

**BCIS 3680**  
**Enterprise Systems (Advanced Java)**  
**Problem #4: Introduction to JSP and servlets**

In past applications, we have been concentrating on producing solutions that were based on local (client-side) deployments. With this solution, we will re-configure the user interface so that the application (or a close approximation thereof) will operate in a web-based (server-side) deployment.

In general, our design has been composed of a “splash” screen which has served as a central “switch board” for our application. The basic configuration of this part of our application has been centered on the use of menus. However, menus are not very typical of web-based applications. Consequently, we will do a bit of redesign of this part of the application (and other segments that have menu choices). The replacement for the menu choices should be a series of command buttons which perform the same (or reasonably comparable) functions.

For purposes of problem #4, we will eliminate the “Expense Maintenance” function from the “splash” screen. However, the other remaining functionalities from problem #3 should be represented in your solution to problem #4. It is not anticipated that you would use the FileChooser class to for purposes of selecting files and navigating within a disk directory. Rather, it makes more sense for this type of application to operate on the basis of a program assigned file name, based on the previously used default (employee number/date value).

For past applications, we have values associated with the employee number, expense code (etc.) were simple selections, with the customer name, expense name (etc.) being supplied by database retrieval. You should replicate this part of the application with an html layout that is supported with Java Server Pages (JSP's). You should strive to reproduce the previous functionality as closely as possible.

The much of the remaining segments of the application has to do with selection of exchange records and associated values, i.e., country code (and associated country name), currency name, and the conversion rate. Again, this should be “web-deployed” and the functional code should be in the form of servlets, where possible. This does mean that a separate (somewhat independent) layout could be used to support identification of values and retrieval of appropriate entries associated with an inventory record.

And, as with previous applications, the screen that shows the employee expense-related data should also provide a “historical record” of all selected purchases. And, again, these accumulated expense items should be written to a file when instructed to do so.

Finally, remember that exit functions should transfer control back up the application structure, most commonly to the “splash” screen. There should be no exit function here. Rather the user should terminate the application by closing the browser.

You should submit normally required supporting material for this application.

**Due Date:** November 18th

**Value:**

10 percent