Project Alexandria Ryan Gannon - 05456983

Proposal

Contents

[Team 1](#_Toc189742621)

[Introduction 1](#_Toc189742622)

[So why is this different? 2](#_Toc189742623)

[But wait aren’t there web-based book/library-management systems? 2](#_Toc189742624)

[Why is open-source important? 2](#_Toc189742625)

[Overview 2](#_Toc189742626)

[Frontend 2](#_Toc189742627)

[Backend 3](#_Toc189742628)

[Book-System 3](#_Toc189742629)

[User-System 3](#_Toc189742630)

[Collaborative-Filtering System 3](#_Toc189742631)

[Technology 3](#_Toc189742632)

[Roles 3](#_Toc189742633)

[Book System designer 3](#_Toc189742634)

[User System designer 3](#_Toc189742635)

[Layout and Art-Work 4](#_Toc189742636)

[Requirements 4](#_Toc189742637)

[The Project Must... 4](#_Toc189742638)

[Secondary requirements include... 4](#_Toc189742639)

[Things that are not goals of the project include: 4](#_Toc189742640)

[Bibliography 4](#_Toc189742641)

# Team

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

# Introduction

Figure The Ancient Library of Alexandria

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.

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]” 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 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.) **(User-System) (Book-System)**
* an Edit Profile page for users to edit their details **(User-System)**
* an Admin Control Panel. **(Over-all System) (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.

## 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.

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 column’s 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”).

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 how has the book on-loan 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”) and a column of “Delete” links.

### User-System

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.

### Collaborative-Filtering System

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.

# Technology

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

# Roles

Book-System designer

who will build the basic book system (adding, deleting, displaying, etc.)

User-System designer

who will build the user system (registration, login/out, profiles, deletion and banning

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

# 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.

## 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 better than basic collaborative filtering system algorithms to produce recommendations.
* 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.

## 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.

# Bibliography

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