COMP0004 Coursework 3 Report

Written By: Cheng Loo

Section 1: Overview of Program

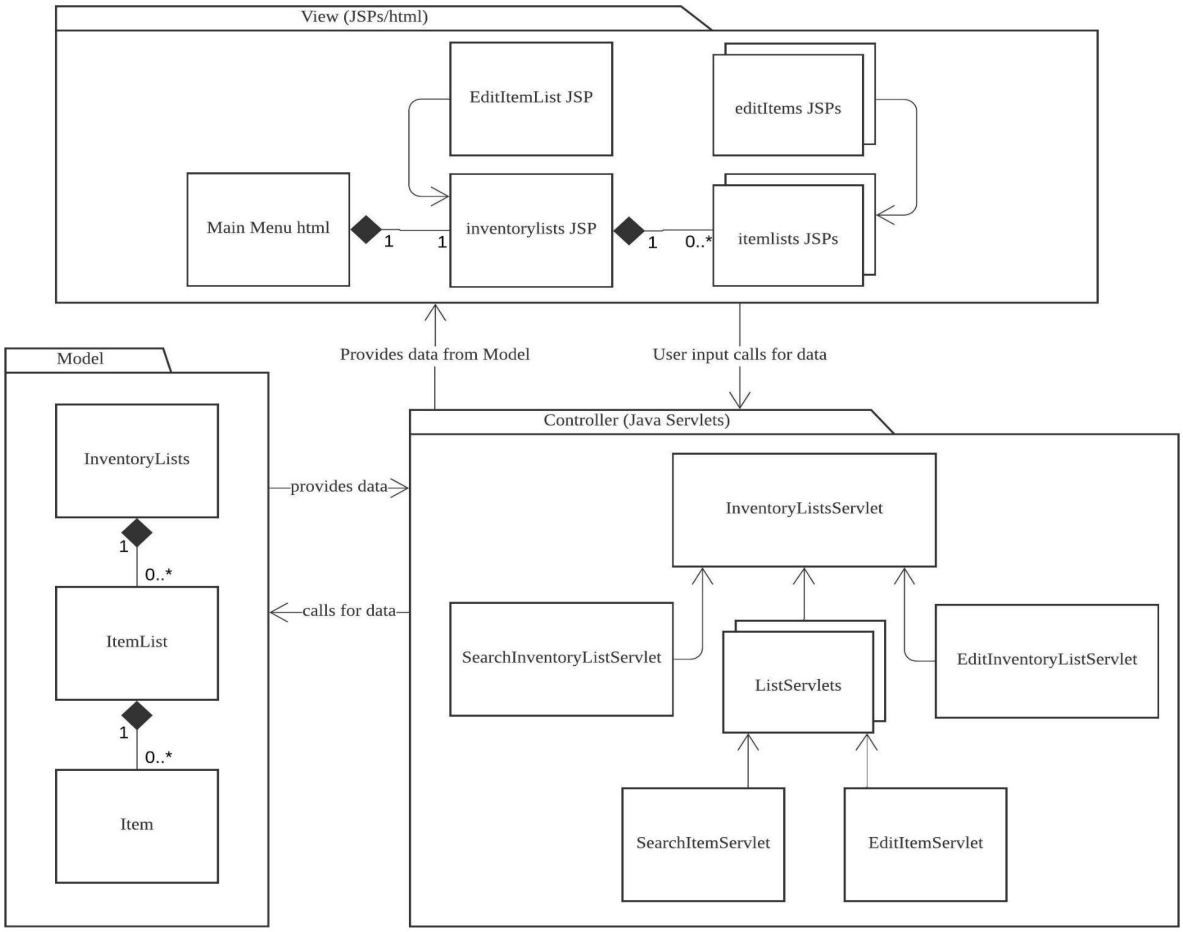
The program emulates a simple inventory system that allows a user to input different categories of items in their inventory.

On the main menu, the user is able to see the address of the main menu webpage, as well as the link to access the inventory. After which the user is brought to the inventory lists page, where they can add their own item categories and a new list page for that category is generated. They can also delete categories and rename them, as well as search for a particular list.

When the link to a category is clicked, the user then accesses the item list page where they can add items to that category, which includes the URL or image if they wish. The user can also delete, rename or edit the item, based on the name of the item. A search input is also included to allow users to search for their item.

All data that is added, removed, or edited is handled by saving and loading from Comma-Separated Values (CSV) files in the database package, and the server constantly updates the database and the webpage every time new data is passed from the user.

Section 2: UML Class Diagram



Section 3: Evaluation

Process

The main design process comes with the perspective of creating a Model-View-Controller (MVC) pattern that normally encompasses all programs that include a UI, like a web application. Since the program is one that is able to create item lists and have items that can be added or removed, an inventory is one of the simplest uses of such a program.

First, the model was created with the intention of making the Item, ItemList, and InventoryLists classes. These classes are responsible for creating the databases when the controller calls for them to be created, which means loading and saving to CSV files. They are also responsible for keeping the item lists and the items within in order and able to be retrieved when they are called for. These classes are then tested out to ensure that the programming was free of bugs before moving on to the next step. All in all, the model is not supposed to know when their classes are called, and should only be responsible for the data storage and extraction.

Next, the controllers, or the Java servlets, are then designed and programmed. These are tricky as I have never worked with servlets before, but got somewhat of an understanding after some tinkering. What the servlets do is basically obtain the inputs or requests from the webpages, and then feed the data to the model to store, or to extract data from the database through the model. The data is then passed on to be shown on the webpage. The main problem I had with this portion is that I could not find a way for one servlet to pass data to multiple webpages, hence I had to create multiple servlets, each for an item list. The controllers are there to determine what information needs to be obtained from the model and what needs to be saved to the database, and what can be displayed on the view. Hence, these are tested in conjunction with the model to ensure that there are no bugs before developing the view of the application.

Then, the view of the app, or the Java Server Pages (JSP), is then designed. The view of the app represents what the user sees when they are using it. The main function of this portion is that when the user inputs a text or presses a button, the controller has to work accordingly to save the input, and update the JSP to show the user that their action has been interpreted, and there is a response. As this program is not focused on the appearance of the webpage, the program focuses on the functionality and simply ensures that it is working properly. After this stage, the MVC structure is complete. Full testing is then done on the entire web application to check for any remaining bugs. Any extensions to the program are then considered at this point, after a basic application is up and running.

Evaluation

Overall, the design process has been rather challenging as I have not programmed a full web application using the MVC model as such. I felt that the design of the classes could have been done better, and proper interfaces could probably have been used for the servlets. However, for most of the classes, I feel like there is an appropriate level of cohesiveness and there is a good use of OO design throughout.

While there may have been some bumps in the quality of the programming, all of the requirements have been met by the program. This warrants some merit in itself, and I think that this is a really good learning experience.