James Stoddard
Yu-Cheng Chen
Andy Fu



# reBook

## Idea

As this quarter began, many college students engaged in the familiar ritual of going to the bookstore. Often, students shell out hundreds of dollars for books that may only be used for a few months. Afterwards, they have the “privilege” of selling them back at an 80% loss. There should be a better way to obtain value from these discarded books. Imagine a system that allows students to easily trade their unused textbooks for needed ones. This is the vision of reBook.

reBook is a book trading hub which aims to make the process of locating your next book fast, free, and automatic. Simply list the books you own, the books you need, your campus and your schedule. The system will automatically propose trades between individuals on campus at agreed upon times. Significantly, reBook will be designed to arrange trades between up to three people. No money changes hands, just books for other books.

A compelling part of this project is that it serves a niche which currently has no obvious leader. Three things set the service apart: The common campus of group members is used to eliminate the hassle of arranged meetings. Trades between multiple parties are proposed to greatly increase the odds of getting a particular book. Lastly, no money is allowed in the system, which eliminates the haggling involved with many alternatives. This combination of features forms a user experience that is much more convenient than alternatives such as eBay, craigslist or freecycle. Ideally this enables students to buy books for their first quarter on campus and then never spend another cent on them. As far as we can tell, there are no direct competitors for this system.

## Software Architecture

The most exciting aspect of this project is the challenge of developing an efficient matching algorithm. People may have the book you are looking for, but not be on campus when you are or have it in a condition that you want. reBook will find the closest match. Alternatively, someone may have your book, but not need anything that you currently have. In this case, the system will try to identify multi-party swaps: everyone meets and you give a book to a second party, they give a book to a third party, and that third party gives you the book that you need. This greatly increases the number of viable trades, but a naïve implementation here would quickly lead to horrible performance as the application scales.

At a high level, this application will be delivered over the web using a design which is responsive for different types of devices (ie mobile vs desktop). This implies that the frontend will be based on HTML, CSS and JavaScript. JavaScript libraries such as JQuery will be employed to reduce the amount of boilerplate code required for the client side. Much of the functionality will be done in asynchronous calls, making it feel more like an application than a web page.

For the back end, we propose Ruby on Rails. Rails simplifies much of the object-database mapping, allowing a team to focus on the application logic as opposed to the drudgery of database manipulation. We are open to alternative technologies, however, and may reconsider if the majority of group members have different experience. We have settled on PostgreSQL as

the database management system. The features for our application map pretty closely to the data which will be stored, so a quick overview of our data helps to illustrate our goals. Proposed major tables include: Books, Book Inventories, Locations, Trades, Users and User Feedback.

Our application will rely on a third party APIs to retrieve information on individual books. Building an entire database of books from the ground up would be out of scale for such a project, and many free resources exist which provide this information. A user will enter an ISBN or title in the web app, which will result in a call to our back end. From there, the call will be delegated to the third party JSON API (likely Google books), and if found, results will be returned for the user to add to their inventories.

Finally, as a stretch goal, we are toying with the idea of porting the completed web app to Android using one of a few third party frameworks. PhoneGap, for instance, supports porting an HTML web-app to native Android with minimal difficulty for developers.

**Challenges and Risks**

There are a few potential problems which come to mind, but none of them are in any way crippling. As noted earlier, multiple party swaps present both opportunity and challenge. We want to generalize our implementation, but at the same time we have to be very careful in its design. If we very consciously devote time to getting it right above all else, we can avoid being caught by surprise with performance issues in scaling.

Another potential risk is that by its very nature, the application needs a decent number of users to be worthwhile. Like any social application, this becomes a self-enforcing loop, but we need to make a conscious effort to cultivate an initial user base. That could take the form of flyers and word of mouth. Closely related to this issue is that, in development, we will not have actual users. We will need to generate a large testing dataset which closely mirrors what an active community would look like. Ruby has many tricks up its sleeve to help us with data generation, but we will want to devote appropriate time to this task.

Lastly, the teams will be larger than what most of us have encountered up until now. This will make collaboration a challenge, and we need spend some time at the outset developing a suitable workflow. There is the additional concern of ensuring we have the proper distribution of skills. Ruby on rails has been tentatively chosen as the framework, but not every team member may know it. There are many great resources for picking it up quickly, but ideally we would want more experience with this. We welcome more contributors who have some experience with Rails. Experience with JSON APIs, automated web testing and graphic design would also be very helpful. Team members need not know all of these technologies, of course; specialists are quite welcome.
.
(http://railsforzombies.org/ is an excellent Rails primer if you can excuse its silliness, while http://ruby.railstutorial.org/ruby-on-rails-tutorial-book is more comprehensive)