COMP0004 Coursework Report

Zhouzhou Zhang, *Bsc Computer Science*

Feature

I implemented a Java Web application for storing notes based on MVC structure. Notes can be read, searched, added, modified and deleted. The content of Notes can include plain text, image and external url. Notes are also catalogued into several categories, including “Casual”, “Important” and “Action”. Notes can be either viewed as a list or individually. When viewed as a list, the detail of notes will be summarized. A note includes a key, a category, a label, a text(content) and a timestamp, where image and external link are optional. For the backend, instead of a database, the source text of Notes is stored in a CSV file and then be loaded into the memory as a HashMap for the performance of searching. Updating the notes will automatically update the CSV file. For the frontend, JSP is used to implement the View section and JavaScript is used to enhance the UI interaction.

Class

The design of classes is mainly in “model” folder. I wrote a class called **Note** to store the data which is read from the CSV file. **Note** is a bridge between CSV file and View section. With **Note** class, to pass the detail of a note, servlet only needs to pass a note rather than a list of source text from CSV file, which improves the scalability and clarity. More importantly, as the detail of Note is abstracted, using this class can significantly reduce the lines of codes. A **Note** object includes id, label, text, timestamp, category (what is this note about), noteType (what is the content, plain text or image?). The label, text, category and noteType are determined by the user, while others are determined automatically. category and noteType are two enum classes. Note object has methods for getter and setter, also with a method to convert itself to a list of string, which is then used to store into the CSV file. Note is also the parent class of **LinkNote** and **ImageNote**.

These two enum classes, **Category** and **NoteType**, stand for the limited options of the category and types that user can choose from. Through using an enum class rather than just a String, the robustness has been improved because invalid input is not allowed. Additionally, the CSV file is much safer as every enum field that wrote in must be a valid enum object. Enum fields only accept enum classes, while a String field may accept anything that is a String. **Category** contains CASUAL, IMPORTANT and ACTION. **NoteType** contains DEFAULT, IMAGE and LINK.

LinkNote and **ImageNote** are the child classes of Note. They have all the fields of Note, but they have one more special field. For **LinkNote** it is an external url and for **ImageNote** it is an image (base64 encoded). Through the use of inheritance, there is no need to rewrite the whole class, with most things are already done by their parent class, Note. This makes the development easier when adding new types of **Note** and the frontend doesn't need to change a lot.

Model is another class. It can be understood as a collection of **Notes**. In **Model** class, there is a HashMap to store every **Note** for its performance and convenience. The **Model** class further abstracts the detail of Note. For the basic operation, **Model** can return every note, or return notes filtered by a category and add notes. Furthermore, when given a **Note**'s key, there are public methods to modify any field of any Note. There are also private methods to read from and write into the CSV file, which are called every time a note is modified. While user doesn't know the detail, this abstraction provides user with a convenient way to interact with the data. A **Model** object should be defined through another class, **ModelFactory**, which makes sure that a **Model** object is correctly initialized.

Process

1. Model reads from the CSV file, store the data as a HashMap of Note.
   User then can see the Note.
2. When user creates new note, updates or deletes any note, the data is POSTed to the servlets. Servlets call Model to add the note.
3. When user filter the notes, servlets call Model to generate a filtered HashMap of Notes and give it to the View. If no filter then just return the original HashMap.
4. Every time Model is called for adding, deleting or modifying, Model automatically saves the data into the CSV file, generate a proper key for new note and update the timestamp.

Quality

I feel good with these classes. I consider myself familiar with OO design and programming. Through this project I enhanced my ability of Backend development and got closer to the practical environment. Though there is a small problem, the size of uploaded image is incredibly restricted, due to the CSV file. It would be much better if I can compress the photo and use a database rather than a file storage. Afterall I am still happy with my job so I would give myself a 4 out of 5.