Project Alexandria

Ryan Gannon - 05456983

Proposal Friday, 08 February 2008

Contents

Team

Ryan Gannon (Project Proposer), Thomas Higgins, Amal Tukur Modibbo

Introduction

In the 3rd Century BCE the greatest library in the ancient would was built, the Library of Alexandria. It survived for centuries before finally burnt to the ground.

Figure 1 The Ancient Library of Alexandria

Project Alexandria is a php web-based Library System based on Delicious Library. “Delicious Library is a media cataloguing application for Mac OS X, developed by Delicious Monster. The software allows users to manage their collections of books... by entering [an ISBN]” (Wikipedia) it also works as a Library system with “borrowers”, “lending” and monitoring “out” and “late” books. Alexandria will provide all of these features and more as will be discussed below.

The idea is nothing original with similar software such as Collectorz.com’s “Book Collector” and Libra for Windows and Perpustakaan for Linux.

So why is this different?

The key point with the above mentioned software is that they are all platform-specific and are all desktop run. Alexandria is web-based this means it allows for;

- Cross platform compatibility (internet based software typically does not require you to have specially Windows, Mac OS, etc.)
- Public access through the internet
    - This would allow users to request books online

- This would also mean that the librarian can operate anywhere there is internet access rather than being tied to one machine or network.

But wait aren't there web-based book/library-management systems?

This is true, but like the above they are not inexpensive (most web-based library-systems have a recurring charge). Alexandria on the other hand will be;

- free
- open-source

Why is open-source important?

Well one of the reasons for the recurring charge is that these online library-systems are based on the creating company's servers, having this software be php-based and open-source allows people to host the library themselves. It will also allow them to edit it to their needs.

One of Alexandria's key features is that it will be designed with plug-ins in mind so that users will be able to add features themselves and distribute them.

Overview

Frontend

Alexandria's frontend will be a combination of XHTML 1.1 and php, styled with CSS. It will consist of;

- a Homepage **(Over-all System)**
- navigation which will appear on all pages **(Over-all System)**
- a login/out facility which can appear on all pages **(User-System)**
- a Registration page **(User-System)**
- a Search and Browse page to allow users to find books **(Book-System)**
- an individual Profile page for each book featuring a variety of details (title, publisher, author, ISBN, description, availability, etc.). This is also where users can request books **(User-System)**. **(Book-System)**
- a Profile page for each user with their details (username, email, books they've read, the rating they've given them. In the case where it is your own profile you are viewing you will see the books you currently have out and their return date as well as books you have yet to rate. And give you your profile address to distribute to friends) **(User-System)** **(Book-System)**
- a Rating Page to allow users to rate and review books they've read **(User-System)** **(Book-System)**
- an Edit Profile page for users to edit their details **(User-System)**
- an Admin Control Panel. **(Over-all System)** (Accessible only to Admins **(User-System)**) This would have;
    - a User Edit and Ban page **(User-System)**
    - a Book Add, Edit and Delete page **(Book-System)**
    - a facility to monitor the state of "requested" or "on-load" books **(Book-System)**
    - a page to control the over functions of the website **(Over-all System)**
    - a page to control the plugins installed **(Over-all System)**
- a "Friends" Page **(User-System)** **(Book-System)**

Note: It should be taken that all parts of the site have some part that comes under the remit of Over-all System. These interactions will be explained in detail below.

In addition it is intended that the site be dynamic and step out of the typical flat representation of information taking inspiration from MagicInk (Victor).

Backend

Alexandria will use php and MySQL to render and manage all data (incl. users, books, "Friend" subscriptions etc.). It will also be designed in a modular fashion; users will be able to scale it from a simple vanity list of their personal collection online to a fully fledged library system.

Book-System

Introduction

Delicious Library uses the ISBN (with optional manual entry) and the Amazon api to retrieve data about the book and add it automatically. Alexandria would expand beyond this by use of isbndb.com.

isbndb.com is a online database working to create a catalogue of all books ever published, as opposed to Amazon as they would be limited to books chiefly published within they're time. This would of course mean that the software is dependent on these sites but it also means that the program itself can be very slim.

Implementation

All functions would be placed in a single file and `included` in the following pages. Functions that affect the database would be written as SQL queries.

- **Book Add, Edit and Delete page:**

Adding a Book:
A php form would give an admin inputs on the book Add, Edit and Delete page. The first would be for the ISBN, this would have a "Retrieve" button. An ISBN entered here would be used to find the relevant data to that book first on Amazon using its api, should the result be *null* then the program would access isbndb.com and use its api to retive the data. This process would be modularised so that access to other api's could be added later. That is to say there would be a `getFromAmazon(isbn)` function in a list of functions ending with `getFromIsbndb(isbn)`.

Should no information be found at any source the user would be advised to enter the data in manually in the remaining inputs. At the end of these would be an "Add" button that would enter the data into a row of a table ("books") of a database created by the user during installation and use the ISBN as the key.

Deleting a Book:
Part of the book Add, Edit and Delete page would be a table of all existing book entries. This table would include a column of "Delete" links. Each link would include the ISBN and using that would call the `deleteBook(isbn)` function that would delete the row of the "books" table where that ISBN is the key. The columns' headers would be linked in such a fashion that clicking on them would order the table be that column in ascending order this would be done by a `orderBy("header")` function.

Editing a Book:
Part of the book Add, Edit and Delete page would be a table of all existing book entries. This table would include a column of "Edit" links. Each link would include the ISBN and using that would call the `editBook(isbn)` function that would produce a form identical to the Add Form (see above). The data for that book would be automatically entered into the various inputs (except the ISBN

which would appear as static text) and be therefore editable. An "Edit" button at the end would take the data in these inputs and update the row of the "books" table where that ISBN is the key.

- **Search and Browse page**

  Browse:
  The Search and Browse page would contain a table of all existing books in the "books" table. Each books title would be linked (the links would contain the book's ISBN) to the book's individual Profile page. The columns' headers would be linked in such a fashion that clicking on them would order the table be that column in ascending order this would be done by a `orderBy("header")` function.

  Search:
  The Search facility would consist of an input of a search term, a drop-down menu which would contain the search category (title, ISBN, author, etc.) and a "Search" button. This would re-render the table so that only entries that matched the search criteria would appear. This would be a function where an SQL query would be used to produce the new table.

- **Individual Profile page for each book**

  Details:
  This part of the page would produce all the details of the book in a well formatted manner. This would include the Title, Author, ISBN, Description, etc. Where the information is not available the page will not render anything, not even the header of that piece of information. As the number of pending requests (which would simply be the number of rows in the "booksRequested" table where the ISBN of that book matched) for a book would be included and its availability (whether it is on-loan or not) (it would be displayed as "on-loan" if an entry exists in the "booksOnLoan" table with a matching ISBN otherwise it will display "Available").

  Also included will be the contents of "booksReview" where the book's ISBN matches that of the rows in that table.

  Request:
  This will appear as a link on the page. It will contain the username of the requesting user and the ISBN of the book. The function `requestBook(username, isbn)` will add these two details to the "booksRequested" table.

- **Facility to monitor the state of "requested" or "on-load" books**

  Requested:
  This will appear as part of the Admin Control Panel. It will be a table which displayed the details of the "booksRequested" table as the username of the requester and the title of the book requested. In addition to these two columns there will also be a column of "Transfer" links that will move the details from "booksRequested" to "booksOnLoan" and a column of "Delete" links that will remove the requests without moving them to "booksOnLoan" table.

  On-Loan:
  This will appear as part of the Admin Control Panel. It will be a table which displayed the details of the "booksOnLoan" table as the username of the user who has the book on-loan (usernames will be linked to the user's profile) and the title of the book. In addition to these two columns there will also be a column of date on which the book was loaned (i.e. the date when the book was transferred from "booksRequested" to "booksOnLoan"), column of "Retuned" links (which will delete this entry and add the details to the "booksRetuned" table) and a column of "Delete" links.

User-System

Introduction

Delicious will allow you to add users who can be loaned books. Alexandria allows users to register online, edit their details, request and be loaned books. This will be done by use of php (including Sessions and Cookies) and MySQL to store details.

Note: It is advised that you read the *Book-System* section first as some element are implemented in a similar manner.

Implementation

- **Registration page**

  Will consist of inputs for username, password, confirm password, email and a "Register" button. The username will be checked to see if there is already an instance of it in the "users" table of the database if so then the user will be advised to use a different one, if not then the passwords will be check to confirm that they are indeed the same if not then the user will be told of this.

  All going well the users details will be added to the database, along with the date and time they joined and will be given the default level and they can login.

- **Login/out facility**

  The following will be accomplished using php's Sessions and Cookies

  If a user is logged-out they will see a form made up of an input for the username, an input for the password and a "Login" button. The username and password will be checked to see if they match and that the user does not appear on the "usersBanned" table. All going well they will be logged-in. When logged-in they will simply see their username in static text as well as all features available to them at their level and a Logout link.

- **Profile page for each user**

  This will feature all details (except password) that exist in the "users" table for the user in question.

  It will also grab all entries in the "booksReview" and "booksReturned" tables of the database.

  Also if a pair (username, ISBN) appear in the "booksReturned" table and not in the "booksReview" table a message will appear asking the user to please rate and/or review the book (This could also appear elsewhere this is why like most of the software it will be a function).

- **Edit Profile**

  This will appear as a series of inputs which will automatically grab the data for the logged-in user and therefore allow them to edit it. The exception being the username which will appear as static text and the password input will appear blank and be followed by a "Confirm Password" input. At the end will be an "Edit" button. The data in these inputs will be used to update the row of the "users" table that has that username as its key.

- **User Edit and Ban page**

  This page will have a table of all users featuring columns for their username, email and level. As well as these there will be columns of "Level" links that will allow the admin to change the level of that user and a "Ban" links column which will move the user's details from the "users" table to the

"usersBanned" table. Finally there will be a "Delete" links column which will remove the user from the database.

Over-All System

Introduction

The "Over-All" System refers to the layout, design and scalability (switching on and off functions, excluding data from public view, etc.) of the software as a website. Chiefly it operates from the Admin Control Panel and interacts with the rest of the site though styling, plugins and the universal elements like navigation.

Implementation

- **Design and Layout**

  The design and layout will be handled using CSS. The Book-System Designer and User-System Designer will write XHTML 1.1 compliant code (without tables) labelling every element with classes and `id`s and making good use of `div`s. The will coordinate with the Over-All System designer giving regular updates on changes to their code to allow the Over-All System designer to create user-friendly layout.

- **Control the over functions of the website**

  There will be a page in the Admin Centre that will produce a table of all functions. This will also include a column of links to activate or deactivate these functions.

  The "functionsRegister" table will act as a register of functions such as Registration, Requisition and Loaning, Public Viewable of Defined Pages, etc. It will contain the function and a Boolean which will dictate if the function can be fulfilled or not (conditionals would be used on the relevant pages and error messages would show should someone try to use them).

- **Control the plugins installed**

  One of the great goals of the Project is to allow plugins so that users can develop safe modifications and distribute them.

  There will be a page in the Admin Centre that will produce a table of all plugins available. There will be a column to (De)Activate them (essentially (un)installing them). The Process by which this is done will be dictated in the plugin themselves (a standard will be developed so this is handled in the same way every time).

- **Homepage**

  The Homepage will be the greatest expression of the control over functions as it will be by default chiefly made up of them and so its appearance will be greatly dictated by the "Control the over functions" page. These functions will include, "The Last x-number of Books Added", "Top x-number of Books Requested", etc. The rest will be user added.

- **Navigation**

  The appearance of the navigation bar will have to be dictated by where the user in logged in or not, their level if they are and whether the function relevant to the link is active. Conditionals will be used for this purpose and it will require the Over-all System designer to liaise with the User-System designer specifically.

Collaborative-Filtering System

Introduction

Aside from the freedom it being web-based, this is the major step-up from Delicious. It will be based on other users that request similar books and their rating of them. User would be encouraged to rate their books when they return them by a message that would appear on the site when they login. Also there would be a form so that they could optionally review books.

In addition to this users would also be able to "subscribe" to other users in a similar manner to LiveJournal's Friends Page, but instead of journal entries you'll be able to see what books your "Friends" are requesting and their rating and review of them.

Implementation

Collaborative-Filtering System:

Most likely Pearson and Mean-Squared-Difference are the algorithms that will be used and the processing will be done by a regular cron job producing a table which will contain recommendations based similar tastes in authors, genres, etc. and the predictions will be made on the ratings given.

Friends Page:

Although not strictly Collaborative-Filtering the Friends Page is intended to help users choose books by looking at their "Friends'" reviews and ratings. Users can distribute their profile addresses freely to whomever they want.

Friendships would be stored in a database where each user would have a row where their username was the key and a column of usernames were their friends.

There would also be a link on this page that would produce a list of the user's Friends and allow them to remove them as a friend.

Comparison to earlier project

An earlier project similar to this one was written by me (Ryan Gannon) during the 2007 winter break. It was a slapped together implementation intended only for use by UCD's Pagan Society for their newly founded library. The reason I did not wait to implement it till later was the fact that the library was to be open on the 28th January and a software like this was not available. An apparent similarity is the list of feature when compared to the below basic requirement, this is because these are requirement that I know through experience must be fulfilled in order for the software to work on a basic level. This doesn't mean however that these goals have been reached. As I said this was a "slapped together implementation", what follows is a detailed comparison and reasons why the pervious implementation not only irrelevant it is almost totally unsuitable for this project.

Previous Project	This Project
Installs	This one is moot. Any changes made to the software at all will require the installation to be changed. This project's features would require massive changes.
Adds books	In one respect this will stay the same as to retrieve data from isbndb.com is the same no matter who/what does it, but as this project has Amazon as its primary source and will be built to add other source with more data than isbndb.com has to offer this like the install would require massive changes.
Deletes books	Now this would mostly stay the same if it were to be kept but again this is because there is only one way of

	doing it.
Edits books	Since Add and Edit are almost the same the same points apply here
Searches	A more robust and scalable search facility will be built for Alexandria
User Accounts	Here's where there are some big difference. The user system in the old project was entirely copied, this one will be done specifically for this software, will be written by some else and will manage much more data because of the CF-System, "Friends" and greater ability to manage your account.
you have to be an admin to edit/add books	
users can request books	
A request and "on-loan" management system	
delete requests or transfer them to on-loan	
delete on-loan books when they're returned	
The books have profiles	Again as with Add and Edit the amount of data will make the old code pointless.
Better styling (no tables)	The styling will be handled by someone else and like installation any changes in the software means changes in the styling.

The most significant consequence of this old code is that I have the experience with php and MySQL in this manner to know what has to be done in the basic form of this project. Also the basic requirement include the addition of;

- Use a basic collaborative filtering system to produce recommendations for users based on the books they rent/request.
- Send email alerts to the librarian and to any persons that have an impending return date.
- Have a "Friends" page

This means that the two project are significantly different.

Real-world Application

It can be hard to predict how a piece of software will be used, a library system like this can be particularly difficult as some many people are involved and so much of it take place in the real-world. This project however has two real-world applications ready UCD's Pagan Society and UCD's Science-Fiction and Fantasy Society. Two very different group of two very different sizes with very different sized libraries.

This will allow us during early development to handle problem that we cannot predict until they happen in the real world.

Technology

PHP, MySQL, XHTML 1.1, CSS, cron, possibly a barcode scanner.

Roles

Book-System designer (Collaborative-Filtering System designer No. 1)

who will build the basic book system (adding, deleting, displaying, etc.) including its data bases, forms and basic interface. Also they will have to create the install script for those tables their code relies on. As well as that they will participate in creating the collaborative filtering system. Throughout they will liaise with the Over-All System designer and, where user data is required, the User-System designer. See also Overview>>Backend>>Book-System

User-System designer (Collaborative-Filtering System designer No. 2)

who will build the user system (registration, login/out, profiles, deletion and banning including its data bases, forms and basic interface. Also they will have to create the install script for those tables their code relies on. As well as that they will participate in creating the collaborative filtering system. Throughout they will liaise with the Over-All System designer and, where book data is required, the Book-System designer. Overview>>Backend>>User-System

Over-all System designer (incl. Layout and Art-Work)

who will receive regular updates on the state of the frontend from the Book and User System designers and create a layout and all art-work. They will be responsible for the interconnected and general parts of the software as a website. They will also be responsible for getting feedback for in relation to the layout and general layout of the site and putting it to use to make it as user-friendly as possible straight out of the box. Overview>>Backend>>Over-all System

Requirements

The Project Must…

- Have a working installer that would create all necessary tables for both users and books
- Adds books (through the ISBN number and optionally by entering details manually)
- Delete/Edit books (you have to be an admin to edit/add books)
- A request and "on-loan" management system
    - Delete requests or transfer them to on-loan
    - Delete on-loan books when they're returned
- Have a search page for users and orders the search results (e.g. by ISBN, Author, etc.)
- User Accounts (by which users can request books and cancel requests)
- The books have individual profiles
- Use a basic collaborative filtering system to produce recommendations for users based on the books they rent/request.
- Send email alerts to the librarian and to any persons that have an impending return date.
- Have a "Friends" page
- Have a dynamic results and description pages to allow users to better judge the book before requesting.

Secondary requirements include…

- Have a WordPress bridge, this would use the WordPress' users, have reports from the library to the WordPress homepage/sidebar and if you login to WordPress you would be logged into the library.
- Use a barcode reader to make entering new books even easier.
- Send the librarian alerts individually or optionally in digest form.
- Use of open-source book websites (e.g. WikiSource, Project Gutenberg, etc.) to add virtual books.
- Import from Delicious Library
- Handling for multiple copies of the same book
- A facility to allow readers to easily create and monitor lists of to-read and read books in the library and to possibly expand this to allow them to do the same with open-source ebooks and online articles.

Things that are not goals of the project include:

- To have a superb collaborative filtering system.
- To catalogue anything other than books.

- Will not be designed with the intent to sell anything, though traffic back to Amazon is.
- Will not read ISBN from captured images.
- Fulfil any need for a Web2.0 application

Bibliography

Wikipedia. (n.d.). *Delicious Library*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Delicious_library