + humanTimeCreation.getDate() + "/" +
humanTimeCreation.getMonth() + "/" + humanTimeCreation.getYear() + "<br/>");
		//message = message.concat("The last time you logged in was at: " + humanTimeLast.getDate() + "/" +
humanTimeLast.getMonth() + "/" + humanTimeLast.getYear());
}

%>

<h1><%= message %></h1>

</body>
</html>
```

ModifyArtistDetails.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm Admin</title>
</head>
<body>
<a id="top" href="Admin.jsp">Home</a>
<a href="Logout.jsp">Logout</a>
```

```
<!-- Get artist details from the form and either remove or update -->
<%
/* Initialise song related variables */
String message = "";
String id = request.getParameter("id");
String name = request.getParameter("name");
name.trim();
String artist = request.getParameter("artist");
artist.trim();
String album = request.getParameter("album");
album.trim();
String year = request.getParameter("year");
year.trim();
String duration = request.getParameter("duration");
duration.trim();
String price = request.getParameter("price");
String quantity = request.getParameter("quantity");
String genre = request.getParameter("genre");
Double doublePrice = Double.parseDouble(price);
int intQuantity = Integer.parseInt(quantity);
String action = request.getParameter("action");

/* Perform simple form validation whether the submitted text fields are empty. */
if(name.isEmpty()){
	out.println("<p>Name field is empty!</p>");
} else if(artist.isEmpty()){
	out.println("<p>Artist field is empty!</p>");
} else if (album.isEmpty()){
	out.println("<p>Album field is empty!</p>");
} else if (year.isEmpty()){
	out.println("<p>Year field is empty!</p>");
} else if (duration.isEmpty()){
	out.println("<p>Duration field is empty!</p>");
} else if(price.isEmpty()){
	out.println("<p>Price field is empty!</p>");
} else if(quantity.isEmpty()){
	out.println("<p>Quantity field is empty!</p>");
} else {

	/* Edit selected song details. */
	if(action.startsWith("edit")){
		/* Establish a connection to the database with server address, username, and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String usernameForDB = "adminaudiofarm";
		String passwordForDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
		Statement statement = connection.createStatement();

		/* Update all song related fields in the database. */
		String query = "UPDATE artist SET name='"
		+ name + "',                    artist='"
		+ artist + "',                  album='"
		+ album + "',                   year='"
		+ year + "',                    time='"
		+ duration + "',                price='"
		+ doublePrice + "',             quantity='"
		+ intQuantity + "',             genre='"
		+ genre + "' WHERE id='" + id + "'";

		statement.executeUpdate(query);

		message = "<h2>Artist details were successfully stored in the database.</h2>";

	/* Remove a selected song. */
	} else if(action.startsWith("remove")) {
		/* Establish a connection to the database with server address, username, and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String usernameForDB = "adminaudiofarm";
		String passwordForDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
```

```
                        Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
                        Statement statement = connection.createStatement();

                        /* Delete the selected song by id. */
                        String queryDelete = "DELETE FROM artist WHERE id='" + id + "'";
                        statement.executeUpdate(queryDelete);
                        message = "<h2>The user was successfully deleted from the database.</h2>";
                }
}
%>

<%= message %>
</body>
</html>
```

ModifyUserDetails.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
        pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
        <title>Audio Farm Admin</title>
</head>
<body>
<a id="top" href="Admin.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Get user details from the form and either remove or update -->
<%
/* Initialise user related variables. */
String message = "";
String queryUserId = request.getParameter("queryUserId");
String forname = request.getParameter("forname");
forname.trim();
String surname = request.getParameter("surname");
surname.trim();
String gender = request.getParameter("gender");
surname.trim();
String country = request.getParameter("country");
surname.trim();
String city = request.getParameter("city");
surname.trim();
String email = request.getParameter("email");
surname.trim();
String username = request.getParameter("username");
username.trim();
String password = request.getParameter("password");
String secretQuestion = request.getParameter("secretQuestion");
password.trim();
String secretAnswer = request.getParameter("secretAnswer");
secretAnswer.trim();
String dateofbirth = request.getParameter("dateofbirth");
secretAnswer.trim();
String action = request.getParameter("action");

/* Get the user credit card amount unless zero. */
String card = request.getParameter("card");
Integer.parseInt(card);
int intCard = 0;
if(!card.isEmpty()){
        intCard = Integer.parseInt(card);
}
```

```
/* Get the admin privileges unless zero. */
String admin = request.getParameter("admin");
int intAdmin = 0;
if(!admin.isEmpty()){
	intAdmin = Integer.parseInt(admin);
}

/* Perform simple form validation checking whether the submitted fields are empty. */
if(forname.isEmpty()){
	out.println("<p>Forname field is empty!</p>");
} else if(surname.isEmpty()){
	out.println("<p>Surname field is empty!</p>");
} else if (gender.isEmpty()){
	out.println("<p>Gender field is empty!</p>");
} else if (country.isEmpty()){
	out.println("<p>Country field is empty!</p>");
} else if (city.isEmpty()){
	out.println("<p>City field is empty!</p>");
} else if(email.isEmpty()){
	out.println("<p>Email field is empty!</p>");
} else if(username.isEmpty()){
	out.println("<p>Username field is empty!</p>");
} else if(password.isEmpty()){
	out.println("<p>Password field is empty!</p>");
} else if(password.length() < 6){
	out.println("<p>The password should be more than 6 characters long!</p>");
} else if(secretAnswer.isEmpty()){
	out.println("<p>SecretAnswer field is empty!</p>");
} else if(dateofbirth.isEmpty()){
	out.println("<p>Date of Birth field is empty!</p>");
} else {
	/* Edit selected user details. */
	if(action.startsWith("edit")){
		/* Establish a connection to the database with server address, username, and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String usernameForDB = "adminaudiofarm";
		String passwordForDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
		Statement statement = connection.createStatement();

		/* Update all user related fields in the database. Use 512 bit password hashing for additional security. */
		String query = "UPDATE user SET forname='"
		+ forname + "', surname='"
		+ surname + "', gender='"
		+ gender + "', country='"
		+ country + "', city='"
		+ city + "', email='"
		+ email + "', username='"
		+ username + "', password=SHA2('"
		+ password + "', 512), secretquestion='"
		+ secretQuestion + "', secretanswer='"
		+ secretAnswer + "', dateofbirth='"
		+ dateofbirth + "', card='"
		+ card + "', admin="
		+ admin + " WHERE id=" + queryUserId + "";

		statement.executeUpdate(query);

		message = "<h2>User details were successfully stored in the database.</h2>";

	/* Remove the selected user from the database. */
	} else if(action.startsWith("remove")) {
		/* Establish a connection to the database with server address, username, and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String usernameForDB = "adminaudiofarm";
		String passwordForDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
		Statement statement = connection.createStatement();
```

```
                        /*  Delete the selected user by matching user id number. */
                        String queryDelete = "DELETE FROM user WHERE id='" + queryUserId + "'";
                        statement.executeUpdate(queryDelete);
                        message = "<h2>The user was successfully deleted from the database.</h2>";
                }
}
%>

<%= message %>
</body>
</html>
```

ProcessLogin.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<html>
<head>
        <title>Audio Farm Process Login</title>
</head>
<body>
<a href="index.jsp">Home</a>
<a href="Register.jsp">Register</a>

<!-- Authorise user against DB -->
<%
/* Initialise user related variables. */
String queryUserId = null;
int userId = 0;
String queryForname = null;
String queryUsername = null;
String queryPassword = null;
String queryAdmin = null;
Boolean userVerified = false;
Boolean adminVerified = false;
Boolean privileges = false;
String verifiedForname = null;
String message = null;
String link = "";

/* Get the submitted form username and password. */
String formUsername = request.getParameter("username");
formUsername.trim();
String formPassword = request.getParameter("password");
formPassword.trim();

/* Initialise password related variables. */
String cryptedPassword = "";
String originalPassword = "";

/* Process user authentication only if both text fields are not empty. */
if(formUsername.isEmpty()){
        out.println("<p>Username field was empty.</p>");
} else if (formPassword != null && formPassword.isEmpty()){
        out.println("<p>Password field was empty.</p>");
} else {

        try{
                /* Establish a connection to the database with server address, username, and password. */
                String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
                String username = "adminaudiofarm";
                String password = "b9YMpCFtm9QPxCjv";
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                Connection connection = DriverManager.getConnection(linkToDB,username,password);
                Statement statement = connection.createStatement();

                /* Get 512 bit hashed password based on the submitted password field. */
                String queryCrypt = "SELECT SHA2('" + formPassword + "', 512)";
                ResultSet resultCrypt = statement.executeQuery(queryCrypt);
                /* Assign the crypted password for later use. */
```

```
                        while(resultCrypt.next()){
                                cryptedPassword = resultCrypt.getString(1);
                        }

                        /* Get all user related details from the database. */
                        String query = "SELECT * FROM user LIMIT 0 , 30";
                        ResultSet resultSet = statement.executeQuery(query);

                        /* Iterate over the result until all user related details are assigned. */
                        while(resultSet.next()){
                                queryUserId = resultSet.getString(1);
                                queryForname = resultSet.getString(2);
                                queryUsername = resultSet.getString(8);
                                queryPassword = resultSet.getString(9);
                                queryAdmin = resultSet.getString(14);
                                /* Check whether there is a match between the submitted username and password against the database
records. */
                                if(queryUsername.matches(formUsername) && queryPassword.matches(cryptedPassword)){
                                        /* Assign user privileges by default. */
                                        userVerified = true;
                                        verifiedForname = queryForname;
                                        userId = Integer.parseInt(queryUserId);
                                        /* Assign user privileges if the field equals one. */
                                        if(Integer.parseInt(queryAdmin) == 1){
                                                adminVerified = true;
                                                verifiedForname = queryForname;
                                                privileges = true;
                                                userId = Integer.parseInt(queryUserId);
                                        }
                                }
                        }

                }catch(Exception exception){
                        out.println("Could not retrieve data from the database.");
                }

                /* If the user/administrator has been successfully authenticated. */
                if(userVerified){
                        message = verifiedForname + " successfully logged in.";
                        if(adminVerified){
                                /* Admin home page. */
                                link = " <a href=\"Admin.jsp\">Admin</a>";
                        } else {
                                /* User home page. */
                                link = " <a href=\"User.jsp\">User</a>";
                        }
                        /* Set the user related session attributed. */
                        session.setAttribute("userId", userId);
                        session.setAttribute("name", verifiedForname);
                        session.setAttribute("privileges", privileges);
                } else {
                        /* Notify the user upon unsuccessful authentication. */
                        message = "Username or password incorrect.";
                }
}

%>

<h1><%= message %><%= link %></h1>

</body>
</html>
```

ProcessRegistration.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<html>
<head>
        <title>Audio Farm Process Registration</title>
```

```
</head>
<body>
<a href="index.jsp">Home</a>

<!-- Get all form entries -->
<%
/* Initialise user related variables. */
Boolean duplicate = false;
String message = "";
String forname = request.getParameter("forname");
forname.trim();
String surname = request.getParameter("surname");
surname.trim();
String gender = request.getParameter("gender");
String country = request.getParameter("country");
String city = request.getParameter("city");
String email = request.getParameter("email");
String username = request.getParameter("username");
username.trim();
String password = request.getParameter("password");
password.trim();
String password2 = request.getParameter("password2");
password2.trim();
String secretQuestion = request.getParameter("secretQuestion");
String secretAnswer = request.getParameter("secretAnswer");
secretAnswer.trim();
String dateofbirth = request.getParameter("dateofbirth");

/* Get the credit card amount from the submitted form. */
String card = request.getParameter("card");
Integer.parseInt(card);
int intCard = 0;

/* Depending on the submitted value assign user credit card amount. */
if(Integer.parseInt(card) == 1){
		intCard = 100;
} else if(Integer.parseInt(card) == 2){
		intCard = 200;
} else if(Integer.parseInt(card) == 3){
		intCard = 2000;
}

/* Perform simple form validation checking whether the submitted form fields are empty. */
if(forname.isEmpty()){
		out.println("<p>Forname field is empty!</p>");
} else if(surname.isEmpty()){
		out.println("<p>Surname field is empty!</p>");
} else if(email.isEmpty()){
		out.println("<p>Email field is empty!</p>");
} else if(password.isEmpty()){
		out.println("<p>Password field is empty!</p>");
} else if(password.length() < 6){
		out.println("<p>Your password should be more than 6 characters long!</p>");
} else if(password2.isEmpty()){
		out.println("<p>Retype Password field is empty!</p>");
		password2.trim();
} else if(!password2.matches(password)){
		out.println("<p>The two passwords do not match!</p>");
} else if(secretAnswer.isEmpty()){
		out.println("<p>SecretAnswer field is empty!</p>");
} else if(dateofbirth.isEmpty()){
		out.println("<p>Date of Birth field is empty!</p>");
} else {

		/* Establish a connection to the database with server address, username, and password. */
		String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
		String usernameForDB = "adminaudiofarm";
		String passwordForDB = "b9YMpCFtm9QPxCjv";
		Class.forName("com.mysql.jdbc.Driver").newInstance();
		Connection connection = DriverManager.getConnection(linkToDB, usernameForDB, passwordForDB);
		Statement statement = connection.createStatement();
```

```
		/* Get all usernames from the database. */
		String queryForDuplicate = "SELECT username FROM user";
		ResultSet resultForDuplicate = statement.executeQuery(queryForDuplicate);
		/* Iterate over the result while checking for username duplicates. */
		while(resultForDuplicate.next()){
			if(resultForDuplicate.getString(1).startsWith(username)){
				duplicate = true;
				message = "<h1>This username already exists!</h1>";
			}
		}
		/* Perform only if there are no username duplicates. */
		if(!duplicate){

			/* Create new user with all form submitted user details. */
			String query = "INSERT INTO user (id, forname, surname, gender, country, city, email, username, password,
secretquestion, secretanswer, dateofbirth, card, admin) VALUES (NULL,'"
			+ forname + "', '"
			+ surname + "', '"
			+ gender + "', '"
			+ country + "', '"
			+ city + "', '"
			+ email + "', '"
			+ username + "', SHA2('"
			+ password + "', 512), '"
			+ secretQuestion + "', '"
			+ secretAnswer + "', '"
			+ dateofbirth + "', '"
			+ intCard +"', 0)";

			/*String query2 = "UPDATE user SET forname='"+forname+"' WHERE id=7";*/

			statement.executeUpdate(query);

			message = "<h1>Registration was successful!</h1>";
			message = message.concat("<h1>" + forname + " welcome to Audio Farm. <a href=\"Login.jsp\">Login</a></h1>");
		}
}

%>

<%= message %>
</body>
</html>
```

Register.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<html>
<head>
		<title>Audio Farm Register</title>
</head>
<body>
<a href="index.jsp">Home</a>

<h1>Please, register to buy items<br/> and view your shopping cart.</h1>

<form name="register" id="register" method="POST" action="ProcessRegistration.jsp">
		<table>
		<tr>
			<td><img src="img/chat-support.png"/></td>
		</tr>
		<tr>
			<td>
				<label for="forname">Forname: </label>
			</td>
			<td>
				<input type="text" name="forname" id="forname"/>
			</td>
```

```
	</tr>
	<tr>
		<td>
			<label for="surname">Surname: </label>
		</td>
		<td>
			<input type="text" name="surname" id="surname"/>
		</td>

	</tr>
	<tr>
		<td>
			<label for="gender">Gender: </label>
		</td>
		<td>
			<input type="radio" name="gender" id="gender" value="Male" checked="checked"/>Male
			<input type="radio" name="gender" id="gender" value="Female"/>Female
		</td>
	</tr>
	<tr>
		<td>
			<label for="country">Country: </label>
		</td>
		<td>
			<select name="country" id="country">
				<option value="England">England</option>
				<option value="Wales">Wales</option>
				<option value="Scotland">Scotland</option>
				<option value="Nothern Ireland">Northern Ireland</option>
				<option value="Republic of Ireland">&#201;ire</option>
			</select>
		</td>
	</tr>
	<tr>
		<td>
			<label for="city">City: </label>
		</td>
		<td>
			<select name="city" id="city">
				<option value="London">London</option>
				<option value="Cardiff">Cardiff</option>
				<option value="Edinburgh">Edinburgh</option>
				<option value="Belfast">Belfast</option>
			</select>
		</td>
	</tr>
	<tr>
		<td>
			<label for="email">Email: </label>
		</td>
		<td>
			<input type="text" name="email" id="email"/><br/>
		</td>
	</tr>
	<tr>
		<td>
			<label for="username">Username: </label>
		</td>
		<td>
			<input type="text" name="username" id="username"/>
		</td>
	</tr>
	<tr>
		<td>
			<label for="password">Password: </label>
		</td>
		<td>
			<input type="password" name="password" id="password"/>
		</td>
	</tr>
	<tr>
```

```
                        <td>
                                <label for="password2">Retype Password: </label>
                        </td>
                        <td>
                                <input type="password" name="password2" id="password2"/>
                        </td>
                </tr>
                <tr>
                        <td>
                                <label for="secret">Select Secret Question: </label>
                        </td>
                        <td>
                                <select name="secretQuestion" id="secret">
                                        <option value="Whats your favourite movie">What's your favourite movie?</option>
                                        <option value="Whats your favourite band">What's your favourite band?</option>
                                        <option value="Whats your favourite song">What's your favourite song?</option>
                                        <option value="Whats your best friends last name">What's your best friend's last
name?</option>
                                </select>
                        </td>
                </tr>
                <tr>
                        <td>
                                <label for="secretAnswer">Secret Answer: </label>
                        </td>
                        <td>
                                <input type="text" name="secretAnswer" id="secretAnswer"/>
                        </td>
                </tr>
                <tr>
                        <td>
                                <label for="dateofbirth">Date of Birth: </label>
                        </td>
                        <td>
                                <input type="text" name="dateofbirth" id="dateofbirth" value="DD/MM/YYYY"/>
                        </td>
                </tr>
                <tr>
                        <td>
                                <label for="card">Card: </label>
                        </td>
                        <td>
                                <select name="card" id="card">
                                        <option value="1">VISA &#8356;100</option>
                                        <option value="2">Master Card &#8356;200</option>
                                        <option value="3">AmEx &#8356;2000</option>
                                </select>
                        </td>
                </tr>
                <tr>
                        <td>
                                <input type="submit" name="submit" id="submit" value="Register"/>
                        </td>
                        <td>
                                <input type="reset" name="reset" id="reset" value="Start Over"/>
                        </td>
                </tr>
                </table>
</form>
</body>
</html>
```

RemoveFromCart.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
```

```
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm Remove Songs to Cart</title>
</head>
<body>
<a href="User.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Remove Songs from Cart -->

<%
/* Initialise song related variables. */
int index = 0;
ArrayList<String> songsId = new ArrayList<String>();
ArrayList<String> id = new ArrayList<String>();

/* Get all purchased songs from the session array. */
songsId = (ArrayList<String>) session.getAttribute("songsId");

/* Clone the session song id array to another id array. */
for(int i = 0; i<songsId.size(); i++){
		id.add(request.getParameter(String.valueOf(i)));
}
out.println(request.getParameter(String.valueOf(1)));

/* Remove only selected ids from the song ids array. */
for(int j = 0; j<id.size(); j++){
		if(songsId.contains(id.get(j))){
			index = songsId.indexOf(id.get(j));
			songsId.remove(index);
		} else {
			out.println("Nothing to remove");
		}
}

/* Replace the session song ids array with the updated array. */
session.removeAttribute("songsId");
session.setAttribute("songsId", songsId);

%>
<jsp:forward page="Cart.jsp" />
</body>
</html>
```

Search.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm User Search</title>
</head>
<body>
<a href="User.jsp">Home</a>
<a href="Logout.jsp">Logout</a>

<!-- Search for songs -->
```

```
<%
/* Initialise song related variables. */
ArrayList<String> id = new ArrayList<String>();
ArrayList<String> name = new ArrayList<String>();
ArrayList<String> artist = new ArrayList<String>();
ArrayList<String> album = new ArrayList<String>();
ArrayList<String> year = new ArrayList<String>();
ArrayList<String> duration = new ArrayList<String>();
ArrayList<String> price = new ArrayList<String>();
ArrayList<String> quantity = new ArrayList<String>();
ArrayList<String> genre = new ArrayList<String>();
ArrayList<String> album_art = new ArrayList<String>();
/* Initialise database related variables. */
String keyword = null;
String searchType = null;
String linkToDB;
String username;
String password;
Connection connection;
Statement statement;
ResultSet result;
String query = null;

try{
	/* Get the submitted search keyword. */
	keyword = request.getParameter("keyword");
	//out.println("keyword: " + keyword);
	/* Get the submitted search criterion. */
	searchType = request.getParameter("searchType");
	//out.println("option: " + searchType);

	/* Establish a connection to the database with server address, username, and password. */
	linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
	username = "adminaudiofarm";
	password = "b9YMpCFtm9QPxCjv";
	Class.forName("com.mysql.jdbc.Driver").newInstance();
	connection = DriverManager.getConnection(linkToDB,username,password);
	statement = connection.createStatement();

	/* Search for a song in the database that matches the user submitted keyword and the user submitted column (criterion). */
	query = "SELECT * FROM artist WHERE MATCH (" + searchType + ") AGAINST ('+" + keyword + "' IN BOOLEAN MODE)";
	result = statement.executeQuery(query);

	/* Iterate over the result until all song related details are assign. */
	while(result.next()){
		id.add(result.getString(1));
		name.add(result.getString(2));
		artist.add(result.getString(3));
		album.add(result.getString(4));
		year.add(result.getString(5));
		duration.add(result.getString(6));
		price.add(result.getString(7));
		quantity.add(result.getString(8));
		album_art.add(result.getString(9));
		genre.add(result.getString(10));
	}

} catch (NullPointerException exception){
	/* Notify the user if no results were found. */
	out.println("No results were found.");
}

%>

<form method="POST" action="AddSongsToCart.jsp">
<table border=1 color="black">
<th>Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Time</th><th>Price</th><th>Quantity</th><th>Genre</th><th>Albu
m Art</th><th>Buy</th>
<%
/* Iterate over the found songs arrays until all data is shown to the user. */
for (int i = 0; i<id.size(); i++){
```

```
                                out.println("<tr>");
                                out.println("<td>" + name.get(i) + "</td>");
                                out.println("<td>" + artist.get(i) + "</td>");
                                out.println("<td>" + album.get(i) + "</td>");
                                out.println("<td>" + year.get(i) + "</td>");
                                out.println("<td>" + duration.get(i) + "</td>");
                                out.println("<td>" + price.get(i) + "</td>");
                                out.println("<td>" + quantity.get(i) + "</td>");
                                out.println("<td>" + genre.get(i) + "</td>");
                                out.println("<td><img src=\"" + album_art.get(i) + "\" width=\"60\" heigth=\"60\" /></td>");
                                out.println("<input type=\"hidden\" name=\"submitted\" value=\"yes\" />");
                                out.println("<td><input type=\"checkbox\" name=\"" + id.get(i) + "\" value=\"" + id.get(i) + "\" /></td>");
                                out.println("</tr>");
}
%>
</table>
<br/><input type="submit" name="submit" value="Purchase" />
</form>
</body>
```

Unauthorised.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<!-- Notify the user that he/she is not authorised to access the web site. -->
<html>
        <head>
                <title>Audio Farm Logout</title>
        </head>
        <body>
                <a href="index.jsp">Home</a>
                <a href="Login.jsp">Login</a>
                <a href="Register.jsp">Register</a>
                <h1>Unauthorised Access!</h1>
        </body>
</html>
```

UploadAlbumArt.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.List" %>
<%@ page import="java.util.Iterator" %>
<%@ page import="java.io.File" %>
<%@ page import="org.apache.commons.fileupload.*" %>
<%@ page import="org.apache.commons.fileupload.servlet.ServletFileUpload" %>
<%@ page import="org.apache.commons.fileupload.disk.DiskFileItemFactory" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
        pageContext.forward("Unauthorised.jsp");
}
%>

<%
/* Initialise file related variables. */
Boolean multipart;
FileItemFactory fileFactory;
ServletFileUpload servletUpload;
List files;
Iterator iterator;
File file;
FileItem item;
String fieldName = "";
String fileName = "";
String contentType = "";
Long fileSize;
```

```
/* Check whether the file was submitted using the multipart/form-data HTML form. */
multipart = ServletFileUpload.isMultipartContent(request);

/* Process the data only if it is a file. */
if(multipart){
		/* Store the file temporary in the memory. */
		fileFactory = new DiskFileItemFactory();
		/* Used to handle multiple files. */
		servletUpload = new ServletFileUpload(fileFactory);
		files = servletUpload.parseRequest(request);
		/* Get all form submitted files. */
		iterator = files.iterator();
		item = (FileItem) iterator.next();
		/* Get the submitted file's field name. */
		fieldName = item.getFieldName();
		/* Get the submitted file's original name. */
		fileName = item.getName();
		/* Get the submitted file's MIME type (e.g. JPEG, PNG, GIF). */
		contentType = item.getContentType();
		/* Get the submitted file's size in bytes. */
		fileSize = item.getSize();
		//out.println(fieldName + fileName + contentType);
		/* Get the storage path of the submitted file. */
		file = new File(config.getServletContext().getRealPath("/") + "img/"+ fileName);
		/* Store the file to the server's hard disk drive at the specified default img directory. */
		item.write(file);
		//out.println("<img src=\"img/" + fileName + "\" />");
} else {
		/* No file was selected for upload. */
		out.println("Please select a file first.");
}

%>

<jsp:forward page="Admin.jsp" />
```

User.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<!-- Prevent unauthorised access to the web application. -->
<%
if(session.getAttribute("privileges") == null){
		pageContext.forward("Unauthorised.jsp");
}
%>

<html>
<head>
		<title>Audio Farm User Home</title>
</head>
<body>
<a href="User.jsp">Home</a>
<a href="History.jsp">History</a>
<a href="Logout.jsp">Logout</a>

<!-- Show songs and enable user to buy -->

<%
/* Initialise song related variables. */
ArrayList<String> id = new ArrayList<String>();
ArrayList<String> name = new ArrayList<String>();
ArrayList<String> artist = new ArrayList<String>();
ArrayList<String> album = new ArrayList<String>();
```

```
ArrayList<String> year = new ArrayList<String>();
ArrayList<String> duration = new ArrayList<String>();
ArrayList<String> price = new ArrayList<String>();
ArrayList<String> quantity = new ArrayList<String>();
ArrayList<String> genre = new ArrayList<String>();
ArrayList<String> album_art = new ArrayList<String>();
String basket = "";
String message = "";

try{
	/* Establish a connection to the database with server address, username, and password. */
	String linkToDB = "jdbc:mysql://localhost:3306/audiofarm";
	String username = "adminaudiofarm";
	String password = "b9YMpCFtm9QPxCjv";
	Class.forName("com.mysql.jdbc.Driver").newInstance();
	Connection connection = DriverManager.getConnection(linkToDB,username,password);
	Statement statement = connection.createStatement();

	/* Get all data from the artist table in the database. */
	String query = "SELECT* FROM`artist` LIMIT 0 , 30";
	ResultSet resultSet = statement.executeQuery(query);

	/* Iterate over the results until all song data is assigned. */
	while(resultSet.next()){
		id.add(resultSet.getString(1));
		name.add(resultSet.getString(2));
		artist.add(resultSet.getString(3));
		album.add(resultSet.getString(4));
		year.add(resultSet.getString(5));
		duration.add(resultSet.getString(6));
		price.add(resultSet.getString(7));
		quantity.add(resultSet.getString(8));
		album_art.add(resultSet.getString(9));
		genre.add(resultSet.getString(10));
	}

	/* Select the currently logged in user's surname and gender. */
	String queryGetName = "SELECT id, surname, gender FROM user WHERE id = " + session.getAttribute("userId") + "";
	ResultSet resultGetName = statement.executeQuery(queryGetName);
	/* Iterate over the results to construct a welcome message such as Mr. Gates or Mrs. Clinton. */
	while(resultGetName.next()){
		if(resultGetName.getString(3).startsWith("Male")){
			message = "Mr. " + resultGetName.getString(2);
		} else if(resultGetName.getString(3).startsWith("Female")){
			message = "Mrs. " + resultGetName.getString(2);
		}
	}

}catch(Exception exception){
	/* Notify the user upon failure. */
	out.println("Could not retrieve data from the database.");
}

/* Initialise cart realated variables. */
ArrayList<String> cartItemsArrayList = new ArrayList<String>();
int cartItems;

/* Get all song ids from the session unless empty. */
if(session.getAttribute("songsId") != null){
	cartItemsArrayList = (ArrayList<String>) session.getAttribute("songsId");
	/* Get the number of songs in the session array. */
	cartItems = cartItemsArrayList.size();
} else {
	cartItems = 0;
}

if(cartItems == 0){
	/* If the number of songs in the session array is zero - show an empty basket to the user. */
	basket = "img/basket-empty.png";
} else {
	/* If the number of songs in the session array is zero - show full basket to the user. */
```

```
		basket = "img/basket-full.png";
}

%>

<a href="Cart.jsp" style="text-decoration: none">Cart: <%= cartItems %><img src="<%= basket %>" style="border: none"/></a>
<h1>Welcome <%= message %>!</h1>

<form method="POST" action="Search.jsp">
Search for: <input type="text" name="keyword" />
<select name="searchType">
		<option value="name">Song Name</option>
		<option value="artist">Artist</option>
		<option value="album">Album</option>
</select>
<input type="submit" value="Search" />
</form>

<form method="POST" action="AddSongsToCart.jsp">
<table border=1 color="black">
<th>Name</th><th>Artist</th><th>Album</th><th>Year</th><th>Time</th><th>Price</th><th>Quantity</th><th>Genre</th><th>Albu
m Art</th><th>Buy</th>
<%
/* Iterate over the songs arrays until all data is shown to the user. */
for (int i = 0; i<id.size(); i++){
					out.println("<tr>");
					out.println("<td>" + name.get(i) + "</td>");
					out.println("<td>" + artist.get(i) + "</td>");
					out.println("<td>" + album.get(i) + "</td>");
					out.println("<td>" + year.get(i) + "</td>");
					out.println("<td>" + duration.get(i) + "</td>");
					out.println("<td>" + price.get(i) + "</td>");
					out.println("<td>" + quantity.get(i) + "</td>");
					out.println("<td>" + genre.get(i) + "</td>");
					out.println("<td><img src=\"" + album_art.get(i) + "\" width=\"60\" heigth=\"60\" /></td>");
					out.println("<input type=\"hidden\" name=\"submitted\" value=\"yes\" />");
					out.println("<td><input type=\"checkbox\" name=\"" + id.get(i) + "\" value=\"" + id.get(i) + "\" /></td>");
					out.println("</tr>");
}
%>
</table>
<br/><input type="submit" name="submit" value="Purchase" />
</form>
</body>
</html>
```

Appendix B (Logbook – Java Server Pages, Servlets, Online Shopping)

TABLE OF CONTENTS

ANP 19/03/1

JAVA SERVER PAGES, SERVLET
ONLINE SHOPPING

01 Objectives

- usage and purpose of Servlets.
- usage and purpose of Java Server Pages.
- benefits and drawbacks of JSP and Servlets.
- draft initial online shopping site structure.

02 Servlets

This Java based technology is used to construct dynamic web pages. They are dynamic because unlike HTML pages which contents do not change based on user input, servlets respond to user requests. The resp is achieved using CGI or common gateway interface. The client does not need to have the latest Java Ru installed to receive CGI output. The reason for that is Servlets are java source code passed by the web serv and translated into HTML which is the only langua apart of XML that the web browser understands.

a The user interaction is done using HTML forms These forms could be text, password (the content is obsc select list, radio buttons, and checkboxes. After the form has been filled and the submit button pressed th form action is sent for processing carrying an array of data stored by two server methods - GET, POST. The difference between the two is that that GET meth displays all carried data in the address bar while POST hides it. Hiding sensible data like passwords is always recommended.

b One of the major concerns with CGI programming is how slow it could become serving lots of web server

30 of

clients. The reason for this is because CGI would spawn a new process for each new request which means that the system resources will be overwhelmed easily. Servlets have a solution to this problem. All servlets are run by a single instance of the JVM. Furthermore servlets use lights-weight processes (threads) which have minimal impact on memory and processing power while at the same time sharing commonly used resources from all other threads.

c In order to bypass CGI servlets need a container which runs along side the web server. There are two popular and free solutions, one developed by Apache – Tomcat, and another developed by Oracle Sun – GlassFish.

d Java servlets API requires that all servlets are extended from HttpServlet which can request with ServletRequest library or respond with ServletResponse library. When working with HTML forms such as text fields, text areas, and select options the servlet gets the data using request.getParameter("Surname") for example. It is also important to note that HTML and Javlets should be seperated, they cannot coexist on .html or .java file, even though Java source code can print out HTML tags (e.g. <p>, <a>, <td>, etc).

e Java Servlets can be used to collect data from submit form, process them with platform methods, and store them in a database such as MySQL or Oracle 11g. To do this the servlet needs a DB connector or drives, special prepare statement for SQL queries. Simple SQL queries are SELECT (records in a table), INSERT (values in a table with a reference), UPDATE (values in a table with a reference), CREATE (table with entries), etc.

31 of

The DB driver simplifies the connection to the My
database for example. There are only four required
string argument - address of the DB server, authentica
with valid username, authenticate with valid passwor
select existing database.

f The user session can be tracked with HTML
form hidden values and cookies which store user informa
useful for web site usability. However their usage is not
recommended because they are stored in plain text
and can be easily stolen by malicious users on the netwo
The Servlets solution is to use javax.servlet.http.
HttpSession interface, which provides a way to ident
a user across more than one page request. Since the
data is stored in objects on the server it is a lot
more secure from exploitation. For example:

```
HttpSession httpSession = request.getSession();
httpSession.setAttribute("username", username);
out.println(httpSession.getAttribute("username"))
```

03 Java Server Pages

JSP was introduced to deal with some of
servlets' drawbacks and ultimately replace this obso
technology with a more compact, script-like version
similar to wasted leaders - PHP, Perl, Javascript.
The biggest shortcoming of servlets was that
the source code still had to be compiled .java
files which are an uncomfortable mixture of Java
and HTML output.
In contrast JSP pages can be either pure Java,
pure HTML, or a resonable mixture of the two.
For example to display a variable in the HTML
code of a page, only the following is needed:

```
<%= java.util.Date() %>
```

32 of

Rather than being compiled manually by the developer, JSPs are parsed by the web server, running Tomcat / GlassFish just like HTML.

a The syntax is also slightly different from servlets. The following are independent Java tags:

```
<%= Java expression %>        (Variable)
<% Java statement %>
<%! Java declaration %>       (Method)
<%-- Java comment %>
```

b Since JSPs are modified servlets they too benefit from pre-defined variables, ready to be used.

request	cookies & hostname
response	response type
out	print writer
session	single client session
application	all clients session
config	ServletConfig
pageContext	access to page attributes
page	alternative to this

c The usage of JSP directives is mainly for inclusion of other JSP pages or Java libraries, such as:

```
<%@ import java.util.*; %>
<%@ page import Calculate %>
...
<% Calculate calc = new calc(x,y,z); %>
<% calc.area() %>
...
public class Calculate {... }
```

d Normally the developer would create an instance of a class in a program and use it. With the advent of JSP the developer can work with object part of class and implement it within other pages of JSP. Such functionality of course is similar to

creating an object from the class blueprint.

```
public class Count {
...
  public int getCount(){...}
  public int increment(){...}

    <% page import = "Count" %>
    <jsp:useBean id = "count" scope = "application
    class = "Count">
    </jsp:useBean
    ...
    <% count.increment(); %>
```

It is important to note that the scope can be either

* application - all sessions.
* session - single client session.
* page - single page.
* request - upon client's request.

e JSP uses a special statement to perform SQL queries. This is the prepared statement which prepares variables to be part of SQL INSERT/UPDATE

```
Class.forName("com.mysql.jdbc.Driver");
Connection connection = DriverManager.getConnection
("jdbc:mysql://localhost", "username", "password

psmt = connection.prepareStatement("INSERT
INTO students" +
" firstname, lastname, studentId) VALUES (?,?

psmt.setString(1, firstname);
psmt.setString(2, lastname);
psmt.setString(3, studentId);
```

34 of

f Due to the evolving complexity of web sites, it is recommended to use redirects to reduce the number of pages in total.

<jsp: forward page = "destination" />

04 Benefits and drawbacks of JSP and Servlets

Servlets

Pros	Cons
Security - seperate the site business logic from the web script.	Unpopular - modern web applications rarely use servlets with Tomcat or Glassfish.
Performance - faster and more efficient than CGI.	Convoluted - testing manually or creating automated cases is a difficult task.
Stability - upon an error or exception, only the page will be corrupted rather than the web server crash.	Maintainability - because HTML and Java source code are mixed into one entity, the application is difficult to support, expand, improved.

Java Server Pages

Pros	Cons
Efficiency - scripting nature of JSP and seperation between programming logic and HTML design.	Project Size - JSP is very useful for small dynamic web sites but becomes difficult to maintain if the web site is very complicated.
Compatibility and Availability - tomcat and the Java platform run on Windows, Mac OS X, UNIX, Linux.	Features and Functionality - even though JSP uses java beans for heavy code, servlets offer a lot more to the developer who is building a complicated web application.
Convinience - no need to compile manually the source code after each alteration.	

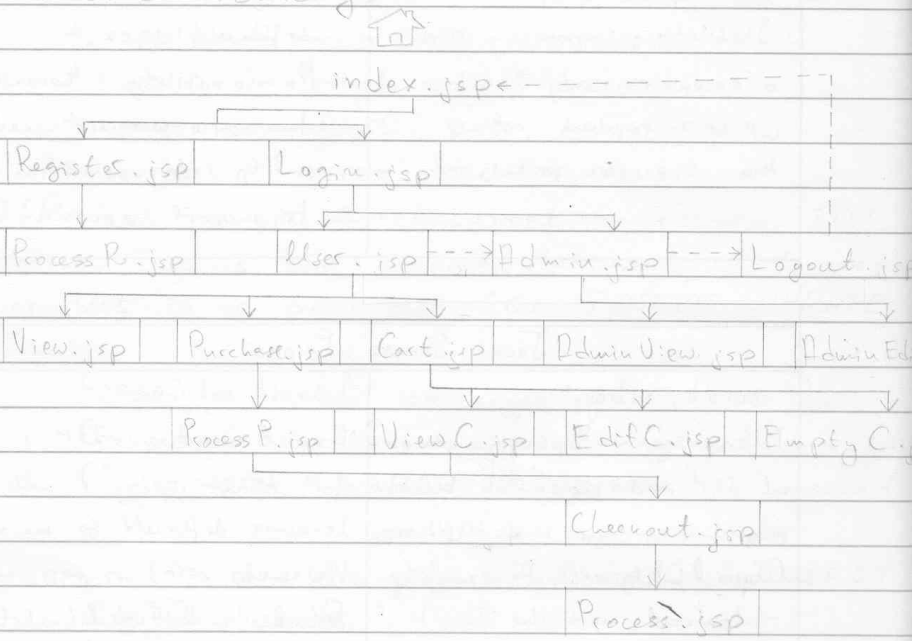
35 of

05 Draft design for the online shopping web app.

a Requirements, aims, and objectives.

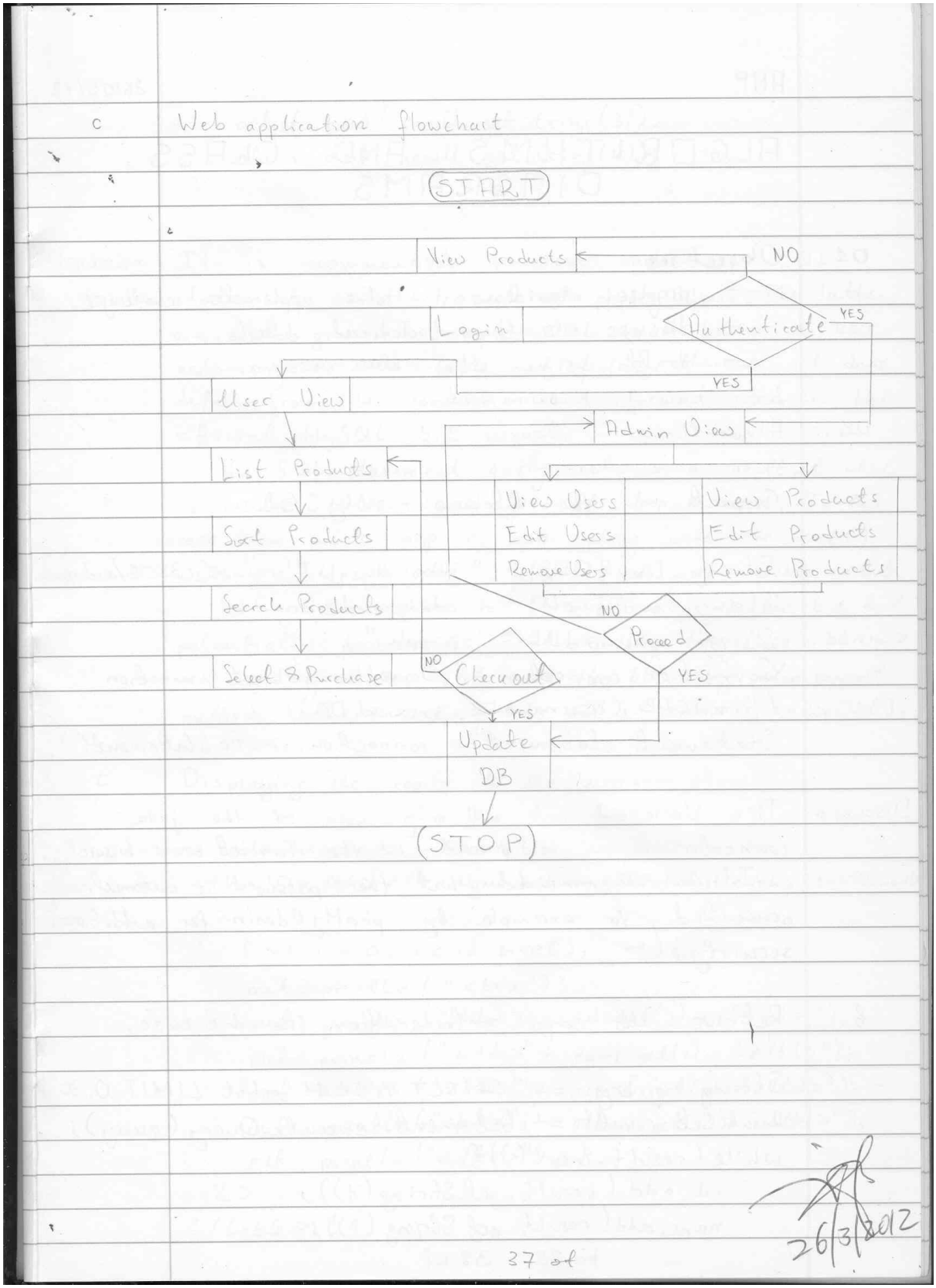
+ usage of JSP or Servlets.
+ products information stored in MySQL database
+ two-levels of login -> user and administrate
+ end-users should be able to:
  * search products
  * browse products
  * select products
  * view shopping cart
  * checkout

b Web Site Hiesarchy



36 of

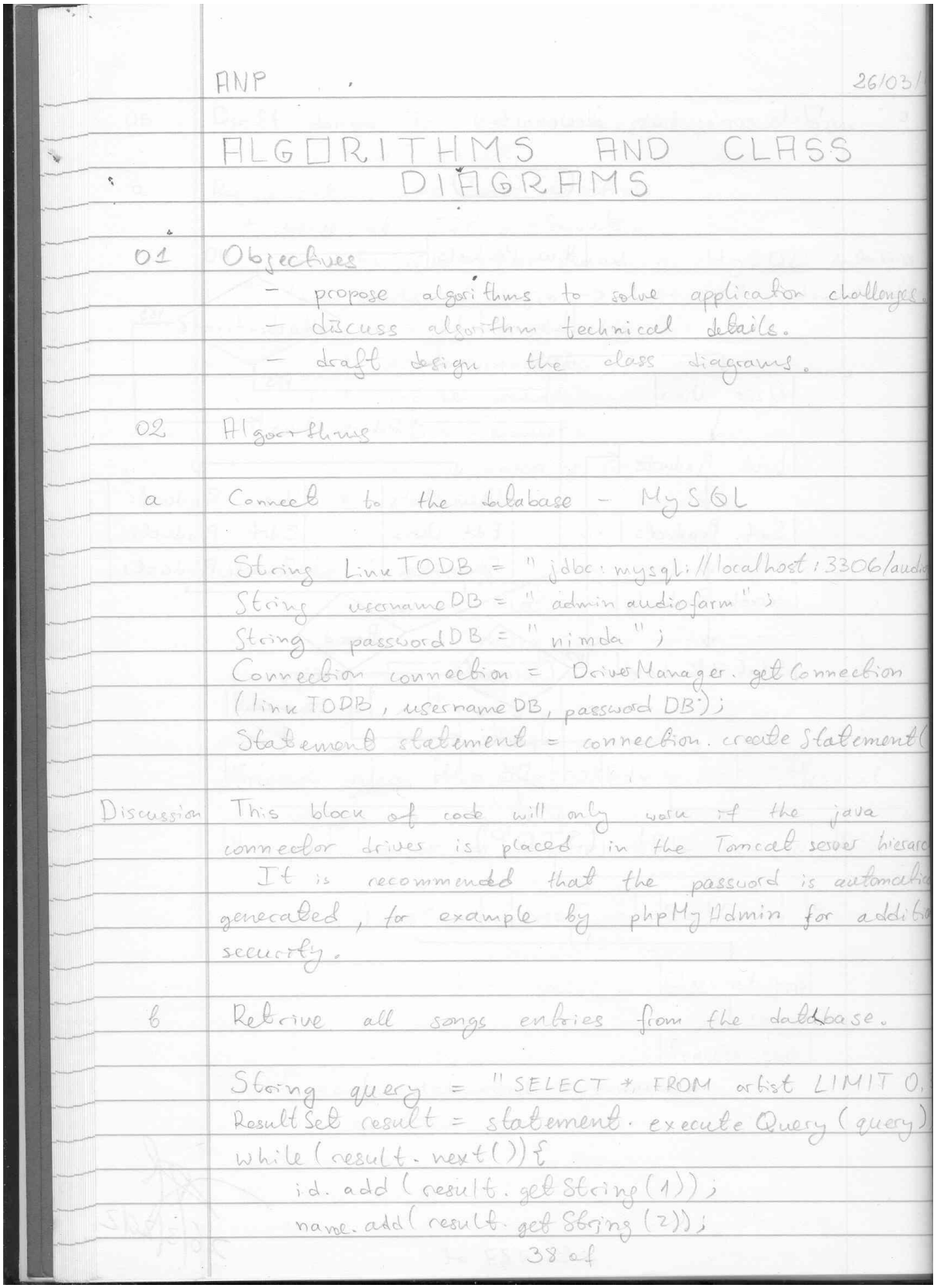
c Web application flowchart



26/3/2012

37 of

Appendix C (Logbook – Algorithms and Class Diagrams)



ANP 26/03/

ALGORITHMS AND CLASS DIAGRAMS

01 Objectives
- propose algorithms to solve application challenges.
- discuss algorithm technical details.
- draft design the class diagrams.

02 Algorithms

a Connect to the database - MySQL

```
String LinkTODB = "jdbc:mysql://localhost:3306/audio
String usernameDB = "admin audiofarm";
String passwordDB = "nimda";
Connection connection = DriverManager.getConnection
(linkTODB, usernameDB, passwordDB);
Statement statement = connection.createStatement(
```

Discussion This block of code will only work if the java connector drives is placed in the Tomcat server hierarc
It is recommended that the password is automatic generated, for example by phpMyAdmin for additio security.

b Retrive all songs entries from the database.

```
String query = "SELECT * FROM artist LIMIT 0,
ResultSet result = statement.executeQuery(query)
while(result.next()){
    id.add(result.getString(1));
    name.add(result.getSting(2));
```

38 of

```
        artist.add(result.getString(3));
        album.add(result.getString(4));
    }
```

Discussion

It is recommended to test each new SQL statement in the console or phpMyAdmin. The latter one auto-generates simple SQL code based on user actions with the table and fields. However it does defer from the convention and format used in the official MySQL 5.5 manual.

An SQL statement query returns a resultset which is useless unless some form of loop is used. In this case the while loop is apt along with the next() method. It will return false if no more results are sent.

Adding the results to String[] arrays is possible but potentially problematic. That is why ArrayList<String> are used which are alot more flexible. They also support methods such as isEmpty(), add(), remove(), indexOf().

c

Displaying the result to the user.

```
<table border=1 colos="black">
<th>ID</th> <th>Name</th> <th>Artist</th> <th>Album</th>
<%
for(int i=0; i<id.size(); i++){
    out.println("<tr>");
    out.println("<td>"+id.get(i)+"</td>");
    out.println("<td>"+name.get(i)+"</td>");
    out.println("<td>"+artist.get(i)+"</td>");
    out.println("<td>"+album.get(i)+"</td>");
    out.println("</tr>");}
%>
</table>
```

Discussion

Displaying the resulting data as plain text / paragraph is not user friendly. Displaying the results as ordered or unordered list was inefficient. That is why the table model was used.

In order to show all songs rather than just one we need an iterator – for loop restricted by the size of id ArrayList.

The get() method retrieves the data much like an ordinary array with an index.

d

Login ⇨ ProcessLogin

```
String formUserName = request.getParameter("username");
formUsername.trim();
String formPassword = request.getParameter("password");
formPassword.trim();
if (formUserName.isEmpty()){
    out.println("<p> Username field was empty.</p>");
} else if (formPassword.isEmpty()){
    out.println("<p> Password field was empty.</p>");
} else {
/* Connect to DB and get all records from user table
...
if (queryUserName.matches(formUserName) &&
    queryPassword.matches(formPassword)) {
    userVerified = true; // USER PRIVILEGE
    if(Integer.parseInt(queryAdmin) == 1){
        adminVerified = true; // ADMIN PRIVILEGE
    }
}
```

Disaission

The form submitted username and password are accessible through request.getParameter and as an argument "name" part of the HTML input form

As a precaution measure they are trimmed from unvoluntary and unnecessary white spaces.

Only if the user name and password are not empty the SQL statement to recove all user table data will be executed.

If the SQL username and password match those submitted from the user, the user privileges will be assigned. Furthermore, if admin field is marked as as one for this user, his/her privileges are elevated to administrator. Even though a user can register to browse, select, search, and buy songs, only the system administrator of the web site and database can mark a user as admin who can manage other users and songs.

e Logout ⇨ Terminate Session

```
String username = (String) session.getAttribute("name");
try {
    session.removeAttribute("userId");
    session.removeAttribute("name");
    session.removeAttribute("privileges");
} catch (NullPointerException exception) {
    message = "Login first, please!";
}
```

Disussion In order to prevent unauthorised access to the web site after the user is no longer active, we need to destroy all user associated session variables. This is done by either removeAttribute() method or clearAll().

If we catch null pointer exception this means that either the session has timed out or someone else is trying to logout who is not an active user.

f Register => ProcessRegistration

```
String forname = request.getParameter("forname");
String surname = request.getParameter("surname");
String username = request.getParameter("username");
...
if (forname.isEmpty()){
    out.println("Field Empty");
} else if (surname.isEmpty()){
    out.println("Field Empty");
} else if (username.isEmpty()){
    out.println("Field Empty");
} else if (password.length() < 6){
    out.println("Password Short");
} else if (!password2.matches(password)){
    out.println("Password Mismatch");
...
} else {
    ...
    String query = "INSERT INTO user
(id, forname, surname, gender, country, city,
email, username, password, secretquestion,
secretanswer, card) VALUES (NULL, '"
    + forname + "', '"
    + surname + "', '"
    + password + "', '"
```

Discussion

The first code blocks gets the form (register) value
The second code block checks whether the values fro
the form are empty which would produce inaccurate
database record. The third code block inserts SQL
statement into the user specific table fields,
the HTML form variables only if non-empty.

The same procedure is used to add song details into the database records.

9 Admin ⇨ Modify Artist Details

```
if (action.startsWith("edit")) {
    String query = "UPDATE artist SET name = '"
    + name + "', artist = '"
    + artist + "', album = '"
    + album + "'
    WHERE id = '" + id + "' ";
    statement.executeUpdate(query);
} else if (action.startsWith("remove")) {
    String delete = "DELETE FROM artist
    WHERE id = '" + id "' ";
    statement.executeUpdate(delete);
}
```

Discussion Depending on the received action type from the HTML form the algorithm either updates artist's details & removes an artist / song identified by its unique id also sent through the HTML form.

The same procedure is used to modify user details.

10 User

The procedure is similar to getting all song details as in the index.jsp. However we get the number of songs resident in the basket.

```
if (session.getAttribute("SongsId") != null) {
cartItems = (ArrayList<String>) session.getAttribute("songsId");
```

```
cartItemsInt = cartItems.size();
} else {
  cartItemsInt = 0;

if (cartItemsInt == 0){
    basket = "basket-empty.png";
} else {
    basket = "basket-full.png";
}
...
<a href="cart.jsp"> Cart: <%= cartItemsInt %>
<img src="<%= basket %>" /></a>
```

Discussion

If the session variable is not empty this means that the user has purchased some songs. If that is true we get the actual number of items / songs with the size() method.

The basket image is either full or empty based on the 'basket' variable which is either zero or a positive integer number.

i

```
Cart

try{
    for(int j = 0; j < songsId.size(); j++){
    String query = "SELECT * FROM artist
    WHERE id = '" + songsId.get(j) + "'";
    result = statement.executeQuery(query);
    while(result.next()){
        id.add(result.getString(1));
    name.add(result.getString(2));
    artist.add(result.getString(3));
    album.add(result.getString(4)); }}
```

44 of 65

```
} catch (NullPointerException exception){
        message = "The basket is empty.";
} catch (Exception exception) {
        message = "DB Error."; }
```

Discussion

This is an efficient loop to retrieve only purchased songs from the database records. The loop iterations are limited to the number of song ids in the session. Then the SQL statement retrieves only those ids from the database correspond to the ones in session. As usual the results are placed in Arraylist. It is important to catch null pointer exception in order to notify the user that the basket is empty.

j Budget

```
String userBudget = "SELECT id, card FROM user
WHERE id = " + session.getAttribute("userId") + "";
result = statement.executQuery (userBudget);
while (result.next()){
        card.add (result.getString(2)); }
```

Discussion

We need to notify the user of his budget. The SQL statement selects only the ID and card according to the session id of the user. Afterwards only the integer value representing the budget is assigned.

k Remove From Cart

```
songsId = (ArrayList<String>) session.getAttribute ("songsId")
for (int i=0; i< songsId.size(); i++){
    id.add (request.getParameter (String.valueOf(i)));
}
```

```
for(int j = 0; j < id.size(); j ++){
    if(songsId.contains(id.get(j))){
        index = songsId.indexOf(id.get(j));
        songsId.remove(index);
    }else{
        out.println("Nothing to remove");
    }
}
session.removeAttribute("songsId");
session.setAttribute("songsId", songsId);
%>
<jsp:forward page="cart.jsp" />
```

Discussion: The loop limited by the number of elements in the session array gets all HTML form elements selected for removal based on their sequencial value of 1, 2, 3 etc. Afterwards a loop limited by the number of values receives checks whether they are contained in the session array. If that is so the element is removed. Next the session array containing the purcheased songs is erased. The same name array is loaded into the session environment with modified (removed) Ids of the songs selected by the user.

- Search

```
String query = "SELECT * FROM artist WHERE
MATCH (" + searchType + ") AGAINST ('" +
keyword + "' IN BOOLEAN MODE)";
result.statementExecute(query);
while(result.next()){
    id.add(result.getString(1));
```

```
name.add(result.getString(2));
artist.add(result.getString(3));
album.add(result.getString(4));
}
} catch (NullPointerException exception) {
    out.println("No records found"); }
```

Discussion: To search for records by user input keyword we need to select all records in the artist table match a particular field (e.g. album) against the inclusive (e.g. +) keyword. Afterwards the results are recorded in song related arrays.

P

```
Checkout
if(songQuantity == 0) {
    warning = "<h1>" + name + " by " + artist + " out of stock";
} else {
    songQuantity -= 1;

budget = Double.parseDouble(card.get(0)) - Double.parseDouble(price);
String query = "UPDATE artist SET quantity ='" + songQuantity +
"' WHERE id ='" + songsId.get(j) + "'";
String query2 = "UPDATE user SET card = " + budget + "
WHERE id = " + session.getAttribute("userId") + "";
```

Discussion: The first condition decreases the song quantity unless 0 which means out of stock. The following SQL statement updates the artist table database records. The budget is also decreased depending on the user's card amount and the price of song. The following SQL statements updates the users table at the id specified by the user id session variable.

03 Class Diagrams

AddArtist
message : String
name : String
artist : String
album : String
year : String
duration : String
price : String
quantity : String
quantityInt : Integer
priceDouble : Double

AddSongsToCart
id : ArrayList<String>
songsId : ArrayList<String>
name : ArrayList<String>
artist : ArrayList<String>
album : ArrayList<String>
year : ArrayList<String>
duration : ArrayList<String>
price : ArrayList<String>
message : String
query : String
result : ResultSet

AddUser
message : String
forname : String
surname : String
gender : String
country : String
city : String
email : String
username : String
password : String
cardid : Integer
admin : Integer
query : String

Admin	
id : ArrayList<String>	country : ArrayList<
name : ArrayList<String>	city : ArrayList<Strin
artist : ArrayList<String>	email : ArrayList<Strin
album : ArrayList<String>	username : ArrayList<Str
year : ArrayList<String>	password : ArrayList<St
duration : ArrayList<String>	cardid : Integer
price : ArrayList<String>	admin : Integer
quantity : ArrayList<String>	query : String
queryUserId : ArrayList<String>	result : ResultSet
forname : ArrayList<String>	
surname : ArrayList<String>	
gender : ArrayList<String>	

48 of 65

Cart
songsId : ArrayList < String>
id : ArrayList < String >
name : Arraylist < String>
artist : ArrayList < String>
album : Arraylist < String >
year : ArrayList < String>
duration : Arraylist < String>
price : Arraylist < String >
quantity : Arraylist < String>
card : Arraylist < String>
message : String
budget : Inteser
query : String
result : ResultSet

Checkout
card : Arraylist < String >
songsId : ArrayList < String>
message : String
name : String
artist : String
price : String
warning : String
songQuantity : Integer
budget : Double
getUserBudget : String
budgetResult : ResultSet
getSongQuantity : String
quantityResult : ResultSet
update : String

index
id : Arraylist < String >
name : Arraylist < String>
artist : ArrayList < String>
album : Arraylist < String>
year : Arraylist < String >
duration : Arraylist < String>
price : Arraylist < String>
quantity : Arraylist < String>
query : String
result : ResultSet

ModifyArtistDetails
message : String
id : String
name : String
artist : String
album : String
year : String
duration : String
price : String
quantity String
priceDouble : Double
quantityInt : Integer
action : String
queryUpdate : String
queryDelete : String

Logout
warning : String
username : String
lastVisit : String
query : String

ModifyUserDetails
message: String
queryUserId: String
forname: String
surname: String
gender: String
country: String
city: String
email: String
username: String
password: String
action: String
card: String
cardInt: Integer
queryUpdate: String
queryDelete: String

ProcessLogin
queryUserId: String
userId: Integer
queryForname: String
querySurname: String
queryUsername: String
queryPassword: String
queryAdmin: String
userVerified: Boolean
adminVerified: Boolean
privileges: Boolean
verifiedForname: String
message: String
Line: String
formUserName: String
formPassword: String

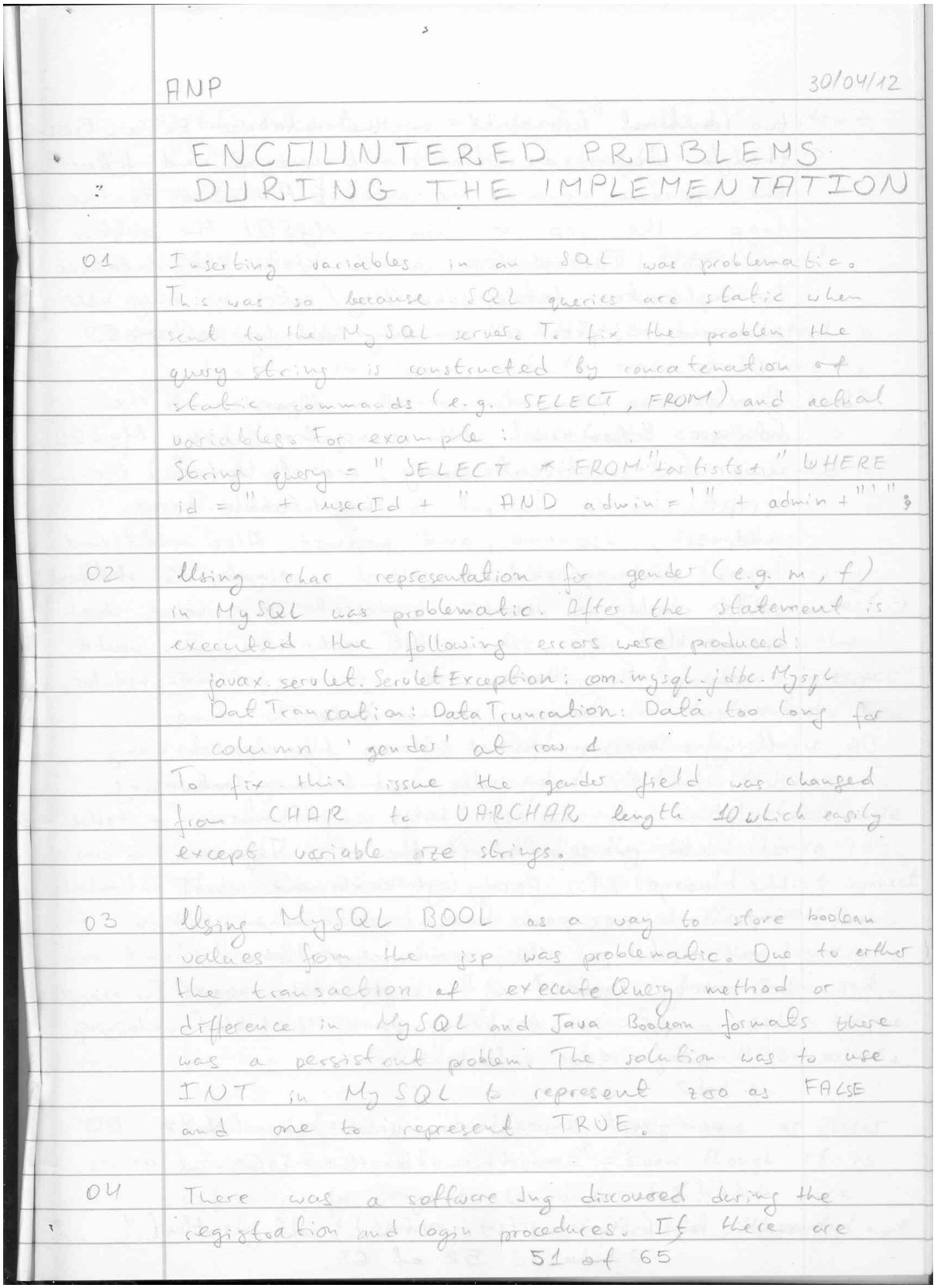
ProcessRegistration
forname: String
surname: String
gender: String
country: String
city: String
email: String
username: String
password: String
password2: String
card: String
cardInt: Integer
queryUpdate: String

RemoveFromCart
index: Integer
songsId: ArrayList<String>
id: ArrayList<String>

Search
id: ArrayList<String>
name: ArrayList<String>
artist: ArrayList<String>
album: ArrayList<String>
year: ArrayList<String>
duration: ArrayList<string>
price: ArrayList<string>
quantity: ArrayList<String>
keyword: String searchType: Strin
query: String result: ResultSet

30/4/2012

Appendix D (Encountered Problems during the Implementation)



ANP 30/04/12

ENCOUNTERED PROBLEMS DURING THE IMPLEMENTATION

01 Inserting variables in an SQL was problematic. This was so because SQL queries are static when sent to the MySQL server. To fix the problem the query string is constructed by concatenation of static commands (e.g. SELECT, FROM) and actual variables. For example:

```
String query = " SELECT * FROM "+artists+" WHERE
id = " + userId + " AND admin = '" + admin + "' ";
```

02 Using char representation for gender (e.g. m, f) in MySQL was problematic. After the statement is executed the following errors were produced:

javax.servlet.ServletException: com.mysql.jdbc.MysqlDataTruncation: Data Truncation: Data too long for column 'gender' at row 1

To fix this issue the gender field was changed from CHAR to VARCHAR length 10 which easily except variable size strings.

03 Using MySQL BOOL as a way to store boolean values from the jsp was problematic. Due to either the transaction of executeQuery method or difference in MySQL and Java Boolean formats there was a persistant problem. The solution was to use INT in MySQL to represent zero as FALSE and one to represent TRUE.

04 There was a software bug discovered during the registration and login procedures. If there are

51 of 65

two identical usernames in the database table. E
though their id numbers are unique and diffe
the username remains the same. Either due to th
loop in the jsp or loop in MySQL the proble
remained. The solution was to check the databa
for duplicates before submitting/retriving any us
information, practice commonly used by web sites.

05 Repetitive use of connection algorithm to the
database. Before each data transaction to the MyS
server (e.g. executeQuery, executeUpdate) the
script has to connect to the database using
address, username, and password. Also additiona
tasks are required to submit a simple SQL state
It would be possible to create java beans wh
are called by the script each time. This would
simplify the source code and make it more modula

06 Using Java arrays of String type posed many
difficulties during loops and even retrieving
information for HTML submit forms or tab
of data presented to the user. The solution w
the usage of ArrayList<String> which automa
expand in size, have useful data manipulation
functions (e.g. add, remove, clear, clone), and
be easily iterated with a for each loop. For exa
this is apt for building an HTML table containi
SQL query items details:

```
<td>
    <input type="text" value="<%=artist%>
    name="artist" maxlength=25 />
</td>
<% for (String item: artist) { out.println("
```

```
<td><input type=\"text\" value=\"" + item +
"\" name=\"artist\" maxlength=\"25\" />
</td>"; } %>

<% for(int i=0; i<id.size(); i++){ ... }
```

07 Providing the user with HTML form nested in a table with song's details was problematic. For example the form would pick-up only the first entry in the <select><option ...> To fix this issue the form was embeded in all table rows.

08 Selecting and getting multiple values (songs) using the HTML checkbox with request.getParameter() method was problematic. This was because the JSP has to know exactly what values to expect and checkbox names cannot be duplicated. For example the script would receive only single, usually the last clicked / selected checkbox. To overcome this issue the names and values were changed to correspond directly to database ids. Thus the processing form script would first connect to the database and then process the HTML form. Alternative solution was to make the names sequential as in "for" loop - from 0 to int. Thus the processing form script can use a simple for loop to iterate over the non-null form values.

09 Deleting a song or user by song name or user surname was not efficient. Even though it is unlikely for songs to be duplicated and more likely for surnames to be duplicated, this did not

during the testing of the application. Howe
as precaution it would be wise to delete
a song / user identified by the unique id.

10 Processing the session array songsId which
contains the ids of the purchased songs w
problematic with null values. For example a
simple check whether the array was empty cra
the script. Thus the following was necessary

```
try {
    if ( songsId.isEmpty() ){
        out.println (" You have not purchased any
    }
} catch ( NullPointerExecption exception ) {
    out.println (" Null Values");
```

11 Searching database entries based upon given
user criteria. To perform full-text searches i
MySQL tables we need the MyISAM engine. T
default engine for MySQL server 5.5 was Inn
which supports transactions. This means that
artist table containing songs details had to
be changed with the following command:

```
ALTER TABLE artist ENGINE = MYISAM
```

Also the fields that were going to be searched
had to be marked with FULL-TEXT as follows

```
ALTER TABLE artist ADD FULLTEXT ( name
albem, artist )
```

54 of 65

To test whether the operation was successful the following query was used to look for "George" in the artist column, which correctly returned "George Michael".

```
SELECT * FROM artist WHERE MATCH(artist)
AGAINST ('George' IN NATURAL LANGUAGE
MODE) LIMIT 0, 30
```

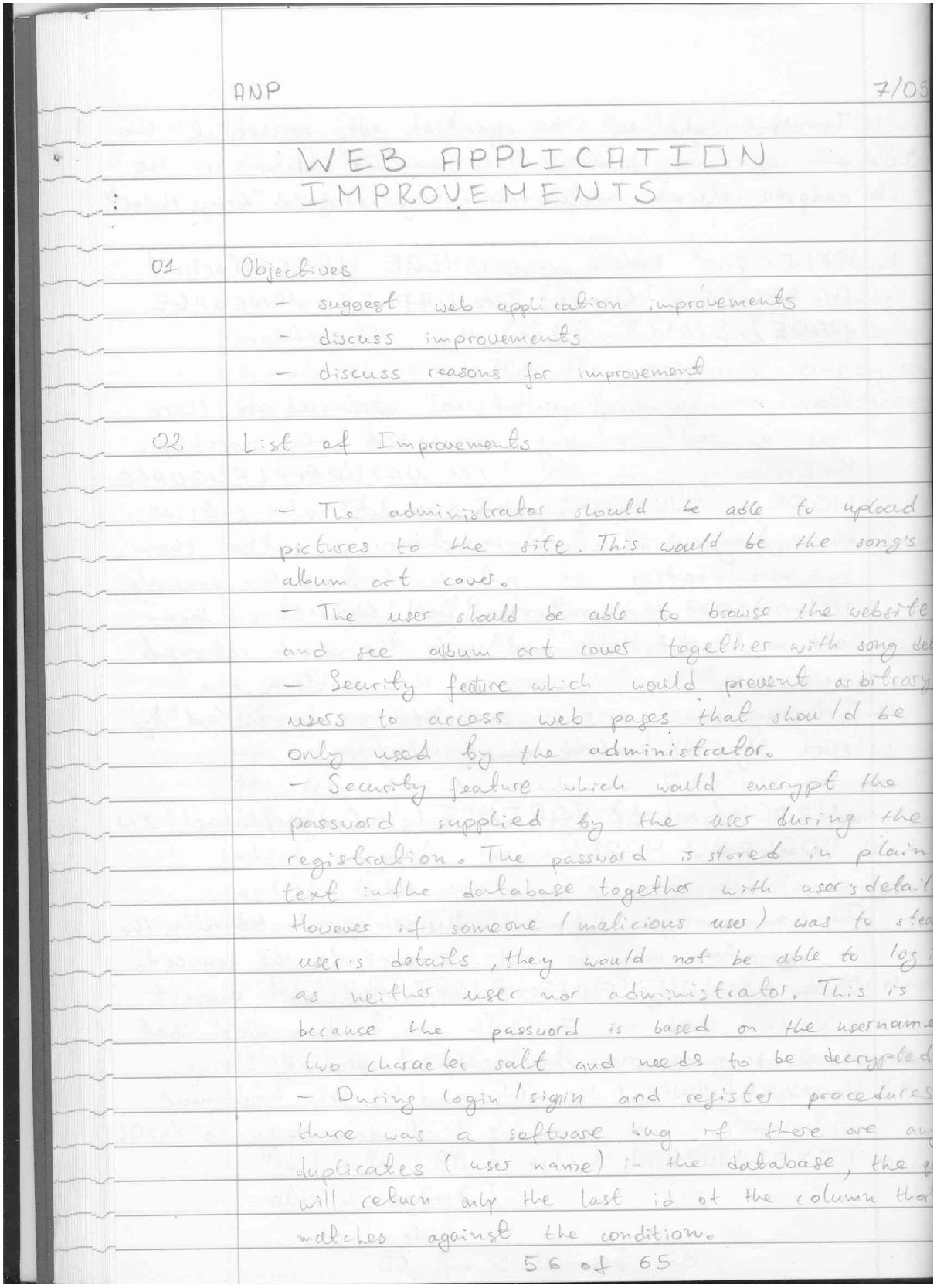
12 During the testing of the application there was a software bug associated with searching. Particularly with the 'IN NATURAL LANGUAGE MODE'. If there are two duplicate entries, the query will return zero results rather than two - correctly or at least one. For example there were two entries for the Duran Duran album - Notorious and yet the search returned zero results. To overcome this problem the following mode was implemented as suggested by the MySQL online documentation:

```
MATCH(artist) AGAINST('+George Michael' IN
BOOLEAN MODE)
```

The + operator means inclusive and optionally the - operator may be used to exclude the keyword. Unfortunately Full-Text search does not support wildcards such as % which can find a word that starts or ends with the specified keyword string. However full-text search benefits from optimal performance when searching text according to MySQL

3/6/2012

Appendix E (Web Application Improvements)



ANP 7/05

WEB APPLICATION IMPROVEMENTS

01 Objectives

- suggest web application improvements
- discuss improvements
- discuss reasons for improvement

02 List of Improvements

- The administrator should be able to upload pictures to the site. This would be the song's album art cover.
- The user should be able to browse the website and see album art cover together with song de
- Security feature which would prevent arbitrary users to access web pages that should be only used by the administrator.
- Security feature which would encrypt the password supplied by the user during the registration. The password is stored in plain text in the database together with user's detail However if someone (malicious user) was to stea user's details, they would not be able to log as neither user nor administrator. This is because the password is based on the username two character salt and needs to be decrypted
- During login / signin and register procedures there was a software bug if there are an duplicates (user name) in the database, the will return only the last id of the column tha matches against the condition.

56 of 65

- The user should be able to remove recently added items to the cart. Such functionality is similar to popular ecommerce web sites such as Amazon, iTunes, eBay etc. This gives the user the opportunity to remove unwanted items rather than logout and start out the procedure again. This would increase user's satisfactions.
- Alter the way the web application performs database SQL queries. Currently before each queryExecute and resultSet the script has to establish the database connection, load the MySQL connector, create new statement etc. It would be recommended to use a java bean or include jsp with DB setting / connection.
- The user should be able to view his or her history as in purchased songs in the past along with a calculated sum of the purchase and date. This idea was influenced by popular web sites such as Amazon and eBay which keep track of the user's activity.
- Based on the previous suggested improvement the user would also be able to view song's suggestions. For example if the songs in the past where jazz, rock, pop genre the web site interface would suggest other songs of the same style. This would yeild more profit for the organisation using this web site.
- As a preventive security measure the web application should be tested for SQL injection. This unfortunately is a problem with many ecommerce web sites where a hacker could view the entire MySQL table records exploiting simple SQL statement substitutions in HTML forms.

57 of 65

03 Improvement Discussion

a Upload picture to the site.

Rather than loading the binary file be
JPEG, GIF, PNG into the database table us
the MySQL format BLOB, it would be better to
use VARCHAR which would simply store the link
to the image located on the local or remote server.
These are two major problems with BLOB. The firs
one is that it makes the database slower during proc
such as SELECT and MATCH/AGAINST. This
because the database is optimised for strings, charter
and integers rather than binary files. The second is
the developer has little control over the image control
for HTML <img>. For example the following code
retrieves an image and then writes out the result

```
String query = "SELECT id, album_art FRO
artist WHERE id = 1";
ResultSet result = executeQuery(query);
response.setContentType("
OutputStream outStream = response.getOutputStre
outStream.write(result.getBytes("album_art");
```

The SQL statement should be constructed as fo
ALTER TABLE artist ADD album_art BLOB NOT

As an alternative the following was used:

```
ALTER TABLE artist ADD album_art VARCHAR(
CHARACTER SET ASCII COLLATE ascii_general_ci
NOT NULL
```

In addition the actual HTML form is also different. W3C recommends the usage of either "application/x-www-form-urlencoded" or "multipart/form-data". Whereas the first is not suitable for large files such as media, the second one is more popular amongst web developers because it can be used to load large amounts of non-ASCII text binary data. For example the following HTML form for uploading album art covers in JPEG, PNG, GIF.

```
<form action="UploadAlbumArt.jsp" method="POST"
  enctype="multipart/form-data">
    <input type="file" name="file" />
    <input type="submit" name="submit"
      value="Upload" />
</form>
```

Afterwards the form can be processed - check multipart content, passe file, iterate over the file to get field name, file name, content type, size in bytes, and finally write the bytes to the selected destination dow drive on the server.

b Users viewing album art while browsing and searching. The address for the image source is stored in the album_art column, which should be retrieved with other song's details.

```
ArrayList<String> album_art = new ArrayList<string>
while(result.next()){
  ...
  album_art.add(result.getString(9));
  out.println("<img src=\""+ album_art.get(i)+"\"/>);
```

c Prevent unauthorised access to sensible web resources such as Admin.jsp
To prevent this from happening we can check whether the session has been created by the user/administrator signing in with correct username and password that match the database records. Placing the following code block before the main execution of JSP code will redirect or rather forward the web browser to a web page notifying the user of unauthoreised a

```
<jsp:forward page="waring.jsp" />
```

In order to make sure the browser is forwar only when the session was not established:

```
if (session.getAttribute("privileges") = null) {
    ...
}
```

The privileges attribute is set by ProcessLogin during successful authentication along with the user id and his/her name.
However there is a problem because the <jsp: forward page="..." /> cannot be embeded with the java code, the following was used:

```
pageContext.forward("Warning.jsp");
```

Publically Accessible Web Pages		
Unauthorised.jsp	Login.jsp	index.
Register.jsp	ProcessLogin.jsp	
ProcessRegistration.jsp	Logout.jsp	

d Encrypt the user supplied password.

To prevent account mismanagement by administrators or other privileged users with access to the web applications database and specifically users details, all passwords are encrypted. This is used by ecommerce websites to also encrypt other sensible information such as credit card details.

MySQL supports several encryption algorithms such as AES (Advanced Encryption Standard), DES (Data Encryption Standard) and hashing algorithms such as SHA1, SHA2, MD5. Hash functions provide similar benefits for creating a secret message that would be infeasible to crack in an ordinary amount of time. In addition they can be computed/calculated faster, support automatic block padding (unlike DES/AES), and are difficult to biject. In order to obfuscate the user supplied password the SHA2 was used which according to MySQL is more secure than MD5 and SHA1. Furthermore MySQL AES/DES require changes to the password column to binary BLOB/string whereas SHA2 uses plain text. According to MySQL there are five options/argument for the functions that measure the strengths in bits of encryption - 224, 256, 384, 512, and 0 (Default)

After some testing and difficulties implementing SHA2 it became obvious that the field had to be changed in order to accommodate the big size of 512 bits key encryption.

ALTER TABLE user CHANGE password TEXT CHARACTER SET ascii NOT NULL

To generate the SHA2 password :

```
SELECT SHA2('"+ formPassword +"', 512)
```

Finally to update the user specific password :

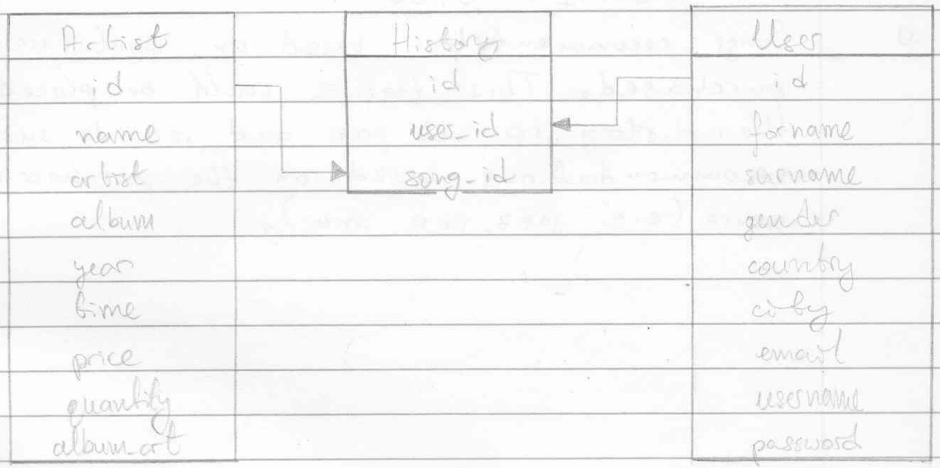
```
UPDATE user SET password = SHA2('" +
formPassword + "', 512) WHERE id = " +
formId + " "
```

e Check for username duplicates during the login procedure. Ecommerce would not allow a user to login with username and password that are the same as someone else's. That is to prevent from accessing the private data of the other user (e.g. credit card, shopping history) and also for efficient web site management

```
String queryForDuplicate = " SELECT username
FROM user ";
ResultSet = resultFromDuplicate = statement.
executeQuery(queryForDuplicate);
while(resultForDuplicate.next()){
    if(resultFromDuplicate.getString(1)
        .startsWith(username)){
            duplicate = true;
            message = " Username already exists
    }
}
if(!duplicate) {
    ...
}
```

62 of 65

f : Implement history web page where the registered user can view past purchases (songs). This common feature amongst ecommerce web sites which could be used as sort of electronic receipt. It is recommended to use a second MySQL table which would store only the user id and purchased songs ids. The last two are also known as foreign id keys to the "history" table, such that:



All three table ids are unique, primary keys, auto increment with each new entry. However the History table user_id and song_id are updated upon checkout.

```
String queryHistory = "INSERT INTO history
(id, user_id, song_id) VALUES (NULL, " +
session.getAttribute("userId") + ", " +
songsId.get(j) + ")";
statement.executeUpdate(queryHistory);
```

h

SQL Injection

For the past couple of years these sort of attacks have become extremely popular amongs web exploiting security hackers. All they require is a web browser and a web page which contains HTML <form> which takes arguments from the user (hacer) and sends them to the JSP server (Tomcat) with POST or GET method which trigger an SQL statement. For example:

```
SELECT * FROM user WHERE name = 'hack or 1=1 --'
```

The 'hack or 1=1 --' is submitted as valid username login query. However regardless of whether hack exists or not the query will return all database entries (1=1) and omit the rest of the SELECT statement because of (--). Another example is the following where a malicious user tries to delete all entries in the database table:

```
SELECT * FROM user WHERE username = 'hack'; DELETE
* FROM user ;
```

To protect against such scenarios the developer should restrict access to the database as such to prevent deletion.

```
CREATE USER 'user'@'%' IDENTIFIED BY '***';
GRANT SELECT, INSERT, UPDATE on audioform;
GRANT ALL PRIVILEGES ON audioform
```

To prevent special character form insert we could use something similar to PHP's pp_escape_string and mysql_real_escape_string. JSP and Servlets use a PREPARED SQL statement to escape special characters.

30/4/2012