

# CNT 4714 – Project Four – Fall 2024

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

**Points:** 100 points (bonus problem potentially adds 15 points – see page 25.)

**Due Date:** Sunday December 1, 2024 by 11:59 pm (WebCourses time)

**FINAL DUE DATE – NO GRACE PERIOD FOR THIS PROJECT!!! – NO EXTENSIONS POSSIBLE.**

**Objectives:** To incorporate many of the techniques you’ve learned so far 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, named: project4, (the 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 MySQL DBMS 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 another database named credentialsDB maintained on the MySQL DB server. (The credentialsDB is created and populated using the credentialsDBscript.sql). This servlet is running as a system-level application with root user privileges (this user will be known as systemapp, more later). The credentialsDB contains a single table named usercredentials of usernames and their associated passwords. Validation consists of matching both the user entered name and password against the values stored in the usercredentials table. If the user entered credentials do not match an entry (row) in the usercredentials table, 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. Finally, a third JSP

page will be reserved for accountant-level users, who do not enter SQL commands directly, but rather execute remote stored procedures (RPCs) that reside on the database server.

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, for accountant-level users, will consist of a listing of various “reports” that can be run based on their selection. The accountant-level user will simply select an option from a list of possible “reports” and the effect of this will be to execute a RPC on the database server and return the results of the report to the accountant-level front-end page. The account-level application will use the `CallableStatement` interface. More details on this interface and remote procedure calls will be covered in detail in the Q&A sessions as well as in additional documentation that supports this project.

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 four different properties files, one for the root-level users, one for the client-level users, one for the accountant-level users, and one for the system-level application which handles user validation (see below). All user accounts (except for the root user) will be created via the script file named: `ClientCreationPermissionsScript.sql`. This script will only need to be run one time after the database scripts (`project4DB.sql` and `credentialsDBscript.sql`) are executed.

The second-tier servlets, are in charge of handling the SQL command interface for the users. The root-level user app, 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, J4, 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 XX 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. Many additional details on the business logic will be covered in the Q&A sessions.

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. This is because client-level users do not have updating privileges on the `project4` database and our business logic can only be potentially triggered by updating operations.

The accountant-level servlet will provide the user with a set of options, where each option is a “report” (an underlying SQL query) that will return the report details to the accountant-level front-end page. Each element in the list of reports that can be generated references a stored procedure that resides on the MySQL server under the database project4. Executing remote stored procedures requires using the CallableStatement interface. Execution of the project4DB.sql script will create the stored procedures needed for this project.

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 project4DB.sql. 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:**

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

```
/* Name:  
   Course: CNT 4714 – Fall 2024 – Project Four  
   Assignment title: A Three-Tier Distributed Web-Based Application  
   Date: December 1, 2024  
*/
```

**Special Note: Due to end of semester time constraints the, December 1, 2024 by 11:59pm due date/time, will be a hard deadline with no extensions possible. NO GRACE PERIOD FOR THIS PROJECT.**

2. The servlet for the accountant-level users, must use the callableStatement interface. Again, no exceptions here.
3. **Do not** implement the business logic as a database trigger.

#### **Input Specification:**

The credentialsDB is created by the credentialsDBscript.sql script. 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 scripts that you will execute against this database included in the:

1. project4rootcommands.sql,

2. project4clientcommands.sql, and
3. project4accountantcommands.txt

All are available on WebCourses under Project 4. As with Project 3 your client-level user (only have client for this project) will have only select privileges on the project4 database. The accountant-level user will have only execute privileges on the project4 database. Also, as with Project 3 your front-end cannot execute an entire script at one time. You'll need to execute the commands in each script one at a time in your application (copy and paste!). Your three different end user accounts are named: root, client, and theaccountant.

Project initiation sequence (must do these steps in the shown order).

1. Run the project4DBscript.sql script to create the project4 database
2. Run the credentialsDBscript.sql to create the credentialsDB.
3. Run the ClientCreationPermissionsScript.sql to set the user accounts and permissions.
4. Create the properties files for all three end-user types (root, client, and theaccountant). These users access only the project4 database.
5. Create the properties file for the systemapp level user. This is the internal app used to validate user credentials, so this properties file contains root-level properties to the credentialsDB only.

### **Output Specification:**

All output is generated by the servlets and should appear in the user's browser as text/html output 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 project4accountantcommands.sql script file (total of 5 different commands). **Note that all screenshots for this project are taken from the perspective of your application, not the MySQL Workbench.**

### **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 December 1, 2024. Note that there is no grace period for this project.**
- (2) The following 21 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.)

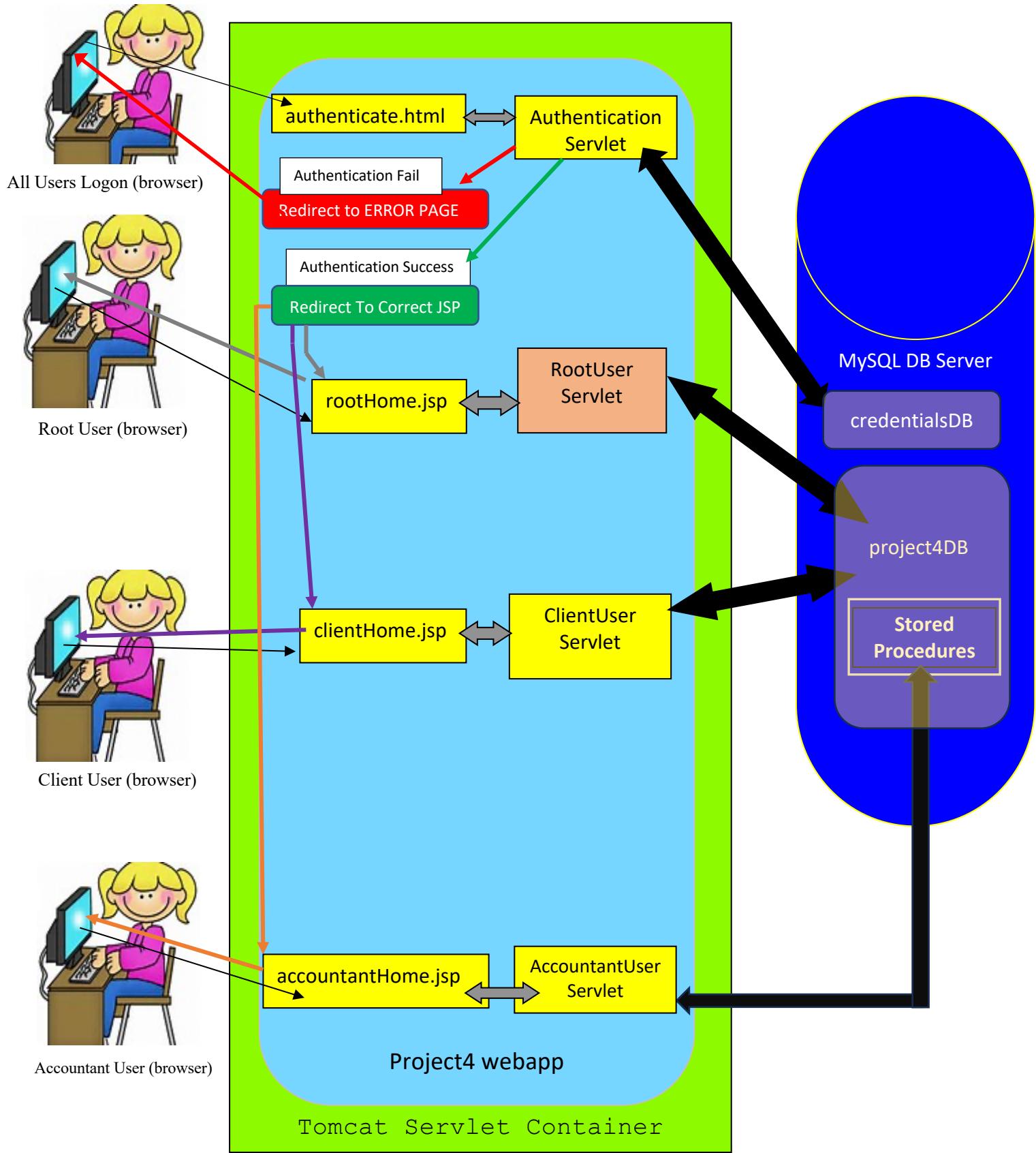
- a. Command 1
  - b. Command 2A
  - c. Command 2B
  - d. Command 2C
  - e. Command 3A
  - f. Command 3B
  - g. Command 3C
  - h. Command 3D
  - i. Command 3E
  - j. Command 4
  - k. Command 5A
  - l. Command 5B
  - m. Command 5C
  - n. Command 5D
  - o. Command 5E
  - p. Command 6
  - q. Command 7
  - r. Command 8
  - s. Command 9
  - t. Command 10
  - u. Command 11
- (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 5 screenshots from the `project4accountantcommands.txt` 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

**Additional Information:**

Be very careful when setting up the directory structures required for the web applications running under your server (Tomcat 10.1.25 or later). 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.

**Important:** Please name your webapp: **Project4**. Let the TAs know if you are doing the bonus problem by attaching a note to your WebCourses submission.

### Schematic Overview of Project Components:



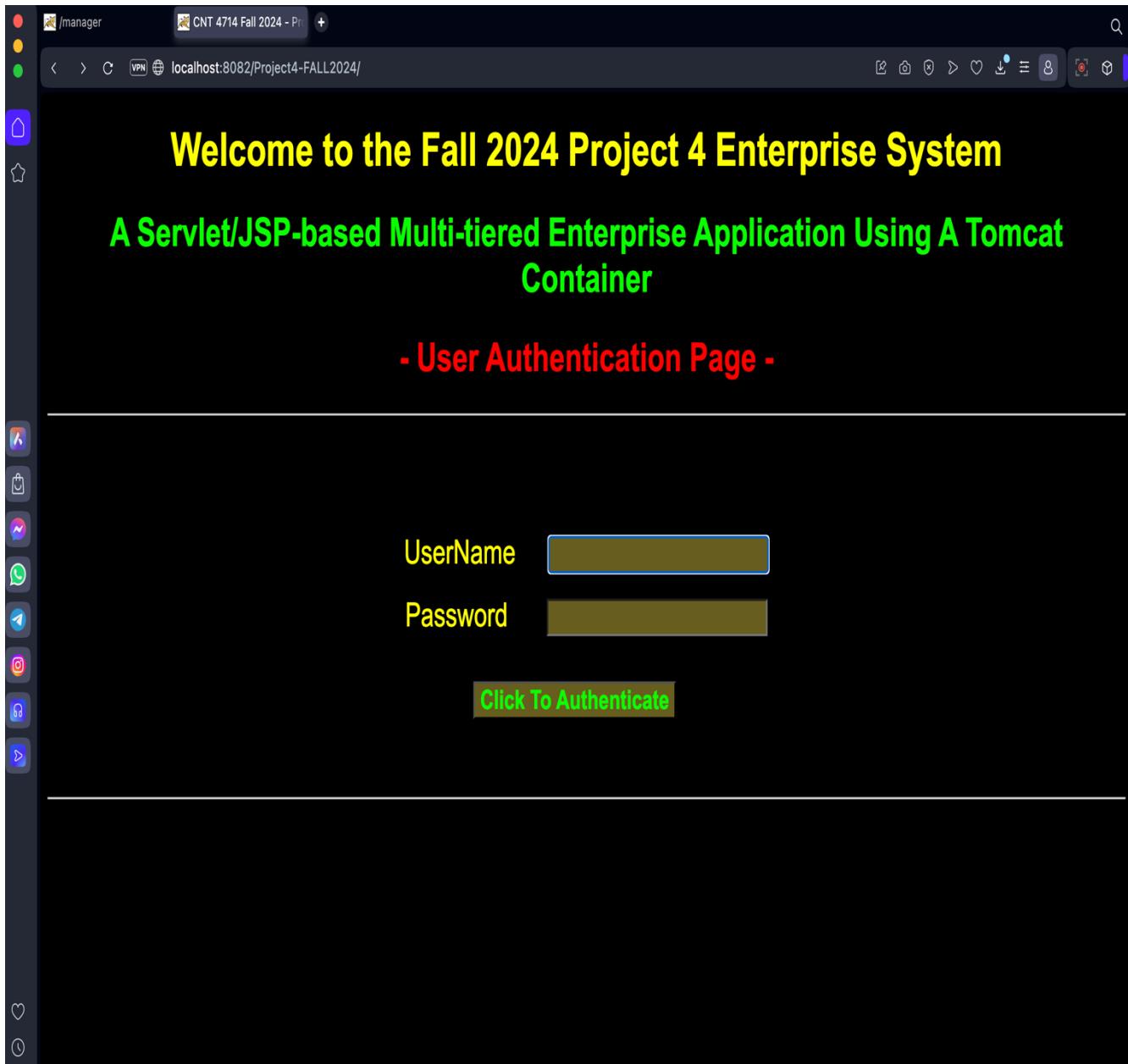
## **Suggested project development approach**

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. Under the Input Specification section above (pages 3-4), run the database creation/population scripts (there are two of them), then run the user creation/permission script.
4. Develop the authentication HTML front-end and the associated error page.
5. Develop front-end .html files rootHome.html, clientHome.html, and accountantHome.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). I will demonstrate this in Q&A sessions.
6. Construct basic **Project4** webapp framework inside Tomcat webapps folder.
7. Construct initial web.xml file in **Project4/WEB-INF**. Additional refinement will be needed later.
8. Deploy files from step 1 above and test/refine in browser of your choice.
9. Create properties files for the root-level users, client-level users, accountant-level users, and a system-level user. These will be placed in the **lib** folder of the **Project4** web app (i.e. **Project4/WEB-INF/lib**).
10. Begin development of the servlets. Basic operation of the root-level servlet and the client-level servlet are the same (more later on the actual differences of these servlets). 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".
11. You will ultimately have servlets supporting the root-users, client-users, and accountant-level users. There is also a servlet that support the authentication.html page. (See the image on page 6 above.)
12. Load servlet test files into Tomcat **Project4** webapp in correct location and perform initial integration testing of complete package.
13. **IMPORTANT: Keep back-up copies of all files within your webapp in a location NOT inside Tomcat. Maintain these backup copies religiously.**
14. 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.
15. Add business logic to the root-level servlet – develop non-bonus version first.
16. **Optional:** implement the bonus-version of the business logic.
17. Recreate the **project4** database, by re-executing the script **project4DBscript.sql**
18. Run through the three user-level command scripts and generate screenshots from your application running these commands.
19. 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.

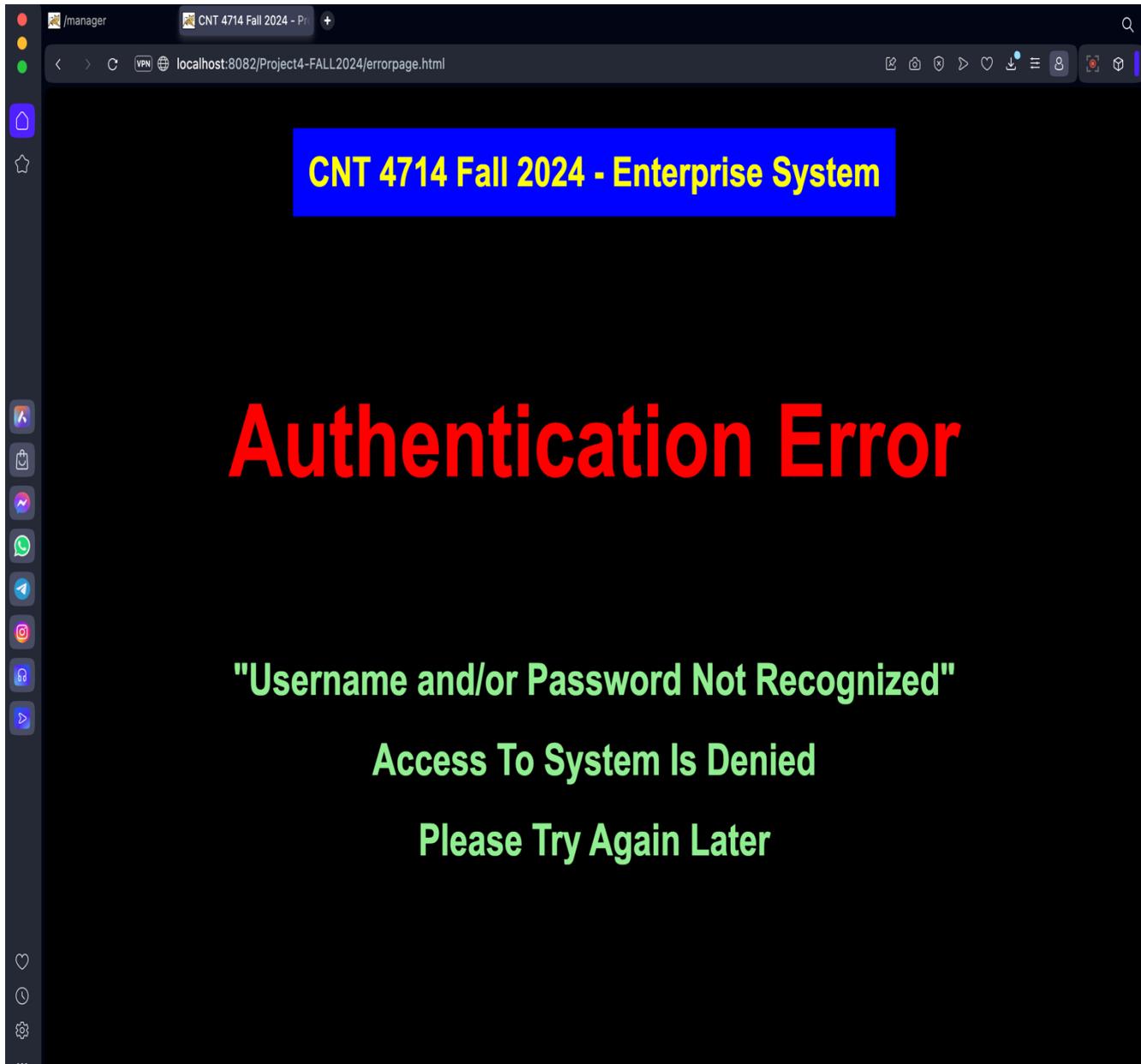
**Some screen shots illustrating the application.**

### **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.csv file. The errorpage.html is displayed.



If the user's login credentials are validated (they match with an entry in the `usercredentials` table in the `credentialsDB`), 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 the following details:

- Title Bar:** /manager CNT 4714 Fall 2024 - Pr...  
localhost:8082/Project4-FALL2024/rootHome.jsp
- Page Content:**
  - Welcome Message:** Welcome to the Fall 2024 Project 4 Enterprise System
  - Section Title:** 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 **root-level** user.  
Please enter any SQL query or update command in the box below.
  - User Input Area:** A blue box containing the SQL command `select * from suppliers`. A red box labeled "User input area." with an arrow points to it.
  - Control Buttons:** Three buttons labeled "Execute Command", "Reset Form", and "Clear Results". A red box labeled "Three control buttons" with arrows points to each button.
  - Execution Results:** A section labeled "Execution Results:" followed by a list item. A red box labeled "All results are returned in this area." with an arrow points to the list item.

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:

- Title Bar:** Shows three tabs: "/manager", "CNT 4714 Fall 2024 - Pr", and "CNT 4714 Fall 2024 - Pr". The second tab is active.
- Address Bar:** Displays "localhost:8082/Project4-FALL2024/clientHome.jsp".
- Content Area:**
  - Welcome Message:** "Welcome to the Fall 2024 Project 4 Enterprise System" in red text.
  - Section Title:** "A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container" in cyan text.
  - Text Message:** "You are connected to the Project 4 Enterprise System database as a **client-level** user. Please enter any SQL query or update command in the box below."
  - Input Box:** A large blue-bordered input area containing the SQL command: "select \* from suppliers".
  - Buttons:** Three buttons at the bottom of the input area: "Execute Command" (green), "Reset Form" (yellow), and "Clear Results" (yellow).
  - Text Placeholder:** "All execution results will appear below this line." followed by a horizontal line.
  - Execution Results:** A section labeled "Execution Results:" with a single bullet point: "-".
- Left Sidebar:** A vertical sidebar with various icons, including a home icon, a star, a search icon, and several utility icons.
- Bottom Sidebar:** A horizontal bar with icons for heart, clock, gear, and three dots.

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

The screenshot shows a web browser window with three tabs open, all titled "CNT 4714 Fall 2024 - Proj". The active tab displays the URL "localhost:8082/Project4-FALL2024/AccountantUserApp". The page content is as follows:

# Welcome to the Fall 2024 Project 4 Enterprise System

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

---

You are connected to the Project 4 Enterprise System database as an **accountant-level** user.  
Please select the operation you would like to perform from the list below.

- [Get The Maximum Status Value Of All Suppliers](#) (Returns a maximum value)
- [Get The Total Weight Of All Parts](#) (Returns a sum)
- [Get The Total Number of Shipments](#) (Returns the current number of shipments in total)
- [Get The Name And Number Of Workers Of The Job With The Most Workers](#) (Returns two values)
- [List The Name And Status Of Every Supplier](#) (Returns a list of supplier names with status)

**Execute Command**   **Clear Results**

All execution results will appear below this line.

---

Execution Results:

(This section is currently blank.)

## 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 three tabs open, all titled "CNT 4714 Fall 2024 - Pr". The active tab's URL is "localhost:8082/Project4-FALL2024/RootUserApp". The page content is as follows:

**Welcome to the Fall 2024 Project 4 Enterprise 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 SQL query or update command in the box below.

```
select * from jobs
```

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

All execution results will appear below this line.

**Execution Results:**

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 three tabs open, all titled "CNT 4714 Fall 2024 - Pr". The active tab displays the URL "localhost:8082/Project4-FALL2024/RootUserApp". The page content is as follows:

**Welcome to the Fall 2024 Project 4 Enterprise 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 SQL query or update command in the box below.

```
select something from parts
```

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

---

All execution results will appear below this line.

---

**Execution Results:**

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

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):

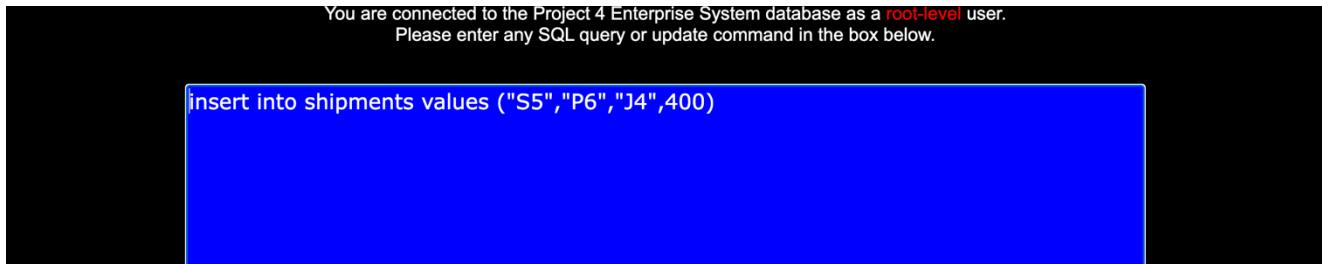
Note the current status of supplier number S5. (Also note status of S1, S12, S17, S21, S22, S3, S44, and S6.)

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 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	4	London
S56	Marianne Vos	8	Zandvoort
S6	Nicola Gianniberti	2	Milan
S7	Christian Albers	3	Orlando
S8	Giancarlo Fisichella	3	Milan
S9	Kimi Rikonnen	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 18-19. Notice that the supplier S5’s status is currently 4.

Root user issues the following insert command:



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

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

An 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).



Welcome to the Fall 2024 Project 4 Enterprise 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 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.

**Execution Results:**

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 Meeuwis	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 15 (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.

The screenshot shows a web browser window with three tabs open, all titled "CNT 4714 Fall 2024 - Pr". The main content area displays the following text:

**Welcome to the Fall 2024 Project 4 Enterprise 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 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.

---

**Execution 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 four tabs open, all titled "CNT 4714 Fall 2024 - Pr". The active tab displays the URL "localhost:8082/Project4-FALL2024/RootUserApp". The page title is "Welcome to the Fall 2024 Project 4 Enterprise System" and the subtitle is "A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container". A message indicates the user is connected as a root-level user and can enter an SQL query or update command. A text input box contains the command "update shipments set quantity = 20". Below the input box are three buttons: "Execute Command" (highlighted in green), "Reset Form", and "Clear Results". A note below the buttons states that execution results will appear below the line. The "Execution Results:" section shows the output: "The statement executed successfully. 57 row(s) affected.", "Business Logic Detected! - Updating Supplier Status", and "Business Logic updated 0 supplier status marks.". A red box highlights a note in the bottom-left corner about an unsafe update.

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

```
update shipments set quantity = 20
```

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

All execution results will appear below this line.

**Execution Results:**

The statement executed successfully.  
57 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 there are currently 57 rows in the table and all 57 rows will now have a quantity of 20.

## 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 13).

The screenshot shows a web browser window with three tabs open, all titled "CNT 4714 Fall 2024 - Proj". The active tab is "localhost:8082/Project4-FALL2024/ClientUserApp". The page content is as follows:

**Welcome to the Fall 2024 Project 4 Enterprise 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 SQL query or update command in the box below.

```
select * from jobs
```

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

---

All execution results will appear below this line.

**Execution Results:**

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 three tabs open, all titled "CNT 4714 Fall 2024 - Proj". The active tab displays the URL "localhost:8082/Project4-FALL2024/ClientUserApp". The page content is as follows:

**Welcome to the Fall 2024 Project 4 Enterprise 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 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.

---

**Execution Results:**

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

## Accountant-level User Examples

The accountant-level interface provides a list of “reports” that can be run against the project4 database. These reports are generated by invoking remote procedure calls to execute stored procedures residing on the MySQL DB server. The remote procedures are invoked using the CallableStatement interface. As with the other interfaces in this web application, the results of the remote procedure execution will be returned to the bottom of the original page.

An accountant-level user select first option.

The screenshot shows a web browser window with three tabs open, all titled "CNT 4714 Fall 2024 - Pr". The active tab displays the "AccountantUserApp" page. The page has a dark background with yellow text. At the top, it says "Welcome to the Fall 2024 Project 4 Enterprise System" and "A Servlet/JSP-based Multi-tiered Enterprise Application Using A Tomcat Container". Below this, a message states: "You are connected to the Project 4 Enterprise System database as an **accountant-level** user. Please select the operation you would like to perform from the list below." A list of six operations is provided, each in blue text:

- Get The Maximum Status Value Of All Suppliers (Returns a maximum value)
- Get The Total Weight Of All Parts (Returns a sum)
- Get The Total Number of Shipments (Returns the current number of shipments in total)
- Get The Name And Number Of Workers Of The Job With The Most Workers (Returns two values)
- List The Name And Status Of Every Supplier (Returns a list of supplier names with status)

At the bottom of the list are two buttons: "Execute Command" (green) and "Clear Results" (red). Below these buttons, a note says "All execution results will appear below this line." Under the heading "Execution Results:", there is a table with one row and one column, containing the value "8".

Maximum_Status_Of_All_Suppliers
8

An accountant-level user selects the fourth option:

Welcome to the Fall 2024 Project 4 Enterprise System

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

You are connected to the Project 4 Enterprise System database as an **accountant-level** user.  
Please select the operation you would like to perform from the list below.

- Get The Maximum Status Value Of All Suppliers (Returns a maximum value)
- Get The Total Weight Of All Parts (Returns a sum)
- Get The Total Number of Shipments (Returns the current number of shipments in total)
- Get The Name And Number Of Workers Of The Job With The Most Workers (Returns two values)
- List The Name And Status Of Every Supplier (Returns a list of supplier names with status)

**Execute Command**   **Clear Results**

All execution results will appear below this line.

**Execution Results:**

jname	numworkers
Really Big Job	500

**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 16), we now get the correct effect for our update command.

/manager CNT 4714 Fall 2024 - Pr CNT 4714 Fall 2024 - Pr CNT 4714 Fall 2024 - Pr +

< > C VPN localhost:8082/Project4-FALL2024/RootUserApp

# Welcome to the Fall 2024 Project 4 Enterprise 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 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.

---

**Execution Results:**

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

**Business Logic Detected! - Updating Supplier Status**

**Business Logic updated 1 supplier status marks.**

### 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 Rikonnen	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 17.

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.