**Section 1**

Launching the General List Manager takes the user to the home page, where the user can navigate the website using the navigation bar. The navigation bar has the following options:

1. Home

The home page is the landing page for when the user launches the program.

1. View All Lists

The user can ‘View All Data’, in which they can see all the lists, the items contained within the list and the data held within an item: name, link, text, URL and image.

They are also able to see all the lists as well as search for a list. The ‘List Editor’ allows the user to add a list, delete one or more lists and change the name of a list. Clicking on a list will display the lists contents.

Viewing an individual list enables the user to add an item to the list, delete and item from the list and view the items data.

Viewing an individual item enables the user to rename the item or delete it from the list, link the item to a list, change the text of the item, change the URL of the item as well as upload, download and clear the item’s file.

1. Search

User can search for an item in one or more lists by selecting the lists they wish to search in. The results of the search are displayed in a table where the user can visit the item itself, or the list the item belongs to.

**Section 3**

Good OO design has been implemented using the ‘Factory Design Pattern’ which encapsulates creating the model and provides flexibility to reintroduce the Model with no impact on the program.

‘Model.java’ handles all the data and I have decided to implement my program such that the data is saved as directories. The program refers to a ‘data’ folder and any folders acts as a list with the name of the folder being the list name. Each list folder may contain folders which may contain folders with the name being the item name. Each item folder contains one of the following; file folder, ‘text.txt’, ‘url.txt’ and ‘link.txt’. The file folder contains the file linked to the item; image, document, etc. The .txt files contains the information linked to that item.

Every action is directly preformed on the ‘data’ directory and retrieving any information is read directly from the directory as well. This is to ensure ‘Requirement 6’, such that any changes to the data are automatically saved.

I had decided to save my data as the following as creating a Java class to hold all the lists and the items within the list as well as the properties of the item could lead to inconsistencies between the data saved and the data within the Java class. Directly modifying changes and reading from the directory would create no inconsistencies as it is being directly read from the data and passed along.

I have also covered the case where the directory/file is missing during the program running. For example, when viewing all the lists, if one of the list directories were to be deleted after the list was displayed on the front-end, the list will be re-created and this is the same for other cases; item directory has been deleted, item’s property file is missing, etc.

Having decided to decompose the problem, I had decided to make each servlet responsible for one action and would redirect to another servlet to initiate a webpage, if needed. For example, adding a list would follow the path: ‘ViewAllLists.jsp’ -> ‘AddListServlet.java’ -> ‘ViewAllListsServlet.java’ -> ‘ViewAllLists.jsp’. This would mean I could reuse classes instead of having to make unique classes for each action and having segments of duplicated code. This is also a good example of OO design as it avoids coupling between two functionalities, so changing one functionality may break the coupled functionality, which requires additional testing to avoid errors.

I have also tried to follow OO design in programming the design for interface rather than implementation. This has been carried out by using interface type on variables, return types of a method or argument type of methods.

I had decided to design the program by breaking down the system into smaller parts and working through related tasks. I decided to break down the website into the following webpages, where each webpage carried out various actions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Navigation Bar** | **Viewing all lists** | **Viewing a list** | **Viewing an item** | **Search for an item** |
| View all lists | Add a list | Add an item | Delete item | Select lists to search |
| Search for an item | View a list | View an item | Set item name | Search for matching queries |
|  | Delete a list | Delete an item | Set item link |  |
|  | Change list name |  | Set item text |  |
|  |  |  | Set item URL |  |
|  |  |  | Set item image |  |

The navigation bar has been added to every webpage to act as a link to the main pages for easier accessibility. ‘viewAllLists.jsp’ and its corresponding servlet requires no input and only needs to be redirected – this is the same for ‘search.jsp’ and its corresponding servlet. As the navigation bar is a separate entity from whatever page it is on, it has been kept as a separate file and has been included to prevent duplicated segments of code. The same has been done for ‘meta.jsp’ and ‘footer.jsp’. This is an example of good OOP design for future maintenance.

Viewing all the lists can be accessed via the navigation bar, viewing a list can be accessed from viewing all the lists, viewing an item can be accessed from viewing a list and changing an item’s properties can be accessed from viewing an item, and so forth.

**Section 2**

