



TomeTracker: Improving Personal Library Use

Brendan Ribera - Informatics Program - University of Washington

Abstract

Computer-based search systems could be very helpful when seeking information within one's library. However, current systems limit what information is shown about any given book, and don't allow the search scope to be restricted to a meaningful set of books. TomeTracker is a new answer to an old question: How fully can you use your books?

This answer takes the form of a website on which users can create and manage a digital library, helping them to classify and organize books. It also provides the ability to perform full text search within the given user's library. Results are displayed with very enlightening heat maps; focusing on a particular book can also show snippets of text that match search criteria. Additionally, the specification calls for a social aspect, in which users could recommend books to friends and participate in book-centered discussions. This prototype is a proof-of-concept, demonstrating a potential solution to a real need.

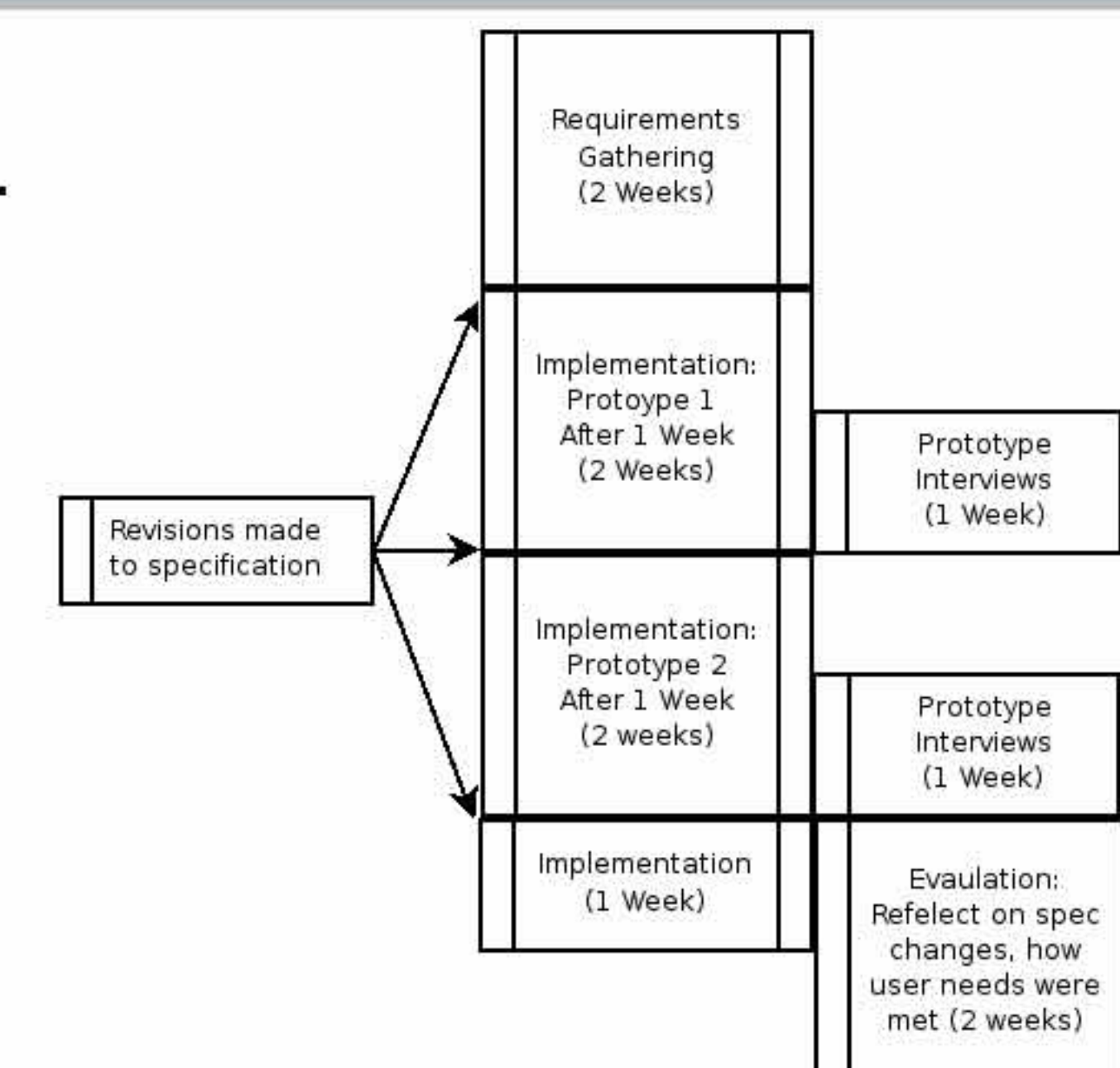
Methodology: Iterative Development

TomeTracker utilized an iterative development cycle. The initial set of requirements was gathered via user interviews. Based upon these initial requirements, a prototype of the system was made.

Iterative development uses prototypes to evaluate the present design and help it evolve into a more useful form. The potential users who assisted during the requirements gathering process also helped evaluate the prototypes. Based on their use of the prototype, the specification was revised multiple times.

This process helps focus the final product on the user's needs, rather than allowing implementation details to cloud or alter those needs.

The image on the right depicts the schedule that was followed during the development process. In actuality, the final prototype interview was conducted informally, and the prototype is the one revised according to the first prototype interview.



Schedule for iterative development

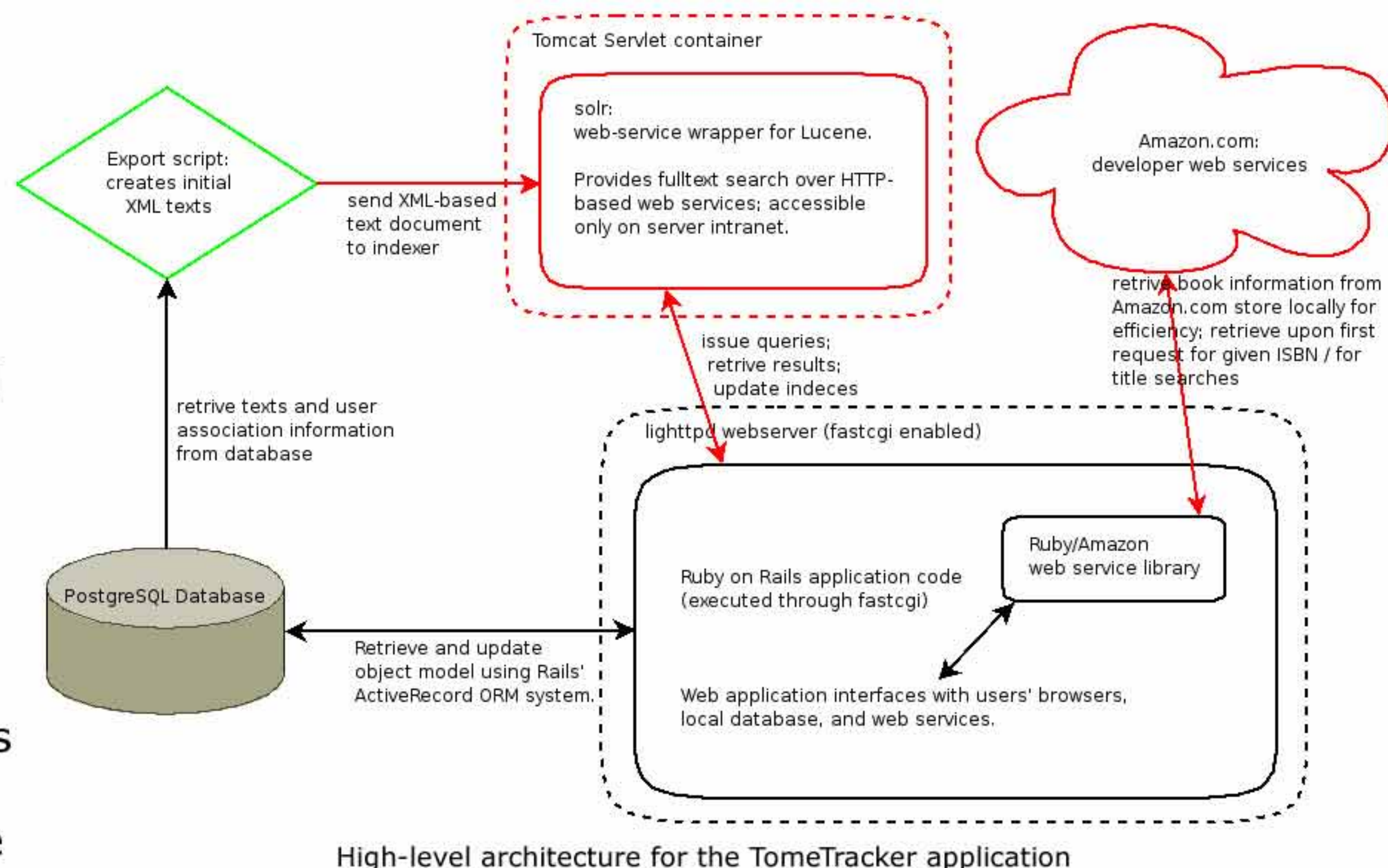
Implementation Process

TomeTracker prototypes were implemented with open source software.

The main application is written in Ruby, using the Rails framework to provide quick changes and easy object-relational mapping.

The prototype system uses 16,000 digital texts from Project Gutenberg. The application uses the Amazon Web Service to provide images and book information in addition to the texts from Project Gutenberg. All of these objects are stored using the PostgreSQL database.

Full text search is implemented using Solr, a project from the Apache Foundation. Solr provides a web service interface to the Lucene search engine, with a simple API for indexing and retrieving documents. The Rails application interfaces with Solr to query the index and update it when users add or remove books from their libraries.



High-level architecture for the TomeTracker application

Final Prototype

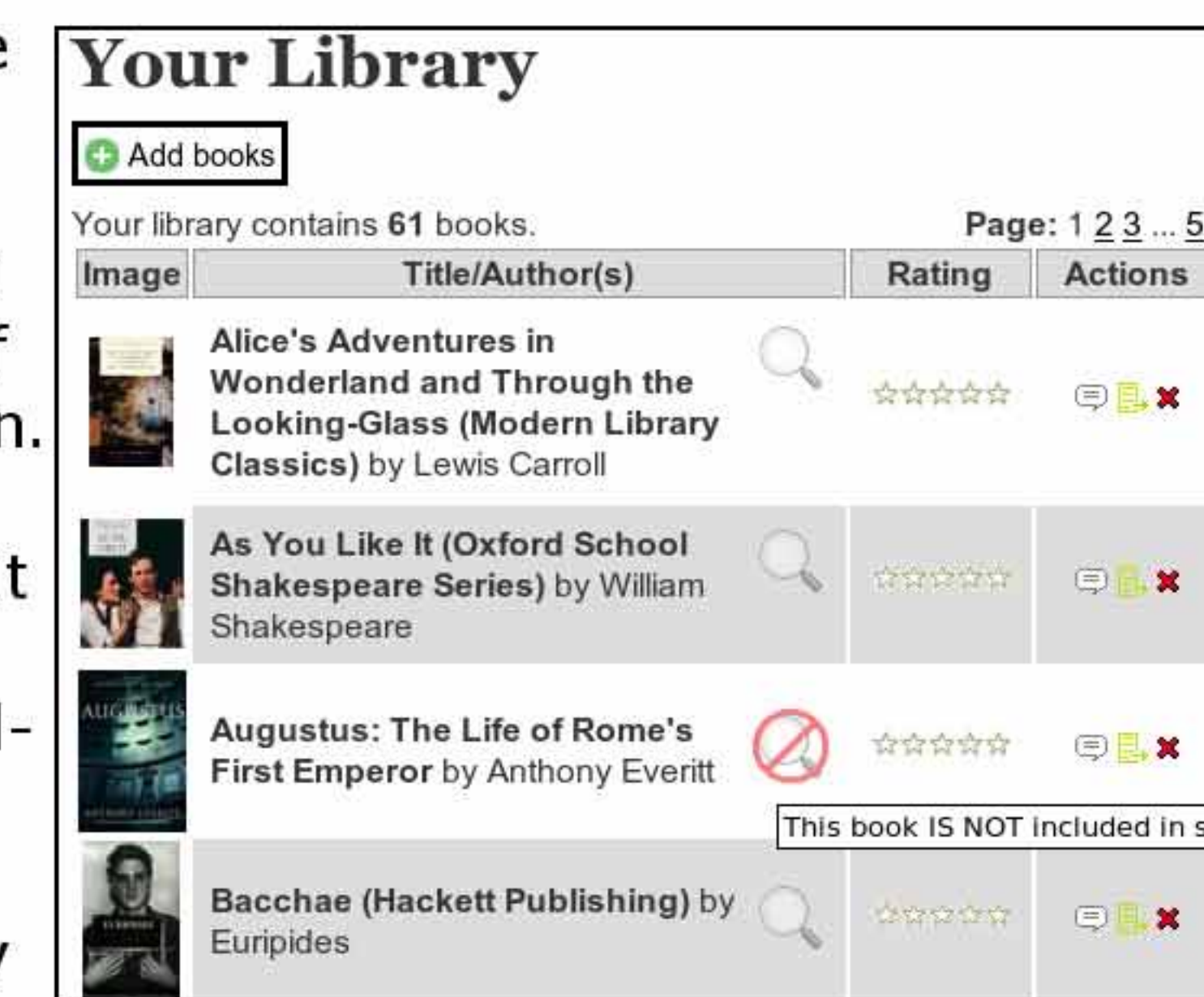
The most tangible results of the TomeTracker project are the working prototypes that were created for testing.

On the right are pictured some examples of the most recent version of this prototype. The top picture shows the view of a user's library, with a few of the 61 books contained therein.

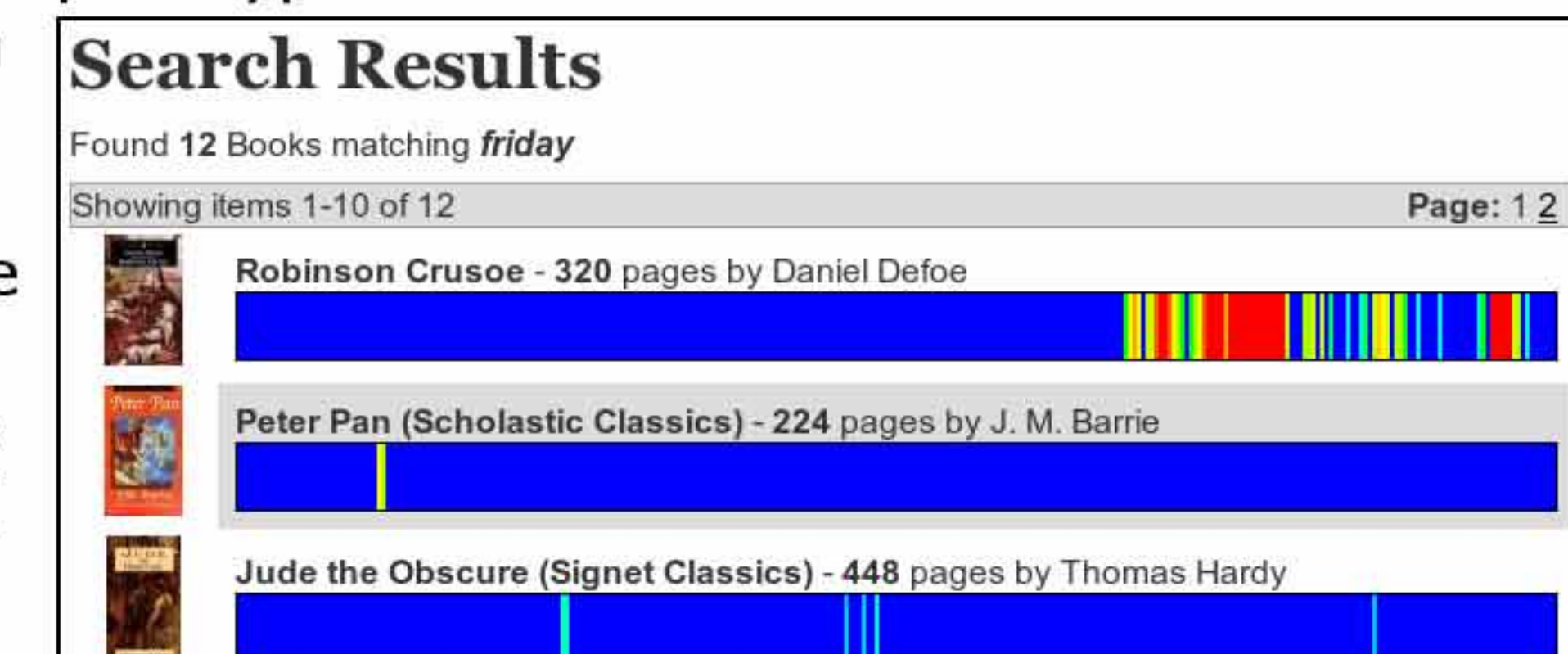
The bottom picture shows an example of the heat map result display, ostensibly the most notable feature of TomeTracker. This particular picture demonstrates how useful such a display can be. The search query is for "friday", and the first result is from Daniel Defoe's "Robinson Crusoe." It shows precisely the moment in the book when the character Friday makes his first appearance, and where he is most talked about.

Users were very impressed with all aspects of the prototype.

Social networking features are part of the design specification, but were not implemented fully by the time the last prototype was completed. In addition to these missing features, users indicate that the 16,000 free books provided at present are too restrictive. In order for this prototype to be useful in a real-world situation, it would need to include copyrighted texts that are currently unavailable.



Library view from the current prototype



Example of heat map styled search results