COMP0004 Coursework 3

Purpose: To develop a Java web application.

Submission: Upload to Moodle before 16:00pm 17th May 2021. To submit create a zip file of your IDE project and upload that, including your report document file (in pdf).

Marking: This coursework is worth 50% of the module mark. Complete as many of the requirements as you can. It is better to submit a well-written, working program that covers a majority of the requirements, than a poorly written program that tries to do everything.

Inadequate (0-39)	Failed to demonstrate a basic understanding of the concepts used in the coursework, or answer the questions. There are fundamental errors, code will not compile, or nothing of significance has been achieved. Very poor report.(F)
Just Adequate (40-49)	Shows a basic understanding, sufficient to achieve a basic pass, but still has serious shortcomings. Code may compile but doesn't work properly. Not all concepts have been understood or all the requirements implemented correctly. Report is just good enough but not well written.(D)
Competent (50-59)	Reasonable understanding but with some deficiencies. The code compiles and runs, the concepts are largely understood. This is the default for a straightforward, satisfactory answer. Most requirements answered, report is reasonable. (C)
Good (60-69)	A good understanding, with possibly a few minor issues, but otherwise more than satisfactory. The code compiles, runs and demonstrates good design practice. Most expectations have been met. All requirements answered. Quite good, clear report. (B)
Very Good (70-79)	A very good understanding above the standard expectations, demonstrating a clear proficiency in programming and design. All requirements answered very well, very good report showing good insight into the design and programming process. (A)
Excellent (80-100)	Programming at a level well above expectations, demonstrating expertise in all aspects. This level is used sparingly only where it is fully justified. All requirements answered excellently, excellent report with real depth and insight. (A+, A++)

Q1. Implement a Java Web Application using the requirements specification below (75%).

Q2. Write a report on your work with this content (25%):

- Section 1: Summarise what features your program implements, half a page maximum.

- Section 2: Provide an UML class diagram for your application. Include your classes and interfaces only, standard Java library classes and interfaces do *not* need to be included. You should include your Java Server Pages (JSPs) and treat them as Java classes, as JSPs are converted into classes. Use basic class icons only, no need to add variables or methods inside the icon, just have the class or interface name, otherwise you should get the UML syntax correct (use the right kinds of lines and arrows). The diagram should easily fit in onto a single page. Note: you should design and draw your own class diagram (neat hand drawn is OK). The IntelliJ IDEA IDE can generate class diagrams, don't simply copy and paste one of them! IDEA diagrams will include lots of dependency arrows (dashed lines) that you do not want to include.

- Section 3: Describe and evaluate your design and programming process. You should reflect on how you went about designing your classes, why they are appropriate classes, whether you have used good OO design practice (e.g., good use of abstraction, cohesive classes), and the overall quality of your work (you decide the criteria for this).

- The overall report should be a *maximum* of 3 pages (sides of A4 paper) in length.

Requirements

Implement a program for as many of the requirements listed below that you have time to complete. You can use any of the classes provided in the standard JDK, plus the Tomcat and servlet classes that are provided via the example code. You should not add any other additional third-party libraries libraries and do not need to use a database. Use JDK 15 (or the JDK you used for the previous courseworks). JDK16 has now been released but you do not need to upgrade to it.

The requirements form a specification for the required program, so they don't have to be answered in order and you may well find it easier to work on each requirement in stages as you develop your program.

Overview

The program is for storing, searching, and viewing lists of information, essentially a general list making application. While this is implemented as a web application assume it is for your own personal use and only accessible by yourself, so no user management or logging on is required. The data should be stored in files, you decide the format.

The user interface should be implemented as web pages viewed in a web browser, using standard html and css. You don't need to implement a sophisticated user interface, or start using css libraries like Bootstrap - this module is primarily about Java programming not creating complex front-ends for web applications! Don't get side-tracked in to spending too much time on the appearance of web pages.

It is up to you to interpret the requirements and decide the details of what you actually implement.

Requirement 1.

The program can store one or more lists of items. Each list has a name or label, and items might include text, an URL, an image, some other kind of data, or a combination of these such as an item with text and an image.

Requirement 2.

An item in a list can be a link to another list of items, making it possible to have lists of lists.

Requirement 3.

Item lists can be created, deleted, or renamed, and viewed in one or more ways.

Requirement 4.

Items in a list can be added, removed, or edited. An item can be a link to another list.

Requirement 5.

One, multiple, or all item lists can be searched, and lists can be located by searching for the name of the list.

Requirement 6.

A list is automatically saved to a file in some format (you decide), so that the user does not have to explicitly load or save to a file. When a list or items are modified the change(s) are immediately written to the file.

Design Notes

You should use Java servlets to receive requests (i.e., when the user clicks an URL or link on the web page, or submits data in a form). Web pages should be generated by Java Server Pages (JSPs).

Remember the Model-View-Controller (MVC) pattern. A servlet is a controller so should contain only the code to receive a request, call methods on the model to carry out actions, and forward to a JSP to display the results.

A servlet will need to get a reference to the Model via the Singleton Pattern. This means that the Model provides a static method to return a reference to a Model object - see the example code for how this works.

The example code (see the Moodle site) and videos show how to set up a web app project. You do need to use Maven to manage, build and run applications as shown. The best way to start is to clone and example project, and then start extending it.