**Critical Evaluation**

**About the Application and the Developer (Me)**

The application follows all the requirements specified in the stock portfolio document except the digital signature requirement. Although it is not a big project but it took me much time to execute appropriate methods with syntax I didn’t knew. I went through numerous errors before completing my project, and now I am quite comfortable with JEE. Thanks to Ian for providing necessary information.

**Securing the Application**

The method I chose for my web application was the basic login. In this method the browser provides the login window. The user name and password are stored in the glassfish server. The advantage of this method for securing the web application is that we can create multiple roles for accessing the content in a different way. We can classify different users in different roles and each user will have different privileges and views for the web application. These privileges can range from full access to no access depending upon the user-roles. Another advantage of this method is the user-group role mapping option. The developer does not require learning the roles of the users defined in the realm, because he can edit that feature and use the option of mapping roles to users defined in the application to that of the groups in the realm. The password stored in glassfish server cannot be accessed by unauthorized user. By setting the user data transport to confidential, setting the https protocol, the information becomes secured within the application so that unauthorized users can access the others private data such as passwords. Another thing which could be done was issuing a certificate which would check the user every time they connect to the application to ensure the transaction process is encrypted and secured with proper authentication. But there some limitations using the HTTPS, it is vulnerable to crypto attacks where data can easily be decrypted while it is transferred in between the browser and the server by unauthorized users such as hackers if they are using the same network as the target.

**Extending the Design**

The Extension in the application will depend on the amount of users it expects to access it at the same time. Every server and database has a limit to a number of users they can allow to access them. Server plays a very important role as it acts as an interface between the client and information. A single problem can arise from a failure in hardware to emergency shutdown because it might not be able to handle much load. So the best way to deal in this situation would be to create one or two backup servers depending upon the requirements, where one server can divert the traffic to another in case of many users or when one server goes down because of a system failure. This will ensure there is no single point of failure for users to access the server unless we assume the probability that all servers go down at once.

**Concurrent Users**

Since many users are going to access the application and there is a higher probability that they will execute a transaction simultaneously, which could cause problems like transactions to fail when creating, updating, deleting in the process. We don’t want this to happen with our system, so access to each database transaction can be controlled depending on the preference the system provides to the user which is the first one who executes a transaction will cause the database to lock for other transactions in queue and once it finishes, the next transaction will take place one waiting in the queue and so on. Database will carry out a successful transaction once it ensures the stable condition of the database, if the transaction causes instability in the database then database will roll back to its original state deleting that transaction. Locking the database will depend on the situation that if once user only wants to read the data while the other is updating it, this could be done simultaneously where as updating the database at once will not be permitted. Further details can be found in the ACID properties of a transaction which stands for atomicity, consistency, integrity and durability.

This is what I would recommend for extension of application. In my application I have 2 ways of accessing it, once can run the application without any authentication and the other is by using the secure admin page for a secured transaction so the user name and password for it is **admin** and **password** respectively.