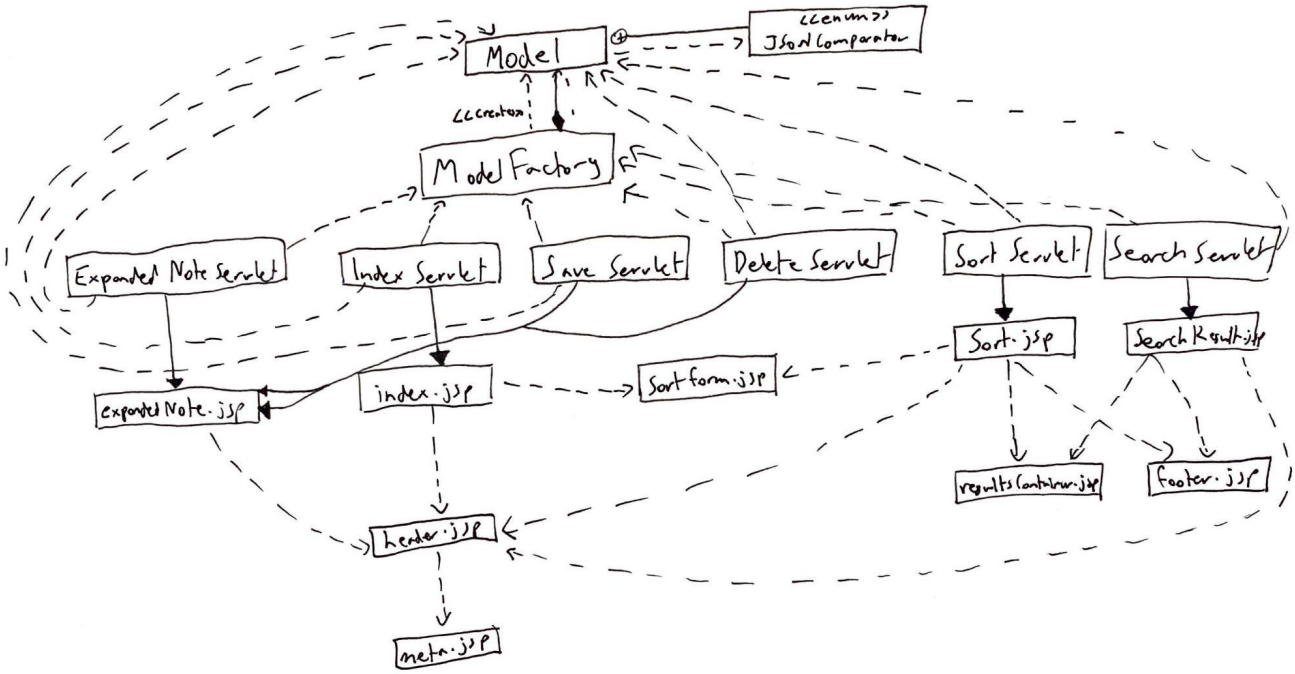
COMP0004

Java Coursework Report

Section 1: Features of the program

The program meets all of the requirements of the specification. Notes have a title section and a content section that can hold text. A summary of all the user's notes are displayed on the main page in the order they were added, with each note's title and content being formatted into a sticky-note-like UI element. Any excess text is truncated so a consistent format is maintained. The index of the note is also displayed on each note. Upon clicking one of the notes, the user is taken to a page allowing them to view the full content of the note and edit the title and content fields. They can then save any changes by simply clicking the save button, and the program automatically saves it to the JSON file holding all of the notes. The user can also delete the note by clicking the delete button, and the program again automatically updates the JSON file, and removes the note from the main page. The user is also able to add a new note, where it is saved to the JSON file automatically and added to the main page. The user can search for a term in the search bar, and the website displays all the notes that contain the search term in the title, content or index of the note. The user is also able to sort the note by title or content in ascending or descending order. Both the sorting and searching results have the same format as the main page, and clicking on the notes in the results page acts the same way as normal, allowing the user to view the full note, edit the fields and delete the note.

Section 2: UML Diagram



Section 3: Evaluation of design

I feel like my classes are overall well designed, and closely follow the well-established Model-View-Controller pattern. The Model class holds everything pertaining to the JSON representation of the note. It contains methods to read from the JSON file that stores the notes data and stores its contents in a private JSONObject instance variable. This variable can only be accessed and changed using get and set methods, following the OOP idea of encapsulation. The class also contains methods to sort and search through the notes. To implement sorting in an object oriented manner, the class contains an enum that implements the Comparator interface and defines the comparator objects needed for the different types of sorting. This comparator is then passed on to the sort method to sort the note based on the sorting parameters. This means that if our data structure or sorting methods change, we can simply change the enum to reflect this, abstracting away a lot of complexity from the sort method. The Model also follows the singleton pattern, requiring the ModelFactory class to be instantiated. Using the singleton pattern for this class

prevents a lot of issues, such as multiple Model instances competing to write to and read from the JSON file, and potentially erasing changes.

The Servlet classes act as the controllers, and deal with receiving HTTPrequests, calling methods on the Model, and invoking JSP pages to display results on a webpage. The servlet classes follow the single responsibility principle and only carry out a specific task, ensuring that they are more robust to future modifications. There are separate servlets for the main operations of the program (save, delete, sort and search). There are two other servlets to handle the two main view modes of our website. The index servlet passes on all of the notes to the index jsp to display the main sticky-note-like page. The expanded note servlet passes on all the information of a certain note to the expanded note jsp to display the full content of the note.

The JSP pages act as the view component. They receive attributes from the servlet classes and display it on a webpage. Common elements such as the navbar, sort forms, results page, and footer are separated out into their own JSPs and included when needed. This reduces repetition and makes implementing any changes to these commonly-used elements much easier as only the specific JSP has to be modified. Most of the necessary operations such as saving, deleting, searching and sorting are performed using HTML forms that send requests and the associated parameters, such as the unique ID of a note, to the servlets. The servlets then make changes to the Model and the JSON file based on these parameters. These changes are then reflected when the servlets call the JSPs again but instead pass attributes from the updated Model.

Overall, I believe my classes are well-considered, have clearly defined responsibilities, follow common OOP principles, and strongly adhere to the MVC pattern. All of these make the program more organized and robust, and make it much easier to add or modify features in the future.