SSC Coursework Breakdown

**Client application** - An application running in a user's machine.

**Web Service** - A web service is an Application Program Interface (API) that runs on the server (Glassfish), which provide data to the client over http through a standardized messaging system.

**Markup Language** - A markup language is a computer language that uses tags to define elements within a document.

**Maven** - Maven is a build automation tool used primarily for Java projects. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. It does this by configuring your java project. For example, setting source folders, classpath etc. based on the maven poms. it allows you to easily get the required jars your application may need

**Pom.xml** - It is an XML file that contains information about the project and configuration details used by Maven to build the project.

**XML (eXtensible Markup Language)** - Designed to store and transport data.

**JAXB (Java Architecture for XML Binding)** - A software framework that allows Java developers to map Java classes to XML representations. JAXB allows storing and retrieving data in memory in any XML format.

**WSDL (Web Services Description Language)** - An XML-based file that basically tells the client application what the web service does. The WSDL file is used to describe in a nutshell what the web service does and gives the client all the information required to connect to the web service and use all the functionality provided by the web service.

A - Core Web Service (weighting - 35%)­

Implement Shares Brokering service

**Theory**: The angular thing helps format the web service operations on the web browser.

**Theory**: The web service does the calculations and the client calls upon the ‘services’ that perform the calculations that are then presents to via the client app.

**Theory**: Unmarshelling data is what allows the web service to access the data in the XML file.

**Theory**: Maybe it is not possible to do the marshalling the way we originally thought. Maybe instead the client application should write the data to a database or database-like file, and the code should marshal it after each run.

**Theory**: Unmarshall XML file first, then marshal with new data + existing data.

**Theory**: Maven is not needed for JAXB to work (Maven isn’t mentioned at all on JAXB’s Wikipedia page). It may potential be a way to make the system work.

Maybe need to convert main class to POJO structure (Copy AAD Work)

**Part A (Done)**

Current shares on trade should be held in an XML file based on an XSD schema containing:

* company name (**DONE**)
* company symbol on the stock exchange (**DONE**)
* number of available shares (**DONE**)
* a complex ‘share\_price’ element containing currency and value (**DONE**)
* date of the last-update of the share price. (**DONE**)

You can opt to use conventional structures to store the data (arrays, Lists, etc.) instead of XML objects. However, this will affect the standard of your work and also progression into Section-B.

**Part B**

*Check Lab 7 part 2*

The web service should allow users to:

* list shares details
* update the number of shares on offer when a purchase is made.

You might opt to utilise JAXB to generate helper Java classes, which allow reading and populating the XML documents.

You may choose to implement the core Web Service using as a RESTful service, but for full marks you need to evidence that you can build & communicate a Shares data structure similar to what can be provided by an XML Schema as in 1.a) above.

*Reading and writing XML (*[*Link*](https://stackoverflow.com/questions/7373567/how-to-read-and-write-xml-files)*)*

*Marshalling data (*[*Link*](https://stackoverflow.com/questions/21751624/how-to-write-data-into-xml-file-using-jaxb)*)*

*More Marshalling (*[*Link*](https://www.intertech.com/Blog/jaxb-tutorial-how-to-marshal-and-unmarshal-xml/)*)*

*Even more Marshalling (*[*Link*](https://stackoverflow.com/questions/13788617/jaxb-marshalling-java-to-output-xml-file)*)*

*JAXB and Root Elements (*[*Link*](http://blog.bdoughan.com/2012/07/jaxb-and-root-elements.html)*)*

*Writing data to file (*[*Link*](https://stackoverflow.com/questions/29473055/how-to-write-to-the-next-blank-line-in-a-document)*)*

*JAXB Marshaller Overwriting data (*[*Link*](https://stackoverflow.com/questions/29135755/jaxb-marshaler-overwriting-file-contents)*)*

*Marshalling multiple pieces of data (*[*Link*](https://howtodoinjava.com/jaxb/jaxb-exmaple-marshalling-and-unmarshalling-list-or-set-of-objects/)*)*

Without Maven

*JAXB Tutorial (*[*Link*](https://www.vogella.com/tutorials/JAXB/article.html) *1)*

**Maybe - Create database, have user write to it with new share data, then after they save the work it needs to be marshalled into “Share\_Data.xml” file (Use** [**LINK**](https://howtodoinjava.com/jaxb/jaxb-exmaple-marshalling-and-unmarshalling-list-or-set-of-objects/) **for help marshalling a list), which can then be accessed by the web service.**

**Maybe – Add POJO file for Shares (Follow guide** [**here**](https://howtodoinjava.com/jaxb/jaxb-exmaple-marshalling-and-unmarshalling-list-or-set-of-objects/)**). Figure out how that code works then apply it to our project.**

**Part C**

Implement a search functionality allowing customers to list shares’ offerings using various criteria such as company details, highest price, etc. More sophisticated search functionality will merit higher marks.

* The choice of client is your decision. Java GUI Apps or JSP are more appreciated but should not be attempted at the expense of providing core functionality