**Wire Frame**

The application will present to the user a tab based interface, the tabs are arranged vertically along the left hand margin.

These tabs are generated programmatically at runtime. Each user type will be presented with a different set of tabs. This provides a view based interface through which users can interact with the system, whilst limiting the access privileges associated with each user type.

**Tab Page**

*Basic wireframe*

**Log On Form**

Upon start up, or after logging out, the user shall be presented with a log on form. Users shall enter their user name and password.

The system does not store passwords directly for security reasons. Instead the system takes the password and uses it to encrypt the username. This produces a cipher text version of the username. The cipher text is stored instead of the raw password.

Once a log on attempt has been issued the system will take the supplied username as a key, then will use that key to query the list of users. If the user is not found the log on will fail. If the user is found, but the cipher texts do not match, then the log on will fail.

Users will have at most three attempts to log on, after which the system will shut itself down.

**User Name**

**Field Value**

**Field Value**

**Pass Word**

**Cancel**

**Submit**

*Log On Form*

**List Tab Pages**

Each entity type will be associated with its own tab. Selecting a tab will result in a list being displayed to the user. The user will have access to a search bar to limit the number of list items presented.

Each list item will have an object associated with it. Clicking on a list item will present a read only form displaying all the items details.

Managers and Staff may have access to an ‘Add New’ or ‘Delete’ button.

**Search Bar**

**List View**

**List Item**

**List Item**

**List Item**

**Delete**

**Add**

*List Tab Page*

**View Item Tab Page**

Once an item has been selected, the user shall be presented with a view form. The view and edit forms differ only in the fact that view forms are read only.

Delete item forms are much the same as View item forms. They are read only and serve the purpose of confirming to the user that they have selected the correct item for deletion.

If the user has sufficient privileges an ‘Edit’ button may be presented. Clicking on this button will allow the user to update the values of these fields.

Some items may have a different action, such as reserve copy of game, return game. Such action buttons will appear at the bottom of the form.

**Field Value**

**Field Name**

**Field Value**

**Field Name**

**Field Value**

**Field Name**

**Cancel**

**Edit**

*View Item Tab Page*

**Edit Item Tab Page**

The user shall be presented with a view form. The view and edit forms differ only in the fact that view forms are read only.

Each field is associated with an appropriate form control, allowing the user to enter the new details. Some fields shall remain read only, such as primary key fields.

Create forms are much the same as Edit forms. Create forms are initially blank and allow for creation of a new item from scratch.

Submitting a change will result in the new values being committed to the item, cancelling the edit will result in the previous values remaining unaltered.

**Field Value**

**Field Name**

**Field Value**

**Field Name**

**Field Value**

**Field Name**

**Cancel**

**Submit**

*Edit Item Tab Page*

**File IO**

We purposely avoid committing ever little change instantly to file. This approach will enforce consistency between the persistent data in the file, and the dynamic data in memory.

File IO is performed at start up, all the required data is brought into memory from file.

During runtime the modifications are performed in memory.

At shut down, the system will commit the changes to file by creating the files anew.

**Run Time**

**Clean Up**

**Start Up**

**To File**

**In Memory**

**From File**

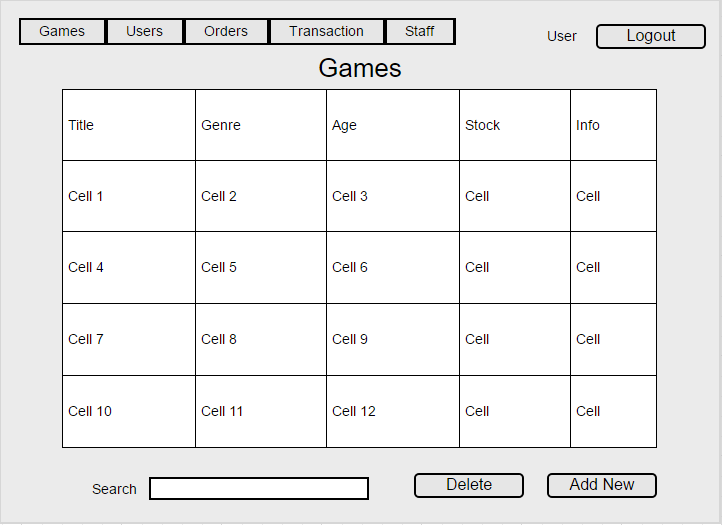
*Data Flow*

**The Context**

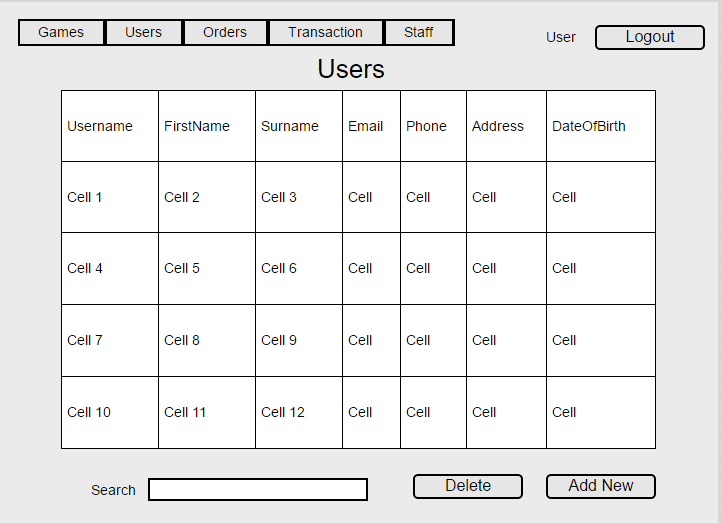
The Context is the sub system that manages the user list, and games list. The context is the master state of the system at its highest level. The context allows us to have a single global list of Users, and a single global list of Games.

At start-up the Users and Games are pulled into memory from file. All manipulation to these lists are performed in memory. Then when it is time to shut down, the entire list of Users and Games are committed to file, making the modifications persistent across sequential runs of execution.

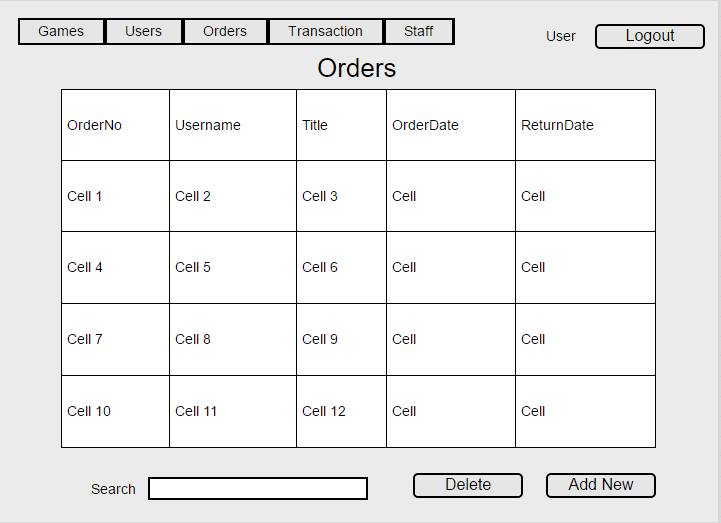
**The Games Page**



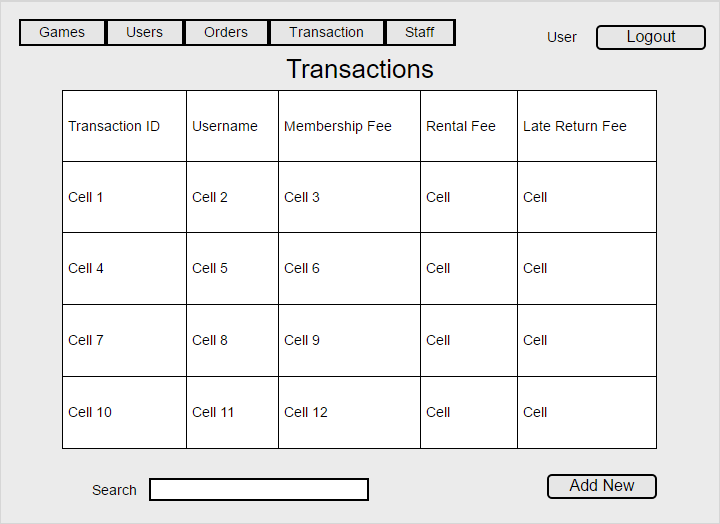
**The Users Page**

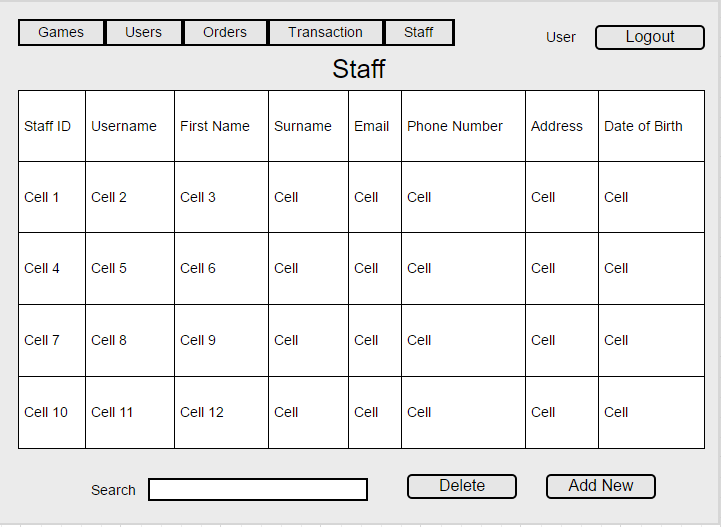


**The Orders Page**



**The Transactions Page**





**Staff Page**