

## CNT 4714 – Project Four – Spring 2023

**Title:** “Project Four: Developing A Three-Tier Distributed Web-Based Application”

**Points:** 100 points (bonus problem potentially adds 15 points – see page 14.)

**Due Date:** Sunday April 23, 2023 by 11:59 pm (WebCourses time)

**Objectives:** To incorporate many of the techniques you’ve learned this semester into a distributed three-tier web-based application which uses servlets and JSP technology running on a Tomcat container/server to access and maintain a persistent MySQL database using JDBC.

**Description:** In this assignment you will utilize a suppliers/parts/jobs/shipments database (creation/population script available on Webcourses under Project 4) as the back-end database. Front-end access to this database by end users will occur through a single page displayed in the client’s web browser. The schema of the backend database consists of four tables with the following schemas for each table:

suppliers (snum, sname, status, city) //information about suppliers

parts (pnum, pname, color, weight, city) //information about parts

jobs (jnum, jname, numworkers, city) //information about jobs

shipments (snum, pnum, jnum, quantity) //suppliers ship parts to jobs in specific quantities

The database will enforce referential integrity via foreign key constraints. The primary key for the shipments table is a composite key consisting of three foreign keys (the primary keys in the suppliers, parts, and jobs tables). Referential integrity means that a shipment record cannot exist unless it links back (via referential integrity) to existing entities on all foreign key values. Thus, a shipment record cannot exist unless the referenced snum, pnum, and jnum already exist in their respective tables.

The first-tier (user-level front-end) of your web-application will consist of an HTML landing page which is used to authenticate end users. The authentication of users is handled via a servlet in the webapp that validates the user entered credentials with those in a credentials file (a text file) maintained on the server-side. The credentials file contains an unknown number of lines where each line contains a username and password separated by a comma (csv file format). Validation consists of matching both the user entered name and password against the values in the credentials file. If the entered credentials do not match an credentials in the credentials file, the user will be denied access to the system. If a match is found, the authenticated the user will be automatically redirected to one of three different JSP pages. One which handles root-level user clients and one which handles non-root-level clients, that allow the users to enter arbitrary SQL commands into a window (i.e. a form) and submit them to a server application for processing. The third JSP page will be consist of dedicated data entry forms for entering new records into the four tables in the database.

The front-ends of all three user applications will utilize JSP technology. The front-ends for the root-level and client-level users, will provide the user a simple form in which they will enter a SQL command (any DML, DDL, or DCL command could theoretically be entered by the user, however we will restrict to queries, insert, update, replace, and delete commands). These two front-ends will provide only three buttons for the user, an “Execute Command” button that will cause the execution of the SQL command currently in the input window, a “Reset Form” button that simply clears any content currently in the form input area, and a “Clear Results” button that will erase the currently displayed data (user optional). The third front-end will be utilized only by naïve data-entry users by filling in a form. The data-entry users will not enter

SQL commands to accomplish their tasks. Rather, their web-application will use the `PreparedStatement` interface and extract the parameters from their forms and issue the SQL command in the background. The data-entry level front end will have two buttons on each form, one for entering the data and one for clearing data and results.

The front-ends will run on any web-based browser that you would like to use. The applications will connect to the backend database via properties files dependent on which front-end page is utilized. This connection must be handled using properties read from a properties file. You will have three different properties files, one for the root-level users, one for the client-level users (all the same as project 3 except for the different database), and one for the data entry-level users. The data-entry level user account is a new user that will need to be created, just as you created the client user for Project 3.

The second-tier servlets, are in charge of handling the SQL command interface for the users. The root-level user app (and the data entry level app – see below), will also implement the server-side business/application logic. This logic will increment by 5, the status of a supplier anytime that supplier is involved in the insertion/update of a shipment record in which the quantity is greater than or equal to 100. Note that any update of quantity  $\geq 100$  will affect any supplier involved in a shipment with a quantity  $\geq 100$ . The example screen shots illustrate this case. An insert of a shipment tuple (S5, P6, J7, 400) will cause the status of every supplier who has a shipment with a quantity of 100 or greater to be increased by 5. In other words, even if a supplier's shipment is not directly affected by the update, their status will be affected if they have any shipment with quantity  $\geq 100$ . (**See page xxs for a bonus problem that implements a modified version of this business rule.**) The business logic of the second tier will reside in the servlets on the Tomcat web-application server (server-side application). This means that the business logic is not to be implemented in the DBMS via a trigger.

The client-level servlet will handle the SQL command interface, just as the root-level servlet does, however, due to the restrictions on the client-level privileges, no business-logic will be implemented in this application.

The data entry-level servlet will provide the user four templates (forms) for the each of the tables in the project 4 database. The correct updating command will be executed by mid-tier level servlets issuing whichever updating commands are appropriate based on which form was submitted by the data-entry user. The updating commands are executed by extracting the parameters from the form and issuing a prepared statement update to the correct database table. You may want to refer to the JDBC notes from Module 3 to refresh your memory of how the `PreparedStatement()` interface differs from the normal `Statement()` interface.

The third-tier (back-end) is the persistent MySQL database described above and is under control of the MySQL DBMS server. You will create and maintain this database via the creation/population script. See the important note below concerning when/how to re-run this script for your final submission.

## References:

Notes: Lecture Notes for MySQL installation and use. Documentation for MySQL available at: <http://www.mysql.com>. More information on JDBC can be found at: <http://www.oracle.com/technetwork/java/javase/jdbc/index.html>. More information on Tomcat can be found at <http://tomcat.apache.org>. Lecture Notes for Servlets. Lecture Notes for JSPs.

## **Restrictions:**

Your source file shall begin with comments containing the following information:

```
/* Name:  
   Course: CNT 4714 – Spring 2023 – Project Four  
   Assignment title: A Three-Tier Distributed Web-Based Application  
   Date: April 23, 2023
```

\*/

**Special Note: Due to end of semester time constraints this will be a hard deadline.**

**Input Specification:** The suppliers/part/jobs/shipments database (named `project4`) that is created/populated by the script `project4dbscript.sql`, is the back-end to this application. All other input comes from the front-end user submitted to the application server based servlet entered as either queries or updates to this database. There are three sets of commands that you are to execute against this database included in the `project4rootcommands.sql`, `project4clientcommands.sql`, and `project4dataentrycommands.sql` available on WebCourses under Project 4. As with Project 3, your client-level user will have only `select` privileges on the `project4` database. The data entry-level user will have only `select`, `insert` and `update` privileges on the `project4` database. Also, as with Project 3, your front-end cannot execute the entire script at one time. You'll need to execute the commands in this script one at a time in your application (copy and paste!). You can run the scripts in the MySQL Workbench if you'd like to compare/see the result sets for each user command. Your three different user accounts should be named: `root`, `client`, and `dataentryuser`.

**Output Specification:** All output is generated by the servlets and should appear in the user's browser as a text/html presented to the user. All MySQL-side errors should be caught and reported to the user via the interface. **IMPORTANT:** Be sure to re-run the `project4dbscript.sql` database creation/population script before you begin creating your screen shots for submission. By doing so you will ensure that the database is in its initial state so that all update operations will produce the values we are expecting to see in your result outputs. Then, just as you did with Project 3, run all commands in sequence from the `project4rootcommands.sql` script file (total of 20 different commands), followed immediately by all commands in sequence from the `project4clientcommands.sql` script file (total of 4 different commands), followed immediately by all commands in sequence from the `project4dataentrycommands.sql` script file (total of 8 different commands).

## **Deliverables:**

- (1) You should submit your entire Project4 webapp folder from Tomcat for this program. If you submit the entire folder, then all of the files necessary to execute your web application will be included with the directory structure intact. Submit this via WebCourses no later than **11:59pm Sunday April 23, 2023**.
- (2) The following 20 screen shots from the `project4rootcommands.sql` script file must be submitted as part of the deliverables for this project. (You can include the screenshots in the top-level of your webapps folder if you'd like, just be sure to include a note that you've done so.)
  - a. Command 1
  - b. Command 2A
  - c. Command 2B
  - d. Command 2C
  - e. Command 3A
  - f. Command 3B
  - g. Command 3C

- h. Command 3D
- i. Command 3E
- j. Command 4
- k. Command 5A
- l. Command 5B
- m. Command 5C
- n. Command 5D
- o. Command 5E
- p. Command 6
- q. Command 7
- r. Command 8
- s. Command 9
- t. Command 10

- (3) The following 4 screenshots from the `project4clientcommands.sql` script file must be submitted as part of the deliverables for this project. (You can include the screenshots in the top-level of your webapps folder if you'd like, just be sure to include a note that you've done so.)
- a. Command 1
  - b. Command 2
  - c. Command 3
  - d. Command 4
- (4) The following 8 screenshots from the `project4dataentrycommands.sql` script file must be submitted as part of the deliverables for this project. (You can include the screenshots in the top-level of your webapps folder if you'd like, just be sure to include a note that you've done so.)
- a. Command 1
  - b. Command 2
  - c. Command 3
  - d. Command 4
  - e. Command 5
  - f. Command 6
  - g. Command 7
  - h. Command 8
- (5) One final screenshot, taken from the MySQL Workbench using a root-user connection, executing the command `select * from suppliers;` This will show the final status of all suppliers after the execution of all commands in the three user-level scripts.

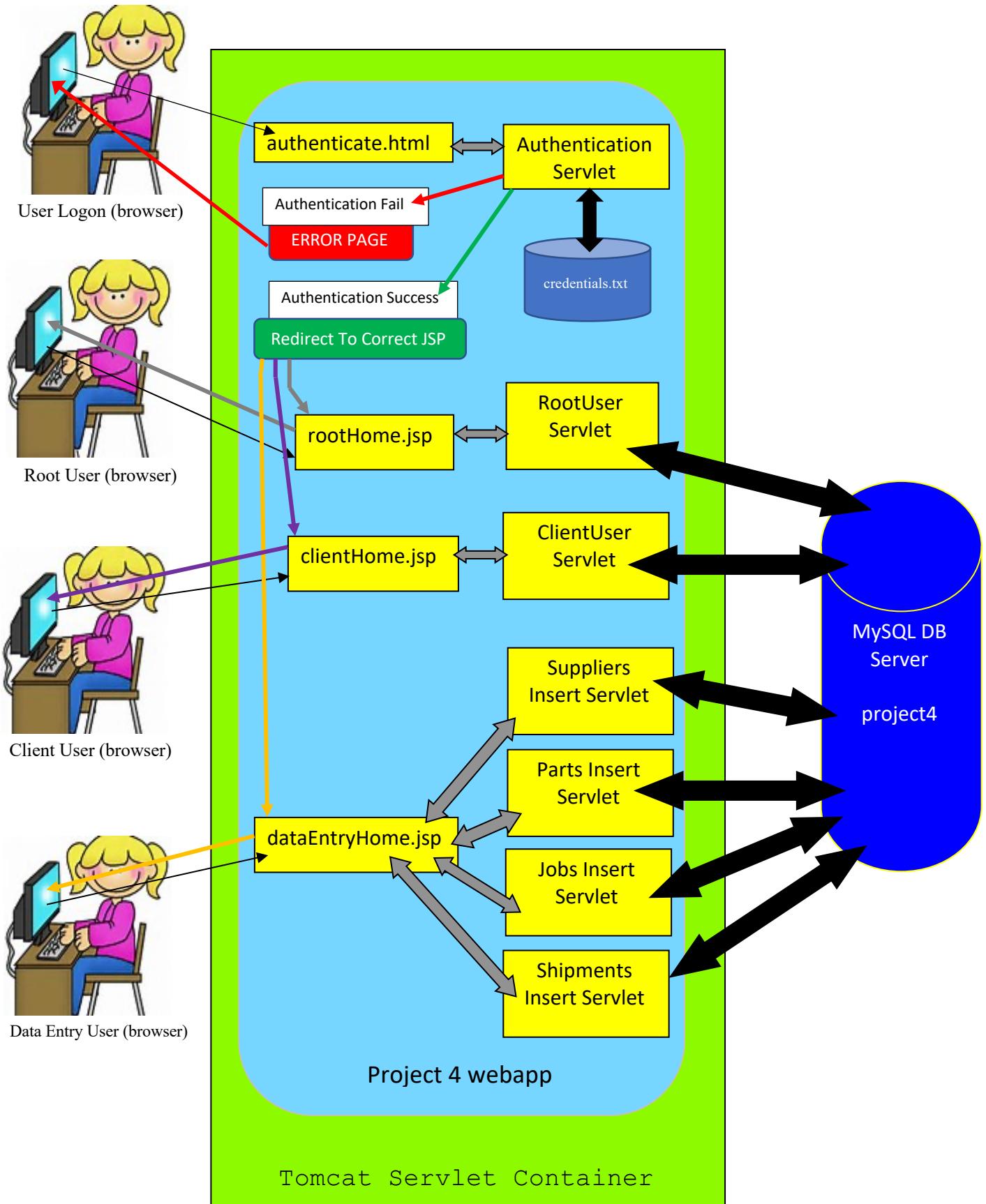
#### **Additional Information:**

Be very careful when setting up the directory structures required for the web applications running under your server (Tomcat 10.1.5 or later – note that Tomcat 10.1.7 was released on March 3, 2023 and I have upgraded my Tomcat to this version – you should really do the same). See the course notes on servlets for the exact directory structure that must be developed. Be sure that your development IDE and the JVM running under Tomcat are of the same vintage.

Attend/watch Q&A sessions for more information and project details. Additional videos for select parts of this project will also be made available.

**Important:** Please name your webapp: **Project 4** Let the TAs know if you are doing the bonus problem by attaching a note to your WebCourses submission.

## Schematic Overview of Project Components:



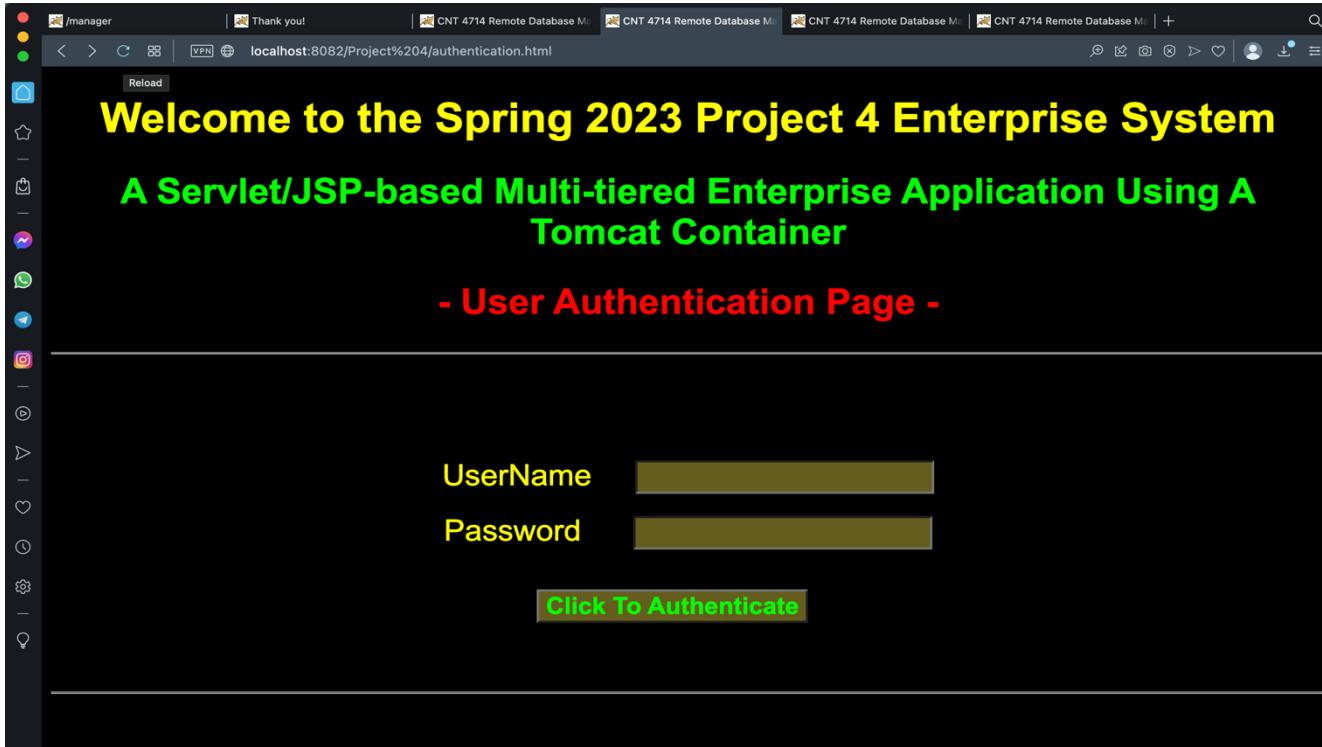
## Suggested project development approach

1. Install Tomcat and run some of the examples from the notes to ensure that Tomcat is installed and configured properly before beginning any steps on the project itself. Do not attempt to start the actual project unless you know that (1) Tomcat is properly configured and running, (2) that your IDE Java and your Tomcat JVM are of the same vintage, (3) you have successfully run at least one known working example.
2. **IMPORTANT: Keep back-up copies of all files within your webapp in a location NOT inside Tomcat. Maintain these backup copies religiously.**
3. Develop the authentication HTML front-end and it's associated error page.
4. Develop front-end .html files rootHome.html, clientHome.html and dataentryHome.html. These will be later converted to .jsp files (see 7 below). This can be done in any editing environment of your choice. Do not specify a specific action for your form submission at this point (use a null string for the action). Will demonstrate this in Q&A sessions.
5. Construct basic Project4 webapp framework inside Tomcat webapp folder. Deploy files from step 1 above and test/refine in browser of your choice.
6. Construct initial web.xml file in Project4/WEB-INF. Additional refinement may be needed later.
7. Create properties files for the root-level users, client-level users, and data entry-level users. These will be placed in the lib folder of the Project4 web app (i.e. Project4/WEB-INF/lib)..
8. Begin development of the servlets. Basic operation of the root-level servlet and the client-level servlet are the same, with only a small difference (more later). So develop the client-level servlet first and copy and paste with modifications for the root-level servlet later. As we will discuss in the Q&A sessions, the initial servlet (for testing) should do nothing more than simply return "Hi".
9. Load servlet test files into Tomcat Project4 webapp in correct location and perform initial integration testing of complete package.
10. Further develop all servlet code to complete the basic functionality of the servlets. This includes modification of the front-end interfaces to become .jsp files so that all results are returned to a single page via a targeted location and not require either a complete browser page refresh or the user to employ the browser "back" button. These techniques will be explained later in the JSP notes and also Q&A sessions.
11. Add business logic to the root-level servlet – develop non-bonus version first.
12. Optional: implement the bonus-version of the business logic.
13. Add business logic to the data entry-level servlet. This will apply only to the servlet handling inserts to the shipments table.
14. Recreate the project4 database.
15. Run through the three user-level command scripts and generate screenshots from your application running these commands. Take final screenshot from the Workbench.
16. You are done! Congratulations! You have developed a three-tier distributed web-based application running in a Tomcat container utilizing a MySQL backend database server.

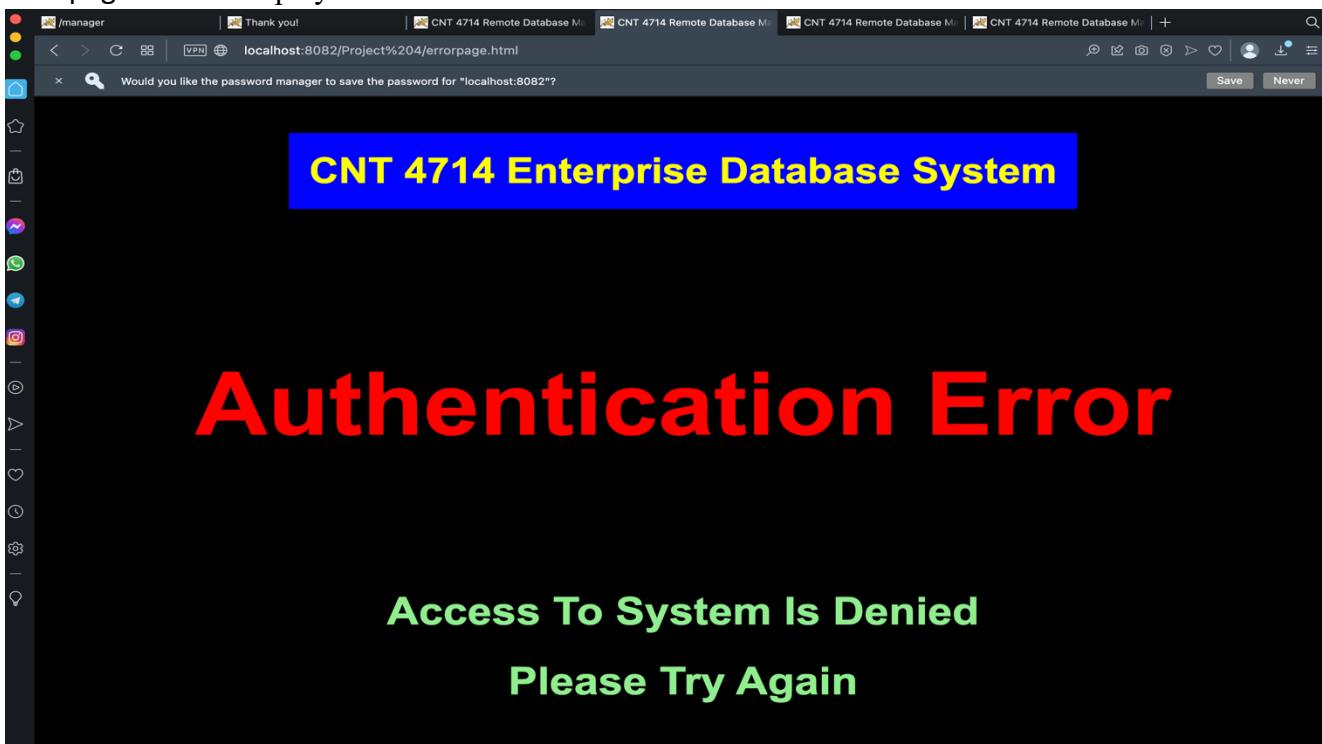
Some screen shots illustrating the application.

## Authentication Page – The Main Initial Landing Page

To run this webapp all users begin on this landing page (this is authentication.html).



If the user enters incorrect login credentials that do not match any found in the credentials.txt file. The errorpage.html is displayed.



If the user's login credentials are validated (they match with an entry in the credentials.txt file), the user will be automatically redirected to a landing page which corresponds to the webapp which they are authorized to access. These landing pages, for each category of user are shown below.

Root-level user interface (a JSP page - initial configuration shown):

The screenshot shows a web browser window with multiple tabs open, all displaying the same page: CNT 4714 Remote Database M... (localhost:8082/Project%204/rootHome.jsp). The main content of the page is:

**Welcome to the Spring 2023 Project 4 Enterprise Database System**

**A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container**

You are connected to the Project 4 Enterprise System database as a **root-level** user.  
Please enter any valid SQL query or update command in the box below.

select \* from suppliers

User input area.

Three control buttons

Execute Command    Reset Form    Clear Results

All execution results will appear below this line.

Database Results:

All results are returned in this area.

Annotations with red arrows point to specific elements:

- An arrow points from the text "User input area." to the text "select \* from suppliers".
- Three arrows point from the text "Three control buttons" to the three buttons: "Execute Command", "Reset Form", and "Clear Results".
- An arrow points from the text "All results are returned in this area." to the placeholder text "Database Results:".

Client-level user interface (a JSP page - initial configuration shown):

(Note that except for the color scheme in the headers and the fact that a client-level user is specified, this page is essentially identical to that of the root user interface.)

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost:8082/Project%204/ClientUserApp
- Title Bar:** Welcome to the Spring 2023 Project 4 Enterprise Database System
- Content Area:**
  - Welcome Message:** Welcome to the Spring 2023 Project 4 Enterprise Database System
  - Description:** A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container
  - Text:** You are connected to the Project 4 Enterprise System database as a **client-level** user.  
Please enter any valid SQL query or update command in the box below.
  - Input Box:** A large blue rectangular area representing the input field for SQL queries.
  - Buttons:** Execute Command, Reset Form, Clear Results
  - Text Below Input:** All execution results will appear below this line.
  - Section Header:** Database Results:

The data entry-level user interface (a JSP page – initial configuration shown).

Welcome to the Spring 2023 Project 4 Enterprise Database System  
Data Entry Application

You are connected to the Project 4 Enterprise System database as a **data-entry-level** user.  
Enter the data values in a form below to add a new record to the corresponding database table.

Suppliers Record Insert

snum	sname	status	city
[ ]	[ ]	[ ]	[ ]

**Enter Supplier Record Into Database** **Clear Data and Results**

Parts Record Insert

pnum	pname	color	weight	city
[ ]	[ ]	[ ]	[ ]	[ ]

**Enter Part Record Into Database** **Clear Data and Results**

Jobs Record Insert

jnum	jname	numworkers	city
[ ]	[ ]	[ ]	[ ]

**Enter Job Record Into Database** **Clear Data and Results**

Shipments Record Insert

snum	pnum	jnum	quantity
[ ]	[ ]	[ ]	[ ]

**Enter Shipment Record Into Database** **Clear Data and Results**

Database Results:

## Root-level User Examples

The following several screenshots illustrate operations from the root-level user interface.

After entering an SQL command, the user simply clicks the “Execute Command” button and the SQL command in the form is executed:

The screenshot shows a web browser window with multiple tabs open, all titled "CNT 4714 Remote Database Ma". The main content area displays a welcome message for a Spring 2023 Project 4 Enterprise Database System. It informs the user they are connected as a root-level user and prompts them to enter a valid SQL query or update command. A text input field contains the query "select \* from jobs". Below the input field are three buttons: "Execute Command" (highlighted in green), "Reset Form" (in red), and "Clear Results" (in yellow). A note below the buttons states that execution results will appear below the line. The results section is labeled "Database Results:" and contains a table with the following data:

jnum	jname	numworkers	city
J1	Operation DB	45	Berlin
J13	Night Strike	350	Paris
J2	Really Big Job	500	Melbourne
J22	Project On-Time	200	London
J3	Small Job	100	Chicago
J4	New Job	50	Berlin
J5	My Job	1	Orlando
J6	A New Job	14	Milan

User makes a mistake entering an SQL command:

The screenshot shows a web browser window with multiple tabs open, all titled "CNT 4714 Remote Database Ma". The active tab is "localhost:8082/Project%204/RootUserApp". The page title is "Welcome to the Spring 2023 Project 4 Enterprise Database System". A green annotation highlights the text "A Servlet/JSP-based Multi-tiered Enterprise A". A red box highlights an error message: "There is no column named something in the parts table." A red arrow points from this message to the SQL query input field. The input field contains the text "select something from parts". Another red box highlights the error message "Error executing the SQL statement: Unknown column 'something' in 'field list'" returned by MySQL. A red arrow points from this message to the error message above it. The bottom of the page has buttons for "Execute Command", "Reset Form", and "Clear Results".

Welcome to the Spring 2023 Project 4 Enterprise Database System

A Servlet/JSP-based Multi-tiered Enterprise A

There is no column named something in the parts table.

You are connected to the Project 4 Enterprise System database as a root-level user.  
Please enter any valid SQL query or update command in the box below.

select something from parts

Error message is returned from MySQL indicating the problem with the

All execution results will appear below this line.

Database Results:

Error executing the SQL statement:  
Unknown column 'something' in 'field list'

Execute Command Reset Form Clear Results

Inserts and updates may cause changes to the supplier status field (business logic is triggered) as shown below:

Current state of the suppliers table (i.e., select \* from suppliers):

Note the current status of supplier number S5. (Also note status of S1, S12, S17, S21, S22, S3, S44, and S6.)

Database Results:			
snum	sname	status	city
S1	Michael Schumacher	1	Berlin
S10	David Coulthard	2	London
S11	Bernard Hinault	7	Paris
S12	Eddy Merckx	1	Brussels
S13	Candice Swanepoel	3	Cape Town
S14	Adriana Lima	4	Sao Paulo
S15	Jennifer Lawrence	6	Owensboro
S16	Fernando Alonso	4	Madrid
S17	Rubens Barrichello	3	Sao Paulo
S18	Tom Boonen	2	Brussels
S19	Johan Messeeuw	1	Eekloo
S2	Juan Pablo Montoya	4	Interlagos
S20	Danilo Rossi	2	Milan
S21	Lizzie Armistead	1	Hempstead
S22	Jan Ullrich	5	Bonn
S3	Dietrich Thurau	1	Berlin
S32	Bernd Schnieder	2	Berlin
S33	Rolf Aldag	3	Berlin
S4	Mark Webber	5	Melbourne
S44	Beryl Burton	4	London
S5	Jenson Button	4	London
S56	Marianne Vos	8	Zandvoort
S6	Nicola Gianniberti	2	Milan
S7	Christian Albers	3	Orlando
S8	Giancarlo Fisichella	3	Milan
S9	Kimi Rikkonen	2	Helsinki

Results from running the user command select \* from suppliers in the input area.

Results from running the query “select \* from suppliers” – to be used to illustrate an update operation explained on pages 14-15. Notice that the supplier S5’s status is currently 4.

Root user issues the following insert command:

The screenshot shows a web browser window with multiple tabs open, all titled "CNT 4714 Remote Database M...". The active tab is "localhost:8082/Project%204/RootUserApp". The page title is "Welcome to the Spring 2023 Project 4 Enterprise Database System" and subtitle is "A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container". A message at the top states: "You are connected to the Project 4 Enterprise System database as a **root-level** user. Please enter any valid SQL query or update command in the box below." Below this is a text input field containing the SQL command: "insert into shipments values ('S5','P6','J4',400)". At the bottom of the page are three buttons: "Execute Command" (green), "Reset Form" (red), and "Clear Results" (yellow). A note below the buttons says: "All execution results will appear below this line." The "Database Results" section is currently empty.

Alert message when an update to the quantity field in the shipments table has caused an update of a supplier's status in the supplier table. Note that the application will use this alert message any time the business logic is tested even if it did not trigger any updates. This means that this message would appear with different values even if no rows are updated (more examples below).

The screenshot shows a web browser window with multiple tabs open, all titled "CNT 4714 Remote Database M...". The active tab is "localhost:8082/Project%204/RootUserApp". The page title is "Welcome to the Spring 2023 Project 4 Enterprise Database System" and subtitle is "A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container". A message at the top states: "You are connected to the Project 4 Enterprise System database as a **root-level** user. Please enter any valid SQL query or update command in the box below." Below this is a text input field containing the SQL command: "insert into shipments values ('S5','P6','J4',400)". At the bottom of the page are three buttons: "Execute Command" (green), "Reset Form" (red), and "Clear Results" (yellow). A note below the buttons says: "All execution results will appear below this line." The "Database Results" section contains a green box with the following text: "The statement executed successfully.  
1 row(s) affected.  
**Business Logic Detected! - Updating Supplier Status**  
**Business Logic updated 9 supplier status marks.**".

After executing update command (the previous insert), the user re-runs `select * from suppliers`. Note that S5's status has been increased by 5, but so too has S1, S12, S17, S21, S22, S3, S44, and S6. Allowing the previous insert command to affect only supplier S5's status is handled by the bonus version of this project (see below). We will discuss the business logic in much greater detail in the Q&A sessions.

Database Results:			
snum	sname	status	city
S1	Michael Schumacher	6	Berlin
S10	David Coulthard	2	London
S11	Bernard Hinault	7	Paris
S12	Eddy Merckx	6	Brussels
S13	Candice Swanepoel	3	Cape Town
S14	Adriana Lima	4	Sao Paulo
S15	Jennifer Lawrence	6	Owensboro
S16	Fernando Alonso	4	Madrid
S17	Rubens Barrichello	8	Sao Paulo
S18	Tom Boonen	2	Brussels
S19	Johan Messeuw	1	Eekloo
S2	Juan Pablo Montoya	4	Interlagos
S20	Danilo Rossi	2	Milan
S21	Lizzie Armistead	6	Hempstead
S22	Jan Ullrich	10	Bonn
S3	Dietrich Thurau	6	Berlin
S32	Bernd Schnieder	2	Berlin
S33	Rolf Aldag	3	Berlin
S4	Mark Webber	5	Melbourne
S44	Beryl Burton	9	London
S5	Jenson Button	9	London
S56	Marianne Vos	8	Zandvoort
S6	Nicola Gianniberti	7	Milan
S7	Christian Albers	3	Orlando
S8	Giancarlo Fisichella	3	Milan
S9	Kimi Rikkonen	2	Helsinki

Notice on page 13 (in the original suppliers table) that supplier S5 had a status of 4. After this update, the business logic has increased supplier S5's status by 5, so it is now 9.

Notice too, that suppliers S1, S12, S17, S21, S22, S3, S44, and S6) also had their status increased by 5, since they already recorded with a shipment in which the quantity was  $\geq 100$  when the insert command was issued, even though the issued command did not affect them directly.. See bonus problem below for a "fix".

Example of an update command that does not trigger the business logic.

Welcome to the Spring 2023 Project 4 Enterprise Database System

A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container

You are connected to the Project 4 Enterprise System database as a **root-level** user.  
Please enter any valid SQL query or update command in the box below.

```
update jobs set jname = "Tough Job" where jnum = "J1"
```

**Execute Command**   **Reset Form**   **Clear Results**

All execution results will appear below this line.

**Database Results:**

The statement executed successfully. A total of 1 row(s) were updated.  
Business Logic Not Triggered!

Example of an update command that triggers the business logic but results in no changes to any supplier's status.

The screenshot shows a web browser window with multiple tabs open, all titled 'CNT 4714 Remote Database M...'. The active tab displays the URL 'localhost:8082/Project%204/RootUserApp'. The page title is 'Welcome to the Spring 2023 Project 4 Enterprise Database System' and the subtitle is 'A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container'. On the left, there is a vertical sidebar with various icons. The main content area contains a text input box with the SQL command 'update shipments set quantity = 10'. Below the input box are three buttons: 'Execute Command' (highlighted in green), 'Reset Form', and 'Clear Results'. A note below the input box states, 'All execution results will appear below this line.' The 'Database Results' section shows the output: 'The statement executed successfully. 56 row(s) affected.', 'Business Logic Detected! - Updating Supplier Status', and 'Business Logic updated 0 supplier status marks.' A red box highlights the note about the unsafe update, and a red arrow points from this note to the 'Execute Command' button.

You are connected to the Project 4 Enterprise System database as a **root-level** user.  
Please enter any valid SQL query or update command in the box below.

```
update shipments set quantity = 10
```

**Execute Command**   **Reset Form**   **Clear Results**

All execution results will appear below this line.

**Database Results:**

```
The statement executed successfully.  
56 row(s) affected.  
Business Logic Detected! - Updating Supplier Status  
Business Logic updated 0 supplier status marks.
```

Note that this update is an “unsafe” update since there is no limiting clause and every row in the table will be updated. In this case all 56 rows in the table will now have a quantity of 10.

## Client-level User Examples

A client-level user issues a command for which they have privileges (note that this is the same result as the one shown for a root-level user on page 11).

The screenshot shows a web browser window with the URL `localhost:8082/Project%204/ClientUserApp`. The title bar reads "Welcome to the Spring 2023 Project 4 Enterprise Database System". The main content area displays a message: "You are connected to the Project 4 Enterprise System database as a **client-level** user. Please enter any valid SQL query or update command in the box below." Below this is a text input field containing the SQL query `select * from jobs`. At the bottom of the input field are three buttons: "Execute Command" (green), "Reset Form" (red), and "Clear Results" (yellow). A note below the buttons states: "All execution results will appear below this line." Below this note, the heading "Database Results:" is followed by a table with the following data:

jnum	jname	numworkers	city
J1	Operation DB	45	Berlin
J13	Night Strike	350	Paris
J2	Really Big Job	500	Melbourne
J22	Project On-Time	200	London
J3	Small Job	100	Chicago
J4	New Job	50	Berlin
J5	My Job	1	Orlando
J6	A New Job	14	Milan

A client-level user issues a command for which they do not have privilege to execute.

The screenshot shows a web browser window with multiple tabs open, all titled "CNT 4714 Remote Database Ma". The active tab is "localhost:8082/Project%204/ClientUserApp". The page content is as follows:

**Welcome to the Spring 2023 Project 4 Enterprise Database System**  
**A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container**

You are connected to the Project 4 Enterprise System database as a **client-level** user.  
Please enter any valid SQL query or update command in the box below.

```
insert into shipments values ("S5","P6","J4",400)
```

**Execute Command**   **Reset Form**   **Clear Results**

All execution results will appear below this line.

**Database Results:**

**Error executing the SQL statement:**  
INSERT command denied to user 'client'@'localhost' for table 'shipments'

## Data Entry-level User Examples

The data entry-level user interface provides a template (form) for the user to enter data values for new suppliers, parts, jobs, or shipments records. Only one form can be submitted at a time. Note: the database does not specify any default values for any attributes in any of the table in the project4 database. Therefore, the user must supply all values to be used by the PreparedStatement interface in each form.

Data entry user enters a new suppliers table record.

Welcome to the Spring 2023 Project 4 Enterprise Database System  
Data Entry Application

You are connected to the Project 4 Enterprise System database as a **data-entry-level** user.  
Enter the data values in a form below to add a new record to the corresponding database table.

Suppliers Record Insert

snum	sname	status	city
S90	Anna-Frieda	35	Stockholm

Enter Supplier Record Into Database    Clear Data and Results

Parts Record Insert

pnum	pname	color	weight	city

Enter Part Record Into Database    Clear Data and Results

Jobs Record Insert

jnum	jname	numworkers	city

Enter Job Record Into Database    Clear Data and Results

Shipments Record Insert

snum	pnum	jnum	quantity

Enter Shipment Record Into Database    Clear Data and Results

Database Results:

New suppliers record: (S90, Anna-Frieda, 35, Stockholm) - successfully entered into database.

In this example, the quantity value will trigger the business logic and update supplier S5's status by 5.

Welcome to the Spring 2023 Project 4 Enterprise Database System

Data Entry Application

You are connected to the Project 4 Enterprise System database as a **data-entry-level** user.  
Enter the data values in a form below to add a new record to the corresponding database table.

Suppliers Record Insert

snum	sname	status	city
[ ]	[ ]	[ ]	[ ]

Enter Supplier Record Into Database    Clear Data and Results

Parts Record Insert

pnum	pname	color	weight	city
[ ]	[ ]	[ ]	[ ]	[ ]

For this example. If the status of supplier "S90" before issuing the insert to the shipments table was 35, then it will be 40 after this command is issued.

Jobs Record In

Query status before insert

Database Results:

status
35

Shipment Rec

Query status after insert

Database Results:

status
40

Enter Shipment Record Into Database    Clear Data and Results

Database Results:

New shipments record: (S90, P4, J5, 400) - successfully entered into database. Business logic triggered.

In this example, the business logic is not triggered as the quantity value is less than 100.

Welcome to the Spring 2023 Project 4 Enterprise Database System  
Data Entry Application

You are connected to the Project 4 Enterprise System database as a **data-entry-level** user.  
Enter the data values in a form below to add a new record to the corresponding database table.

Suppliers Record Insert

snum	sname	status	city
<input type="text" value="S90"/>	<input type="text" value="P3"/>	<input type="text" value="J4"/>	<input type="text" value="30"/>

**Enter Supplier Record Into Database** **Clear Data and Results**

Parts Record Insert

pnum	pname	color	weight	city
<input type="text" value="S90"/>	<input type="text" value="P3"/>	<input type="text" value="J4"/>	<input type="text" value="30"/>	<input type="text" value=""/>

**Enter Part Record Into Database** **Clear Data and Results**

Jobs Record Insert

jnum	jname	numworkers	city
<input type="text" value="S90"/>	<input type="text" value="P3"/>	<input type="text" value="J4"/>	<input type="text" value="30"/>

**Enter Job Record Into Database** **Clear Data and Results**

Shipments Record Insert

snum	pnum	jnum	quantity
<input type="text" value="S90"/>	<input type="text" value="P3"/>	<input type="text" value="J4"/>	<input type="text" value="30"/>

**Enter Shipment Record Into Database** **Clear Data and Results**

Database Results:

New shipments record: (S90, P3, J4, 30) - successfully entered into database. Business logic not triggered.

In this example, the attempt to insert the record violates referential integrity and is not allowed.

Welcome to the Spring 2023 Project 4 Enterprise Database System  
Data Entry Application

You are connected to the Project 4 Enterprise System database as a **data-entry-level** user.  
Enter the data values in a form below to add a new record to the corresponding database table.

Suppliers Record Insert

snum	sname	status	city
<input type="text" value="S5"/>	<input type="text" value="P6"/>	<input type="text" value="J7"/>	<input type="text" value="133"/>

**Enter Supplier Record Into Database** **Clear Data and Results**

Parts Record Insert

pnum	pname	color	weight	city
<input type="text" value="S5"/>	<input type="text" value="P6"/>	<input type="text" value="J7"/>	<input type="text" value="133"/>	<input type="text" value="CA"/>

**Enter Part Record Into Database** **Clear Data and Results**

Jobs Record Insert

jnum	jname	numworkers	city
<input type="text" value="S5"/>	<input type="text" value="P6"/>	<input type="text" value="J7"/>	<input type="text" value="CA"/>

**Enter Job Record Into Database** **Clear Data and Results**

Shipments Record Insert

snum	pnum	jnum	quantity
<input type="text" value="S5"/>	<input type="text" value="P6"/>	<input type="text" value="J7"/>	<input type="text" value="133"/>

**Enter Shipment Record Into Database** **Clear Data and Results**

Database Results:

Error executing the SQL statement:  
Cannot add or update a child row: a foreign key constraint fails

## BONUS PROBLEM: 15 points

Instead of allowing any update/insert of a quantity  $\geq 100$  to affect **any** supplier with a shipment involving a quantity  $\geq 100$ , adjust the business logic portion of your application so that an insert/update of a quantity greater than 100, causes a change to the status of **only** those suppliers directly affected by the update. For example, using the case shown above, when inserting the row (S5, P6, J4, 400) into the shipments table, only the status of supplier S5 should be increased by 5 (see screen shot below). However, an update such as: UPDATE shipments SET quantity = quantity + 50 WHERE pnum = "P3", would increase by 5 the status of every supplier who ships part P3 in a quantity  $\geq 100$  after the update has been issued.

NOTE: If you elect to do the bonus problem, submit only this version of your application. Do not also submit the non-bonus problem version. Let the TAs know if you've elected to do the bonus problem or not.

I will provide many hints for the bonus problem during the Q&A sessions for this project.

With the correct business logic (the bonus version) in place, issue the original insert command as above (on page 14), we now get the correct effect for our update command. See next page.

Welcome to the Spring 2023 Project 4 Enterprise Database System

A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container

You are connected to the Project 4 Enterprise System database as a **root-level** user.  
Please enter any valid SQL query or update command in the box below.

insert into shipments values ("S5","P6","J4",400)

**Execute Command**   **Reset Form**   **Clear Results**

All execution results will appear below this line.

**Database Results:**

The statement executed successfully.  
1 row(s) affected.

Business Logic Detected! - Updating Supplier Status

Business Logic updated 1 supplier status marks.

Notice the difference in the output message this time compared to the one on page 14 that was using the “normal” logic.

## Database Results:

snum	sname	status	city
S1	Michael Schumacher	1	Berlin
S10	David Coulthard	2	London
S11	Bernard Hinault	7	Paris
S12	Eddy Merckx	1	Brussels
S13	Candice Swanepoel	3	Cape Town
S14	Adriana Lima	4	Sao Paulo
S15	Jennifer Lawrence	6	Owensboro
S16	Fernando Alonso	4	Madrid
S17	Rubens Barrichello	3	Sao Paulo
S18	Tom Boonen	2	Brussels
S19	Johan Messeuw	1	Eekloo
S2	Juan Pablo Montoya	4	Interlagos
S20	Danilo Rossi	2	Milan
S21	Lizzie Armistead	1	Hempstead
S22	Jan Ullrich	5	Bonn
S3	Dietrich Thurau	1	Berlin
S32	Bernd Schnieder	2	Berlin
S33	Rolf Aldag	3	Berlin
S4	Mark Webber	5	Melbourne
S44	Beryl Burton	4	London
S5	Jenson Button	9	London
S56	Marianne Vos	8	Zandvoort
S6	Nicola Gianniberti	2	Milan
S7	Christian Albers	3	Orlando
S8	Giancarlo Fisichella	3	Milan
S9	Kimi Rikonen	2	Helsinki

Notice that this time, with the improved business logic that only the supplier directly affected by the insert has had their status updated, all other supplier status values remain unchanged. Compare with table on page 15.

No changes to S1, S12, S17, S21, S22, S3, S44, or S6 this time.

Only supplier S5 had a change of status due to the insertion of the row (S5, P6, J7, 400) as they were the only supplier affected by this update.