COMP0004 Coursework

Purpose: To develop a Java web application.

Submission: Upload to Moodle before 16:00pm 17th March 2025. Submit a zip file of your IDE project and upload that, including your report document file (in pdf).

Marking: This coursework is worth 10% 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 may not compile or run, 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 enough requirements implemented correctly. Report is just good enough but not that 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. The majority of requirements implemented, report is reasonable. (C)
Good (60-69)	A good understanding, with possibly a few minor issues, but otherwise a proficient answer. The code compiles, runs and demonstrates good design practice. Most expectations have been met. Nearly 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 following the requirements specification below (80%).

Q2. Write a report on your work with this content (20%):

- Section 1: Summarise what features your program implements, half a page maximum.
- Section 2: 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 but be realistic in your evaluation.
- The overall report should be a *maximum* of 2 pages (sides of A4 paper) in length.

Requirements Specification

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 the Jakarta EE libraries for servlets and JSPs that are provided via the example code (see links at the end). You should not add any other additional third-party libraries, except for reading/writing csv, json, or xml format files, and do not need to use a database. Use Java 23.

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

Overview

The program is for storing, searching, and viewing collections of notes. 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 of those files (e.g., text, json, etc.).

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, but you might want to experiment.

This module is primarily about Java programming not creating complex front-ends for web applications! Don't get side-tracked into 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 kinds of note. Each note has a name or label, might hold text, a URL, an image, some other kind of data, or a combination of these such as a note with text and an image. A basic note just holds text.

Requirement 2.

An index is used to hold the collection of notes and the index can be displayed. Clicking on an index entry displays the corresponding note.

Requirement 3.

Notes can be created, deleted, edited, and renamed. Notes can be viewed in one or more way, for example, in sorted order, in the order added, a summary, or the full note. When a note is created it is added to the index, and removed when deleted.

Requirement 4.

The index and the notes in the index can be searched to display a list of matches. Clicking on a match displays the content of the relevant note. Ideally, a free-form text search is supported, meaning that the search string matches any text in a note. If you store images or other kinds of data think about how they can be searched.

Requirement 5.

Notes and the index are automatically saved to files in some format (you decide), so that the user does not have to manually load or save to a file. When an index or a note is modified the change(s) are immediately written to the files.

Requirement 6.

Add the functionality to group notes into named categories, for example, reminders, appointments or birthdays. Update the way notes are displayed, indexed and searched accordingly. A category can be represented as index, so that an index entry can now be another index of notes. Notes can be in more than one index.

Requirement 7 (Optional challenge)

Add your own functionality to the application, for example more advanced formatting of notes, additional kinds of notes, and so on. This is a chance to demonstrate your programming abilities but stick to the use of standard Java, Jarkata EE, etc. as outlined earlier. It must be possible to build your program using Maven, without having to install any other software.

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 it 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 and notes on Moodle show how to set up a web app project. You do need to use Maven to manage, build and run applications as shown – don't use alternative application structures that the IDE supports. The best way to start is to clone the example project.

Example Web Application Projects on GitHub:

https://github.com/UCLComputerScience/COMP0004HelloWorldWebExample

https://github.com/UCLComputerScience/COMP0004JavaWebAppExample