Improving the Textbook Exchange System at MIT

by

Rodrigo Ipince

S.B., Computer Science and Engineering, MIT (2009)
S.B., Mathematics, MIT (2009)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2011

Author .....
Department of Electrical Engineering and Computer Science
February 1, 2011

Certified by .....
Professor David Karger
Professor of Computer Science
Thesis Supervisor

Accepted by .....
Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Improving the Textbook Exchange System at MIT

by

Rodrigo Ipince

Submitted to the Department of Electrical Engineering and Computer Science
on February 1, 2011, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Textbooks have become one of the biggest expenses of a college education. We attempt to solve this problem by providing students with better information and tools to facilitate book exchange. More specifically, we focus on the subproblem of using technology to build an effective local textbook market. This thesis makes three main contributions. First, it integrates a local sales system into BooksPicker, making it the first tool that does price comparison between both local and non-local (online) sources. Second, it introduces automatic pricing strategies that dynamically price books based on current market prices and book condition. Third, it provides an offer prioritization mechanism that uses location and social information to sort competing offers.

Our system was deployed at MIT during the Fall 2010 semester and measured in terms of traffic, local market effectiveness, and through user feedback. Traffic increased three-fold and the number of transactions increased six-fold when compared to each of the two previous semesters. Usage of automatic pricing increased the probability of sale while decreasing the time in market of local offers. We did not obtain enough data to properly evaluate the offer prioritization mechanism.

Thesis Supervisor: Professor David Karger
Title: Professor of Computer Science

Acknowledgments

This thesis would not have been possible without the guidance and support of my friends, family, and mentors. I wish to offer them my deepest gratitude in helping me get through the last stretch of my journey at MIT. In particular, I want to thank:

David Karger, for his invaluable advice and incredible insight that never ceased to amaze me.

Anne Hunter, for always being available to help navigate the crazy waters of the Institute.

My mother, father, and brother, for always believing in me and pushing me forward.

Lina, for her undying support and company, from the very beginning to the very end.

Jonathan and Sinchan, for bringing me into the world of BooksPicker, and with whom it has been a pleasure to work with ever since.

Jose and Gabriel, for keeping me company as we struggled together through so many sleepless nights.

Thank You,

Rodrigo Ipince

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Textbooks have become one of the biggest expenses of a college education. The higher education textbook market in the United States alone is estimated at over $8.2 billion [8]. We believe the market is inefficient and that students often pay more than necessary for their textbooks. This thesis attempts to address this problem by providing students with better information and tools to facilitate book exchange. More specifically, we focus on the subproblem of using technology to build an effective local textbook market.

Our work was built on top of the existing BooksPicker tool, a system that helps MIT students search for cheap bundles of books from multiple online vendors. The thesis has three main contributions. First, we add local market functionality to BooksPicker, making it the first tool that does price comparison between both local and non-local (online) sources. Second, we use an automatic pricing mechanism to price books based on current market prices and the book's condition. Third, we use an offer prioritization mechanism to sort competing offers based on location and social information. Even though our focus was on building the local market, considerable effort was put into improving the BooksPicker service as a whole.

Our system was deployed at MIT during the Fall 2010 semester to an audience of over 4,000 undergraduate students. We measured its performance in terms of traffic, local market effectiveness, and through user feedback in the form of surveys. We saw a sharp increase in BooksPicker usage, with the number of transactions

increasing six-fold compared to each of the two previous semesters. We also saw an increase in sale probability and decrease in market time for offers using automatic pricing. Unfortunately, we did not obtain enough data to properly evaluate the offer prioritization mechanism.

The rest of the thesis is organized as follows. Chapter 2 offers motivation and related work. Chapter 3 describes how the system behaves from a user's perspective, as well as discussing the main design considerations. Chapter 4 explains how the sytem was implemented. Chapter 5 describes how the system was tested and presents the all the results. Finally, Chapter 6 examines the results and makes recommendations on future work.

Chapter 2

Background

This chapter offers motivation and presents related work.

2.1 Motivation

Textbook prices have always been a problem for college students and they have been constantly increasing over the past few years. In the most recent Undergraduate Cost of Living Survey at MIT, administered in November 2009 by the Committee on Undergraduate Admissions and Financial Aid (CUAFA), it was shown that over 25% of students spend over $800 on books per year [4]. In fact, the MIT Financial Aid office budgets $1,050 for books per student per year [11].

High textbook prices have been such a problem that the Higher Education Act (HEA) recently introduced requirements regarding textbook information. The HEA states that effective July 2010, educational institutions must disclose the retail price and ISBN of required and recommended textbooks for the upcoming semester. This should give students ample time to search for their books and get better deals.

We believe more can be done to help students lower their book spending. More savings can be achieved through college-specific solutions that allow students to buy and sell their books while providing them with relevant information, such as regular and buy-back prices at multiple stores, what books are needed for which classes, and feedback on how useful are books for particular classes. The work presented on this

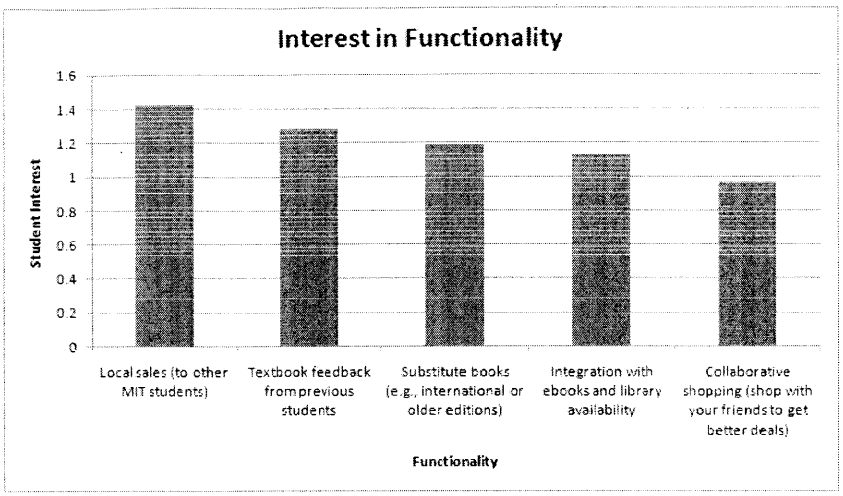

Figure 2-1: Survey responses to 'Which of these ideas would you like to see in an online tool that helps you get your textbooks?' Students were asked to choose 'Not Interested,' 'Sounds OK,' or 'Please do it now!' for each category. Those options were then mapped to scores of 0, 1, and 2, respetively, and averaged to be displayed in the graph.

thesis attempts to be that solution, but focuses on the subproblem of building a local market.

At the beginning of this project, we conducted a survey targeted at the MIT undegrad population to better understand the student need. There were 520 responses, equivalent to about 12% of the undergrad population. Figure 2-1 shows the students' preference towards feature ideas presented in the survey. According to the survey, a tool that allows students to sell their books locally was the most wanted feature.

We also asked students what they usually do with their textbooks after they are done using them for their classes. Figure 2-2 shows their responses. According to the survey, most students at MIT actually keep their textbooks. It is unclear whether this is by choice or due to other reasons, but we still argue that a local sales system fills a big need in the MIT community. Almost 50% of the college textbook market is dominated by on-campus bookstores [8], largely because of their convenience. A local market can be just as convenient, while not having the typically high prices that college bookstores have. However, the volume of books is admittedly much lower, especially among upperclassmen who tend to keep textbooks within their fields of

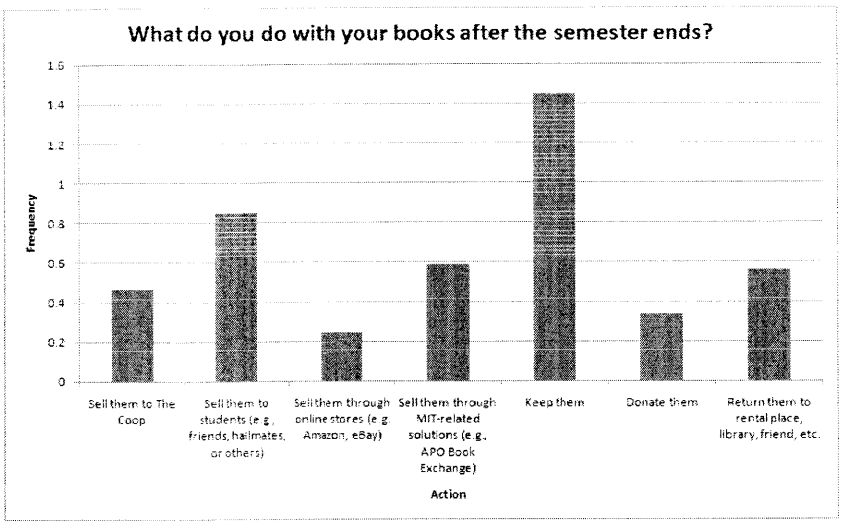

Figure 2-2: Survey responses to 'How do you get rid of books?' Students were asked to rank each category as 'Never,' 'Sometimes,' or 'Often.' Those options were then mapped to scores of 0, 1, and 2, respetively, and averaged to be displayed in the graph.

study. For instance, computer science majors will most likely keep their copy of the CLRS algorithms book, while they are probably not interested in keeping their freshmen chemistry books.

A more interesting question is *why* students do not get rid of their books. Figure 2-3 shows the main pains associated with selling books at MIT. This question was open-ended to avoid any kind of bias set forth by the survey. The 364 responses were then categorized manually.

To build an effective local sales solution, we need to account for these problems. Our solution primarily tackles the problems of pricing and finding a buyer, while secondarily addressing the problems of time, relevance, commitment, transportation, and coordination. Specifically, we address pricing by introducing an automatic pricing option. The pricing problem manifests itself in two ways: choosing a good selling price and being able to sell books at a reasonable (high) price. The automatic pricing method chooses the price by primarily using current market prices as a reference and setting the price to be competitive but not too low. The method also helps books sell fast, addressing the issue of time.

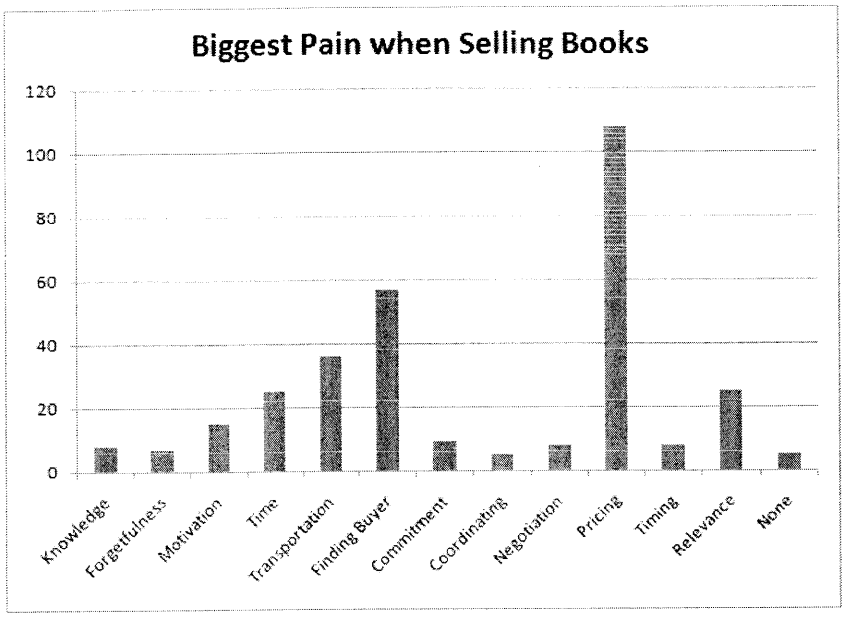

Figure 2-3: Report of students' biggest pain when selling books. To clarify: Knowledge refers to not knowing where or how to sell; Time refers to the process taking long to complete while Timing refers to waiting for the right time to sell books (upcoming semesters); Commitment refers to finding buyers that actually follow through with their promises; Coordinating refers to setting up a meeting time and place with the buyer; Negotiation refers to dealing with people who want to negotiate prices further; Pricing refers to both choosing a sell price and to selling the books too cheap; Relevance refers to the difficulty of selling books that are outdated or no longer needed for a particular class. This was an open-ended question summarized manually. Total number of responses: 316.

To address the issue of finding a buyer, our solution incorporates Facebook and location information to prioritize offers to potential buyers, as well as personalizing them whenever possible. This approach also helps with commitment, since it increases the likelihood of buyers and sellers knowing each other, as well as addressing transportation and coordination due to the usage of location information (e.g., it is easier to carry out a trade if the parties involved live next to each other). Lastly, our system offers book search by class functionality which we harness to help with the problem of relevance. When a student sells a book for a class, the book will appear in the class search results even if it was not originally associated with that class. By being associated to a class, old editions can attain more visibility, therefore increasing their chances of being sold. Chapter 3 contains detailed explanations of how these

features were designed and why.

Finally, we should mention that while this work is performed at MIT, the problem it tries to solve is present in many colleges. MIT serves as a case study and as a model to design the service around, but the entire system was designed with extensibility in mind so that it can be easily implemented in other colleges.

2.2 Related Work

This section presents some of the alternatives students have for selling their books.

2.2.1 BooksPicker

In 2009, three MIT students, including the author, started working on BooksPicker, a tool that helps students buy their books for less. At the start of this thesis, BooksPicker had already been live during the Fall 2009 and Spring 2010 semesters at MIT. It provided three major features: textbook information, price comparison, and bundle optimization. Textbook information allows students to find the needed textbooks by searching for a class number. Price comparison allows users to search for the cheapest prices across multiple stores (at the time only Amazon, eBay, and Alibris). Lastly, bundle optimization finds the cheapest price for a group of books, which may be less than the sum of the cheapest prices for each book individually.

This thesis takes BooksPicker as a starting point and builds the local market functionality on top of it. However, it is worth noting that while the author is a member of the BooksPicker team, the work described on this thesis was performed *separately* and entirely individually, with very few exceptions. That is, the work performed is not BooksPicker's, nor does this thesis own any of the other BooksPicker work. We only used BooksPicker as a vehicle for us to add new functionality, perform tests, and evaluate the system.

Before BooksPicker started, there were already a few solutions out there that offered similar functionality. There is a large number of comparison shopping engines, such as NexTag, as well as several textbook-specific solutions. The most similar so-

lution at the time of writing is BigWords, which in addition to comparison shopping, also includes bundle optimization (which they call "Multi-Item Price Comparison" [12]) and international editions[1]. However, no solution offered the class search functionality that BooksPicker created.

2.2.2 The COOP

The COOP, MIT's local bookstore, will buy back books that have been requested by professors for future semesters. They buy back titles at 50% of the price paid by the student, and they sell used books at 75% of their 'new' price [9]. That is, upon resale, The COOP makes either 25% or 37.5% of the 'new' price of each book, depending on whether the student bought it new or used, respectively. This is a high profit margin for an entity that is not much more than an intermediary. For books sold from students to students, that margin can be distributed across the buyer and seller.

2.2.3 APO Book Exchange

Another common place for students to sell their books is the APO Book Exchange. APO is a co-ed service fraternity that runs multiple service projects such as the book exchange. The program works by allowing students to bring in their books and set a (fixed) regular sale price, as well as a final-day sale price. The books are then sold during the first week of the semester. APO handles all transactions and later pays the sellers for the books they sold, while keeping 5% of every transaction [2]. After the exchange itself, students can pick up their unsold books or choose to leave them, in which case they will be donated to charity. The exchange handles in the order of a thousand books per semester, half of which are usually sold.

This program is convenient for students since they can drop off their books at the end of the semester for them to be sold at the beginning of next semester. Students

[1]International editions are book versions meant to be sold outside the United States. They often contain exactly the same contents as their US counterpart, but are priced at vastly lower prices, sometimes going as low as 5% of the US edition's price. Thus, they are a great cheap alternative for college students.

do not need to store the books in the meantime. Moreover, they do not have to coordinate with buyers individually whenever they sell a book.

The system has its drawbacks too. One of the main problems is that students do not know how to price their books. For instance, this semester (Fall 2010), we saw two copies of the same book in comparable conditions, one priced at $15 and the other one at $100. Incidentally, the demand for that book is very high, so both copies sold. The problem with pricing is not only that some books are priced too low, but some books are priced too high and thus do not sell. Other drawbacks include that students need to carry all their books to the APO office (located in the MIT Student Center) and that money for sold books will not be available to them until a few weeks after the exchange ends.

Despite these problems, the APO Book Exchange remains one of the most popular solutions on campus, and the survey corroborates this fact. This program is probably the only MIT solution that has had a continued presence on campus, since it is maintained by a student organization that continues to live even after students graduate.

2.2.4 Other MIT-specific Solutions

There have been other student-driven attempts to build local textbooks markets at MIT. A few examples are MIT412, CampusBeacon, and BookX. However, all of these have died due to poor maintenance after the developers left MIT.

A noteworthy difference is that all of these solutions have catered exclusively to the local market. In contrast, our work integrates both local and external offers for books. In fact, we put considerable effort into incorporating other local book sources, such as the APO Book Exchange and The COOP itself. We believe there is great value in having a 'one-stop shop' when it comes to buying textbooks, and missing a single important book source would make the solution lose a lot of its added value.

2.2.5 External Solutions

Apart from the MIT-specific solutions, there are a number of external alternatives where students can sell books. However, we argue that these alternatives are inadequate for MIT and other college markets due to audience, convenience, and money matters.

E-commerce sites such as Amazon or eBay are too much of a hassle for most students to use them for selling. First, they often require sellers to register and build a reputation; a seller with no reputation will have a hard time making sales. In contrast, local MIT buyers will almost automatically trust other student sellers, if only by the fact that they are both at MIT. Second, students are responsible for prompt order fulfillment (packaging and shipping). In terms of audience, students have to deal with strong competition in a saturated market. Though the online market is bigger than the local market overall, it is fragmented across many sites and there might be less people interested in these college-specific books, many of which are custom MIT editions. Lastly, these alternatives also have high costs for the sellers. For instance, Amazon charges a 15% referral fee, on top of a "closing fee" of $2.34 for each book sold [3].

Other simpler alternatives like Craigslist and Facebook Marketplace are inadequate mainly due to the audience. Most students do not even consider these alternatives when buying books, since they serve a much broader market. Though people can (and do) sell books through them, they are mostly used as a market for furniture, home rental, and other big objects, such as cars, instruments, or home appliances. Since very few students shop for books through them, they are not a good place to sell books either.

Another alternative that is emerging is Belltower Books. Belltower focuses on convenience. The company hires local school representatives that will buy books from students at set prices. The local rep is supposed to make it as convenient as possible for students, even going to the seller's dorm to pick up their books. In terms of buy-back prices, Belltower sometimes offers more than The COOP, while

sometimes it offers less. Local reps keep 6% of every transaction, and Belltower later resells the books through Amazon, eBay, or other online stores [5]. Unforunately we do not have any data to show how successful or unsuccessful this alternative is.

Chapter 3

Design

This chapter describes how the system behaves from a user's perspective, as well as presenting a discussion of our important design choices.

3.1 System Specification

This section describes how the system behaves from a user's perspective. We walk through the system behavior by going through the two usage scenarios: buying and selling books.

3.1.1 Buying

In order to buy books, the first thing the user does is search for them. A user can perform a *class search* or a direct book search (using ISBN or keyword), as depicted in Figures 3-1 and 3-2, respectively. Note that the class search results are different from book search results. The class search returns the list of books associated to the class, *including* information about the book's necessity for the class and any relevant notes, while the regular book search only includes the matching books' information. Sometimes there is no data available for a class. In those cases, we present a link to the class itself, as shown in Figure 3-3.

The next step is to build a bundle. The user can 'Pick' books to add them to their

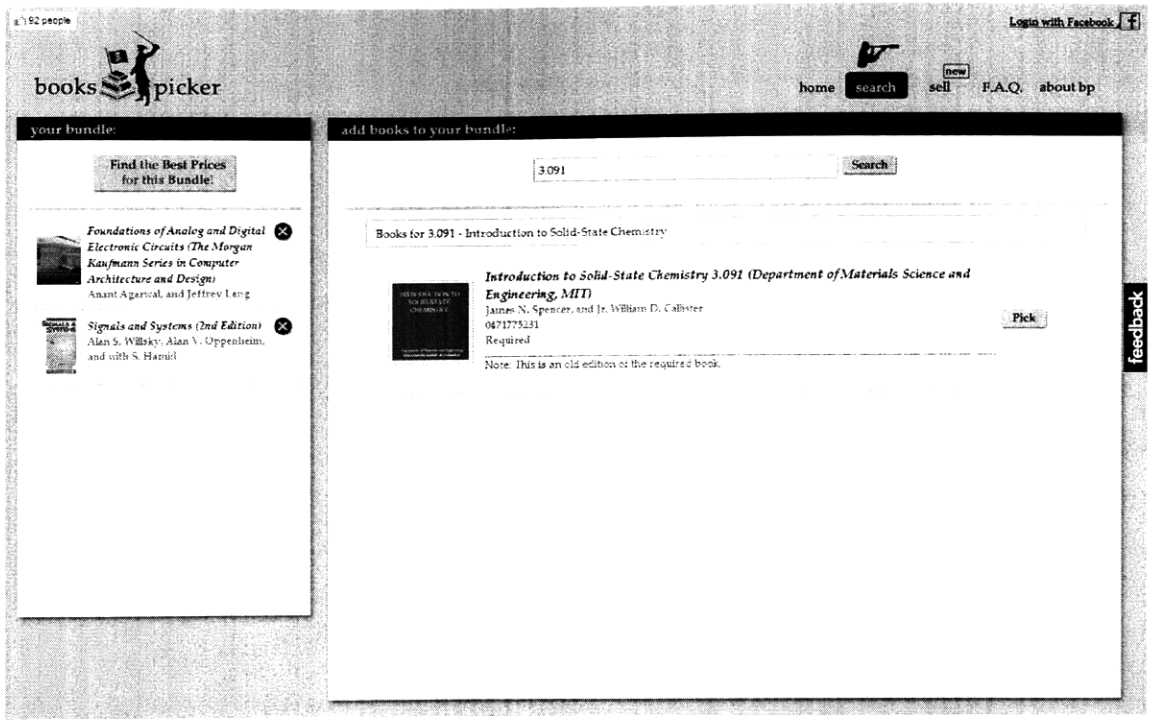

Figure 3-1: Screenshot of a class search.

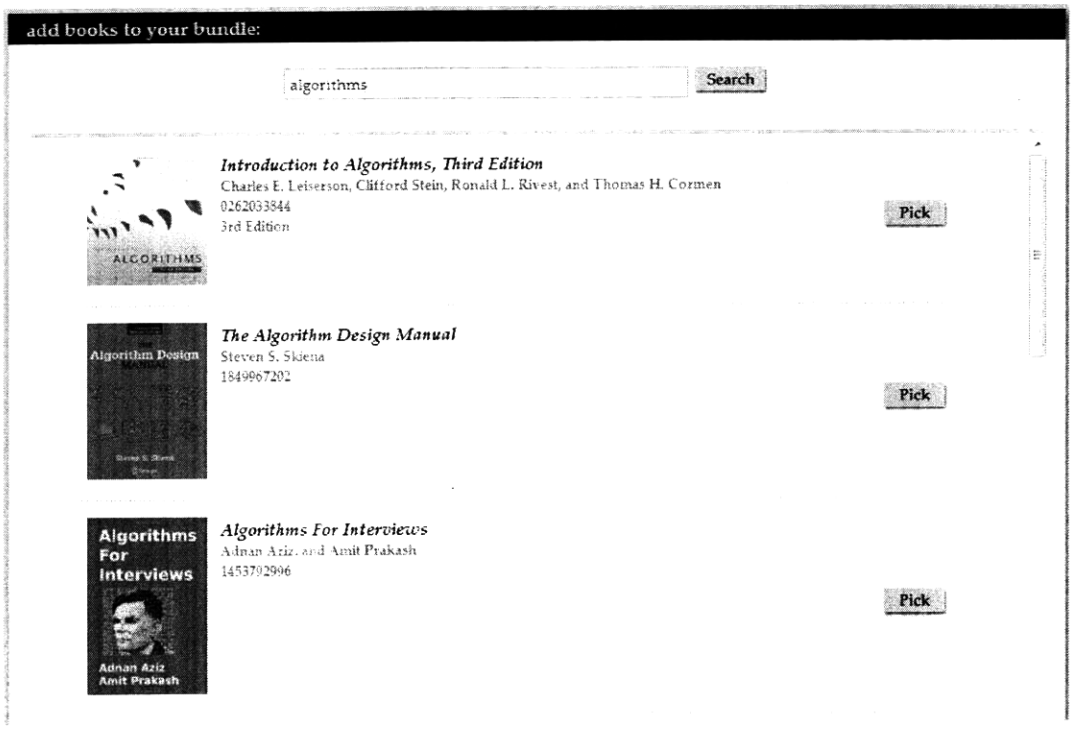

Figure 3-2: Screenshot of a book search.

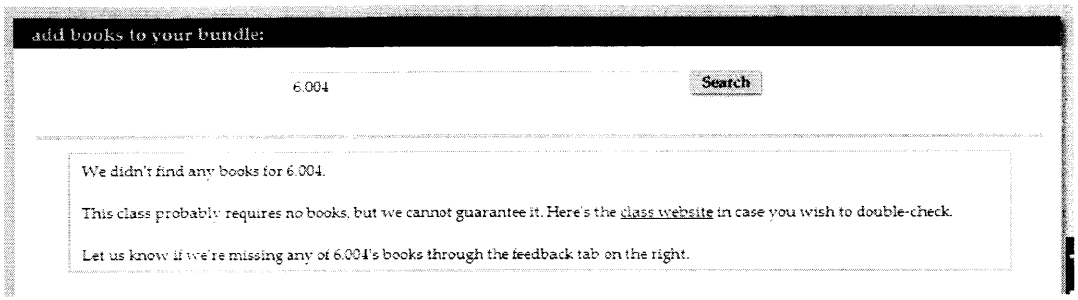

Figure 3-3: Message shown when a class search yields no results.

bundle, as shown in Figure 3-1. Next, when the user clicks on 'Find the Best Prices for this Bundle', we fetch all the offers available for the bundle's books and perform bundle optimization as explained in Section 2.2.1. The current BooksPicker version supports non-local as well as local stores. The supported stores non-local stores are Amazon, AbeBooks, Alibris, and Half (eBay), while the local stores include The COOP, the APO Book Exchange, and BooksPicker itself (local market).

For each book, the best two offers are shown. We show two offers instead of one so it is clear to the user that there is more than one option available. The user can then expand the the table to show all available offers, as depicted in Figure 3-4. Offers are sorted by *total* price using bundle optimization[1]. However, whenever two BooksPicker offers are available and they are the same price, they are sorted using other information. We first show the offer that is closest to the user in terms of location. If the distances are the same or location information is not available, we use the social distance between the buyer and seller; whichever seller is socially 'closest' to the buyer is shown first. In addition, the seller's name will be shown as 'A friend' or 'A friend's friend' whenever applicable. Lastly, if we are unable to differentiate the offers in this way, we show the oldest offer first. Section 3.2.4 further explains why we sort the offers in this way.

Finally, when the user clicks on 'Buy Offer', what happens depends on the whether

[1]Bundle optimization implies that the cheapest offer for a book is not necessarily the best offer to buy, and therefore it may not be shown first. Instead, the guarantee is that when *all* the top offers are taken together, they result in the cheapest cummulative price for all the books (by using discounts and promotions). The cheapest offer is actually the best offer over 80% of the time though.

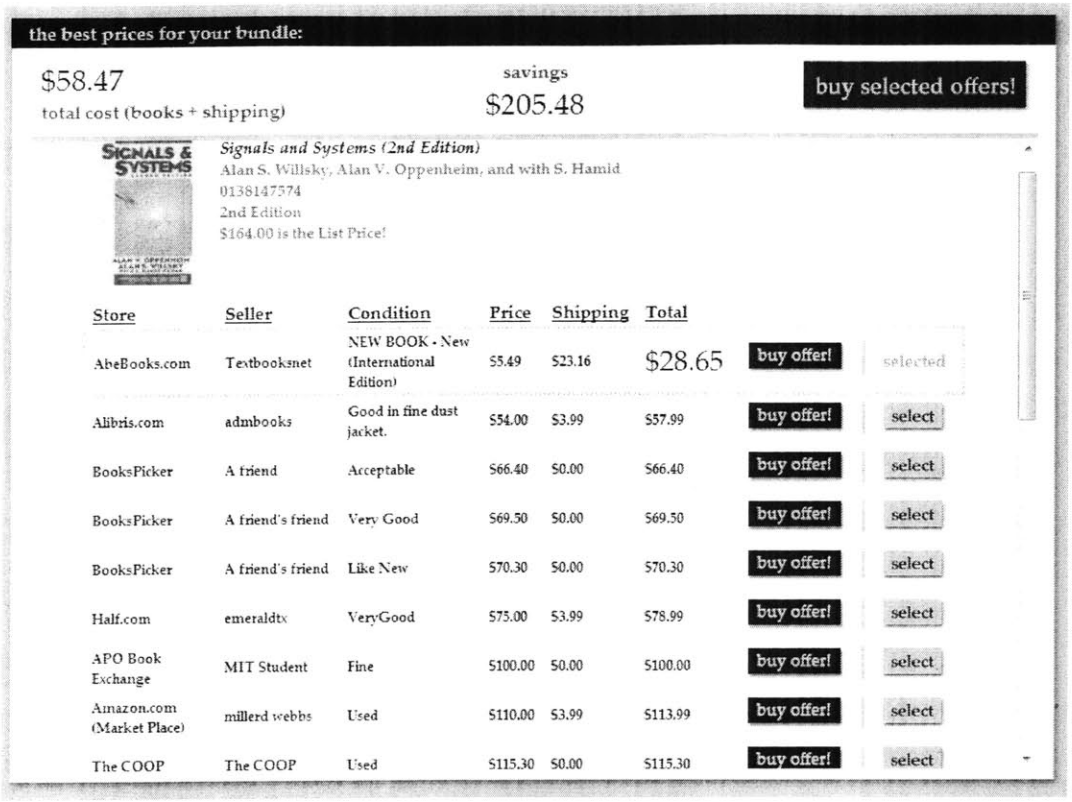

Figure 3-4: Offer table showing current offers for a book.

the offer clicked is a BooksPicker offer or not. When the user clicks on a non-BooksPicker offer, they are taken to the corresponding site where they can purchase the book. Else, they are taken to an offer view within BooksPicker where they can see more details about the offer and buy it, as seen in Figure 3-5. Upon clicking 'Buy Offer' again, we put the buyer and seller in contact with each other by sending them an email. BooksPicker does not handle any money itself and serves only to connect buyers with sellers for free.

When a user "buys" a local BooksPicker offer, we immediately take the offer off the market so nobody else can buy the same offer. This is why we ask users to agree that they actually intend on buying the book before clicking on 'Buy Offer'. We assume that the offers in the market are substantiated, meaning that if an offer exists on our system, then it is indeed available in reality. This is unlike other local markets, such as Craigslist, where the posting remains active for a week, regardless

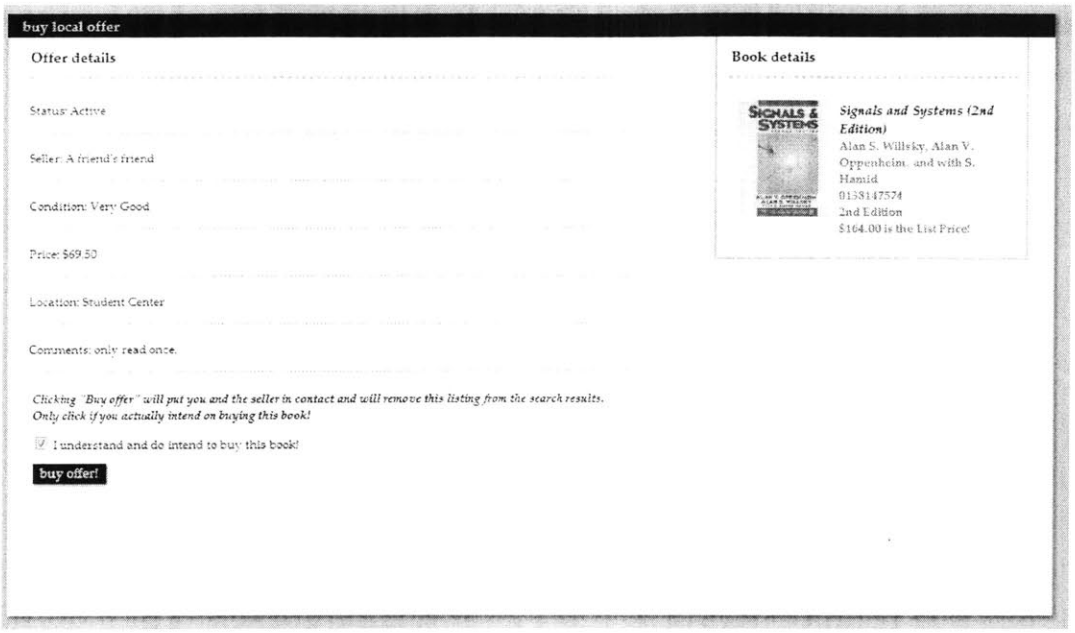

Figure 3-5: Local offer view shown to potential buyers.

of how many buyers contact the seller. In addition, we do not allow the same user to buy the same book (from BooksPicker) within a 6 hour period. We do this to prevent users from contacting multiple sellers with the intention of only buying one book. We assume that users are only interested in a single copy of a given book. This strategy is further discussed in Section 3.2.2.

3.1.2 Selling

In order to sell books through BooksPicker, the seller must first login with Facebook. They can then proceed to post their books through the interface depicted in Figure 3-6. A seller is only required to provide the book's ISBN, its condition, and a price. When the seller provides the book ISBN, we fetch the book's information and the current market prices for that book, as shown on the right-hand side of the figure. The market prices are included to aid the seller in setting a price.

In terms of pricing, the seller has the choice to set their own fixed price or choose our automatic pricing option. When using automatic pricing, we calculate a price for the book every time it is searched for. The price depends on the current market

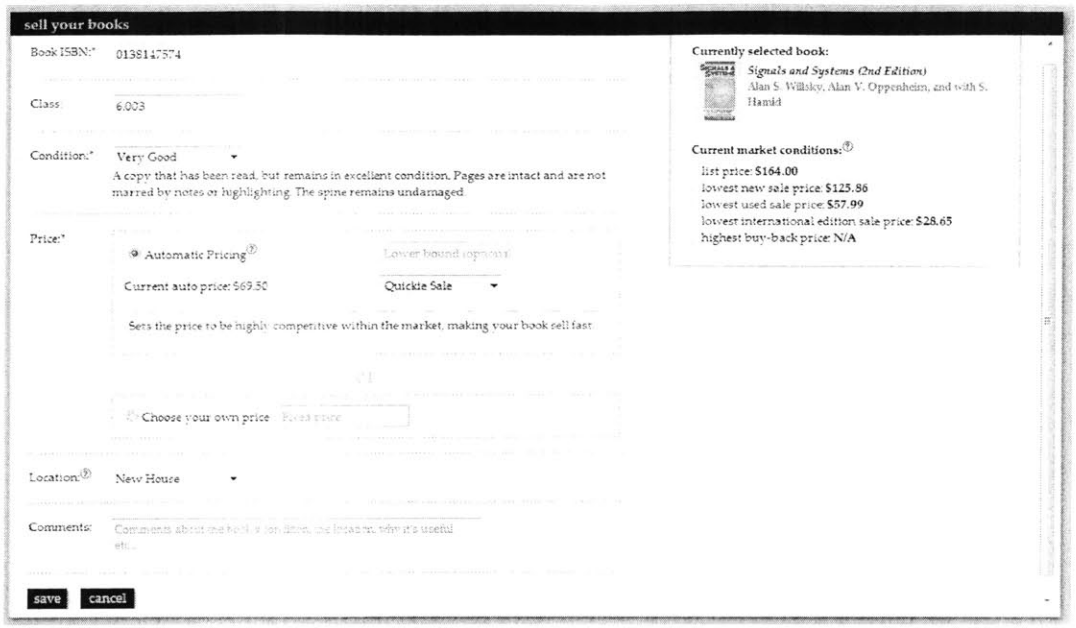

Figure 3-6: Interface used to create and edit local offers.

prices for that book and the book's condition. When using automatic pricing, the seller is required to choose one of the two pricing strategies:

Quickie Sale Sets the price to be highly competitive within the market, making the book sell fast.

Money Lover Sets the price competitively but not as much. The book may take longer to sell but the seller will get more money.

The seller can also see a live preview of what the automatic price would be at the time of posting. And as a safety measure, the seller can set a lower bound on the automatic price.

The method used to calculate the automatic price is extremely simple. A reference price is chosen from the market prices and is applied a multiplier that depends on the strategy and the book's condition. The reference price is set to be the second lowest price found on the market, after excluding student offers and offers for international editions. Here, student offers include those found on BooksPicker and the APO Book Exchange. When there are less than two qualifying offers available, the lowest price

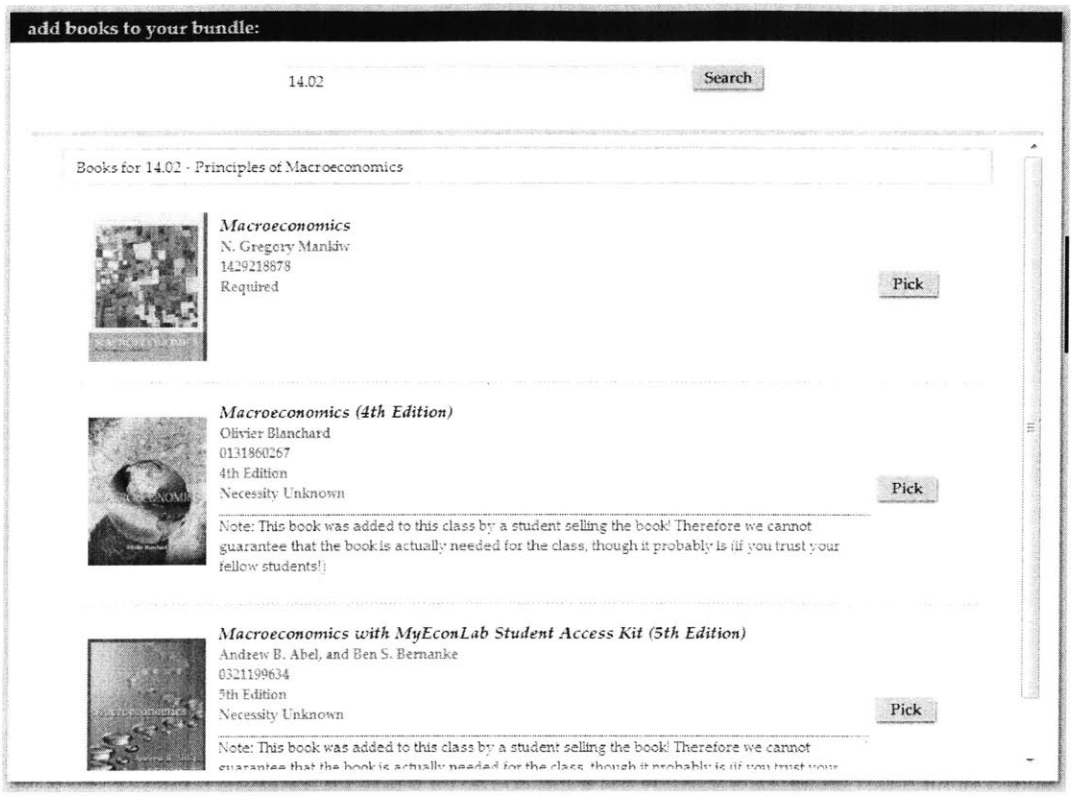

Figure 3-7: View showing books added to a class through local offers.

is used. The reasoning behind the automatic pricing method is further discussed in Section 3.2.3.

Note that in some cases, not enough information might be available to calculate an automatic price. When this occurs, the offer is not shown to the potential buyer. This ocurrence is extremely rare though; it only happened to one offer during our evaluation period.

The seller can optionally include a class number, location, and comments in their listings. When the seller sets a class number whose book list does not include the book being sold, the book will be added to the search results for that class, together with a cautionary note as shown in Figure 3-7. Otherwise the search results remain unchanged.

When the seller includes location information, BooksPicker will use that information to prioritize their offer over others, as discussed in Section 3.1.1.

The seller can view and manage their offers through the interface in Figure 3-8.

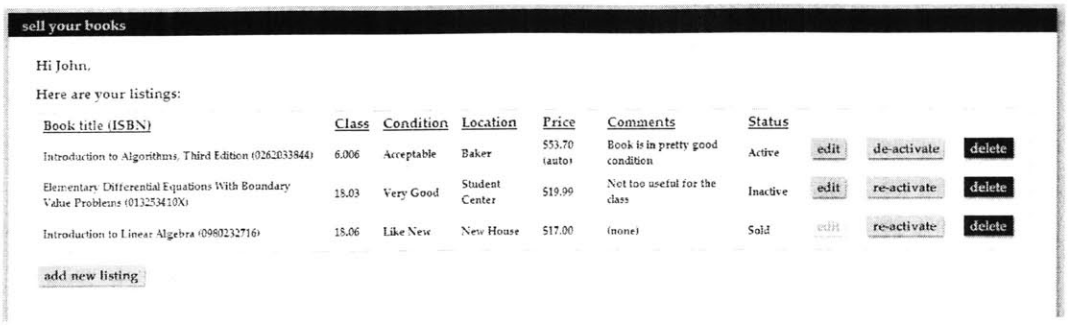

Figure 3-8: Interface used to manage local offers.

Sellers can edit, activate, de-active, or delete their offers. Note that a seller can see and re-activate a sold offer. This is allowed to account for the case in which the offer is de-activated due to someone "buying" it, but the transaction is not completed.

Lastly, the offers are not live indefinitely; they expire after 5 days. Upon expiration, the seller is sent a notification email through which they can re-activate the offer if they wish.

3.2 Design Considerations

This section discusses some alternative ideas to the current implementation. In analyzing these questions, we sometimes consulted a focus group of students. The number of students consulted varied from 5-10 depending on the question being discussed. The students were chosen from our own social network and conversations were held informally.

3.2.1 Book List Data

The initial class book list data used to power the class search in BooksPicker comes from both MIT and The COOP. Additional data is added through crowdsourcing (see Section 3.2.5). We use the MIT Data Warehouse [6] to obtain the data whenever available. However, since MIT's data remains incomplete, we sometimes rely on The COOP. The COOP's data is available at their website, which allows users to select classes and see their book lists. To avoid cluttering the interface, we decided not to

include the information source with each book listing, though we may include it in the future.

An option we considered was to use a fallback mechanism in which data from previous semesters is displayed whenever there is no data available for the current semester. The MIT course catalog currently uses this technique. However, we chose not to do this since we anticipated students to heavily rely on our data to decide which books to buy, especially freshmen. By the time students search on BooksPicker, they are already thinking about buying books. We therefore prefer to show no data at all than to show potentially erroneous data. In the future, might consider showing old data with appropriate markup indicating that it may not be reliable, while allowing users to confirm or deny such listings.

3.2.2 Local Sales System

In our local sales system, when a user is interested in buying a book, he can request to buy it by clicking 'Buy Offer'. From this point forward, we refer to this action as a user "buying" an offer (with quotations). At that point, the listing is taken down and both the seller and buyer are put in touch through email. We assume that most sales go through to completion. In case one does not, the notification email sent to the seller also contains a link they can click on to re-post their book. If the buyer retracts, he must reply directly to the seller so they can re-post the book. Currently there is no way for the buyer to cancel through our system.

We opted for a local market model where every offer is assumed to be substantiated and all communications are one-to-one. In contrast, Craigslist uses a many-to-many model. In such, buyers typically contact multiple sellers in hopes that one of them will respond. Likewise, sellers typically receive multiple inquiries and respond to whichever offer they prefer, usually based on convenience, arrival time, price, and trust.

While this model might work well for Craigslist, it is non-ideal for a local book market. In the book market, once a student finds a book they want at a good price, they just want to buy it. In fact, according to our initial survey, students do not

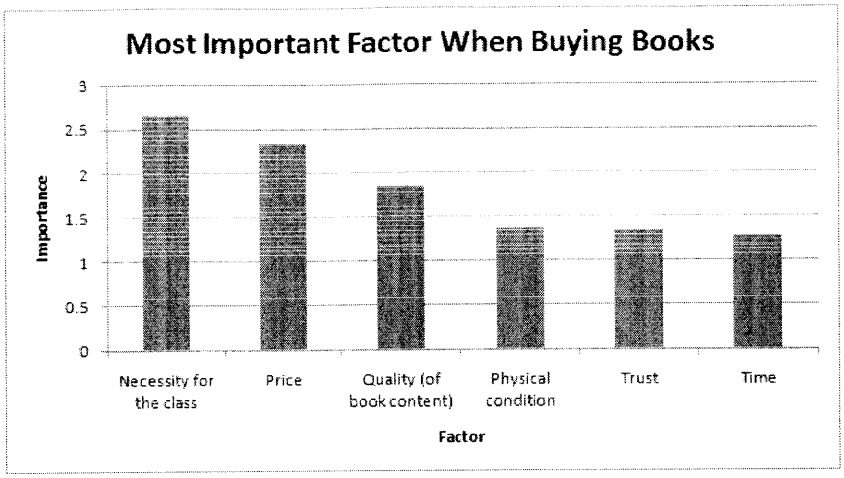

Figure 3-9: Student response on what are the most important factors to consider when buying a book. Students were asked to choose 'Not Important,' 'Somewhat Important,' 'Important,' or 'Very Important' for each category. Those options were then mapped to scores of 0, 1, 2, and 3, respectively, and averaged to be displayed in the graph.

care as much about the book's condition (see Figure 3-9). This situation is different than Craigslist's, where the objects traded are typically furniture or cars and there is a lot more variation. Therefore, we assume that when a user "buys" an offer from another, the buyer actually intends on buying the book as is not "shopping around". It is therefore safe to remove the listing from the market.

In addition, removing listings from the market as the users "buy" them helps us tackle the problem of stale listings: listings for books that are no longer available. Stale listings are a nuisance for buyers and decrease the overall quality of the market. To minimize them, we chose to expire listings after 5 days. We decided to expire listings after 5 days due to the fast-paced nature of the market. The highest book-buying activity happens during the first 2 weeks of classes each semester. During those weeks of extremely high activity, it is likely that books sell in under 5 days. And, since it is likely that students will attempt to sell their books through multiple means in parallel, then the book is likely to be sold within 5 days, giving us reason to believe that any listing older than 5 days is stale. When a listing expires, the seller is notified by email and they can quickly repost their book if they need to.

Stale listings not only emerge from books being sold through other means. In theory, a user could find a local offer for a book in BooksPicker and contact the seller directly for the book, leaving the listing alive. For that reason, we hide the seller's information, forcing the buyers to "buy" the offer in order to proceed. Sellers could (and sometimes do) circumvent this by adding their contact information as a comment in their listings, though this is discouraged.

Notice that we implemented a traditional market where sellers set prices and buyers decide whether to buy or not. In such markets, tedious work (undercutting and bargain hunting) typically results in more money made or saved. We considered other market types where this type of word would not be necessary. For instance, a double-bid system in which sellers set their lowest price, buyers set their maximum price, and a clearing house figures out which transactions to make so that overall utility is maximized. We did not pursue these types of models because, according to the focus group, they seemed overly complex and perhaps even unnecessary. Students, especially sellers, are not interested in an efficient market where every penny is maximized; they just want a simple way to sell their books.

3.2.3 Automatic Pricing

Pricing was the most important problem mentioned in our initial survey (results on Figure 2-3). We introduced automatic pricing to aid the seller in choosing a price that would be low enough for the book to sell, but high enough for the seller to be satisfied with their return.

Even though there are many ways to determine a book's price while considering condition, book quality, and other factors, we primarily focus on other market prices to select a price. Since competing books are essentially the same, price tends to be what makes or breaks an offer. Indeed, as shown in Figure 3-9, price is the second most important factor when buying books, after the book's necessity in the class. However, when comparing two instances of the same book, their necessity for the class is the same, so the price effectively becomes the most important factor.

We divided the automatic price algorithm into two parts: choosing a price (or

Table 3.1: Multiplier offsets based on book's condition.

Condition	Offset
Like New	+1%
Very Good	0%
Good	-2%
Acceptable	-4%

prices) to use as a reference and then setting the automatic price in relation to the reference. The reference price is chosen as the second lowest price found on the market, after excluding student offers and offers for international editions. Here, student offers include those found on BooksPicker and the APO Book Exchange. Next, we apply a multiplier to the reference price that depends on both the user-chosen pricing strategy and the book's condition. More specifically, the automatic price is set to:

$$\text{autoPrice} = \text{referencePrice} * (\text{baseMultiplier} + \text{conditionOffset}).$$

The base multipliers depend solely on the strategy chosen, and are set to 88% and 98% for the "Quickie Sale" and "Money Lover" strategies, respectively. The condition offsets are shown on Table 3.1. Note that the automatic price ranges between 84% and 99%.

The reason why we exclude student offers when choosing the reference price is twofold. First, the local market is completely different from the online market. Some students are willing to price their books extremely low either because of lack of information about pricing, a desperate need to sell, or goodwill. Therefore their prices are sometimes substantially lower than those in the online market and do not reflect the "true" market price. Second, malicious users could manipulate the system by adding cheap offers of their own in order to lower the price of a local offer of their choosing (provided the offer is using automatic pricing of course).

We decided to let the user choose their pricing strategy to give them some control over their pricing technique. We understand that a student's goal is not to maximize

profit on their sold books, but instead to guarantee a sale at a reasonable price. On the other hand, from talking to students, we learned that students have wildly different views of what a "reasonable" price is, so we concluded that a one size fits all solution to pricing would be inadequate. We believe the strategy approach is a good compromise between complexity and control.

There are a couple edge cases to account for. When there is limited market data, we might not have enough information to choose a reference price. If there are less than two qualifying offers, the lowest price is taken as reference (as opposed to the second lowest). If there is no market data at all, we are unable to choose a reference price, so the automatic price is undefined and we do not show the offer in the search results. This almost never happens (it happened once during our trial), but we do keep track of when it does.

In order for sellers to choose the automatic pricing system, it is pivotal that it inspires trust. To do so, we make two simple but important choices. First, we allow the seller to set an optional lower bound price. The automatic price is used unless it falls below this lower bound, in which case the price is set to the lower bound instead. Notice that this method essentially simulates having all the sellers online all the time competitively adjusting their prices whenever the market changes (as the automatic price changes), without them having to be actually involved and while keeping them safe from selling for too little. Second, a live preview of what the automatic price would be is shown to the seller as they select a price. The preview refreshes as the seller changes the pricing strategy, the book's condition, or the lower bound. In future semesters, once we have the data, we should also include how much average gain (if any) a seller using automatic pricing gets over one that does not, both in terms of time and money.

The sections below present and discuss alternative approaches that were considered.

Alternative Reference Prices

We considered several techniques to choose the reference price. The simplest and most intuitive reference is the cheapest price overall. This approach is problematic though, because the cheapest price on the market might be unusually low. Since outliers are relatively uncommon, we opted for simplicity and chose the second lowest price. Having more than one outlier is extremely uncommon, and if there are no outliers, then the second lowest price is typically very close to the lowest, which serves our purpose.

An alternative to the simplistic solution of choosing the second lowest price is to take the cheapest price that fulfills certain conditions (e.g., the cheapest price that is within one standard deviation from the set's mean). Alternatively, we could choose the cheapest price as long as the difference between that price and the next one is less than $x\%$ of the mean price (thus abnormally low prices are ignored). These approaches become problematic when the price range in the market is large; more specifically, when there are "price clusters", groups of low prices and groups of higher prices. Though not common in general, price clusters tend to appear with older books, because normal retailers keep selling them at relatively high prices, while resellers sell them for extremely low prices.

Instead of looking at cheapest prices, another alternative is to choose the median price. Using the median price clearly solves the outliers problem. However, choosing the right strategy to select the automatic price then becomes problematic since it heavily depends on the overall price range. The cheapest price might be very close or very far from the median. An easy fix that uses the same principle is to choose the second or third cheapest price instead of the median, which is exactly what we do.

A more interesting reference price considered is one that is an auto-tuned weighted average. Here, we use a weighted average of the prices, where each vendor has a weight. In turn, these weights change depending on the vendor's demand. For example, if buyers keep buying from Amazon even though it is not the cheapest option, then a seller's competitor really is Amazon. A seller is then interested in beating

Amazon, not necessarily beating AbeBooks or whichever other vendor actually has the cheapest price.

To assign weights, we could use a feedback mechanism such that whenever a user clicks on an offer for vendor X they are casting a vote for it. The weights are then calculated based on the relative votes. But note that Amazon will probably have many more offers than, say, the APO Book Exchange, so its probability of receiving a vote is higher. Thus, to be fair, we count the number of offers each vendor shows and normalize the voting using that information.

This approach is desirable because the reference price tunes itself to the current buyer preference. We did not implement it because prices might be too volatile initially; the reference price might take a while to converge to a "good" reference price. However, we believe this could be a great way to choose a reference price and we will consider using it the future, while seeding the algorithm with the data gathered this semester.

Lastly, another consideration in choosing the reference price is the book's condition. Though not the most important factor in determining the price, the book's condition should have some bearing. A book that is "Like New" should sell for more than an "Acceptable" one. To account for this, we considered having the reference price be condition-specific. However, this is not easy due to the unstructured nature of the data. The condition description that we can gather from the vendors is an arbitrary string. We could classify such strings into discrete conditions by applying a Bayesian classifier or by looking for certain keywords. For simplicity, we did not perform this classification and deferred the condition aspect of our automatic pricing method to the strategy phase.

Alternative Strategies

As with the reference price, we considered several pricing strategies to apply once a reference price has been chosen. After speaking to a few students, it was clear that most students took an "undercutter" approach to selling their books. We mirror this approach, but instead of undercutting by fixed amounts, we use a multiplier-based

approach. As discussed earlier, the automatic price ends up ranging from 84% to 99% of the reference price. We considered having this range include or even go past 100%. It seems students are willing to pay a premium to avoid shipping delays, so this might be reasonable in some cases. Also, pricing at 100% of the reference price is still desirable because even if two offers are the same price, the local offer is shown first due to the fact that it is local. And if they are both local, we fall back to our prioritization mechanism, discussed in Section 3.2.4, to determine which is shown first.

Note that undercutting may be upsetting to other student sellers, who we try to protect. Our chosen strategy avoids undercutting other students because it ignores student offers when choosing a reference price. Therefore the only prices that are undercut are those from online sellers. Also, note that if two students use the same strategy, they will not undercut each other; their prices will be set to be equal and offers will be prioritized based on other parameters. These measures do not prevent a user from setting a fixed price that undercuts everyone else. We do not attempt to solve this problem because its possible solutions (e.g., making it hard to change prices) seem to be more trouble than the problem they solve and because we expect manual undercutting to be very rare.

Another approach to the pricing strategy is to do something that uses more than one reference price. A possibility we explored was to use an upper bound reference price and a lower bound one, and set the price to somewhere in between these bounds (say, 70% in between them). In practice, we would use the cheapest market price (or a price close to it) as the upper bound, while using the highest buy-back price as the lower bound.

We should note that the reason students undercut other sellers is not merely to have the cheapest price, but also to be the first offer shown to the potential buyers. They are willing to lose some profit to increase the likelihood of making a sale. In order to account for this, we considered having the strategy take placement into account. That is, the strategy maximizes a function that is a combination of profit and placement. In order to implement this approach successfully, we need to know

how much is the seller willing to give up in profit for a gain in placement.

Given that there is little to no data to support any of these strategies over the others, we opted for simplicity and implemented the multiplier-based strategy described earlier. Due to lack of time, we were unable to implement and test any others.

3.2.4 Offer Prioritization

In order to prioritize offers for the user when prices are the same, we gathered location and social information. After consulting with students, we concluded that the most important factor to them is location. Since the local market is constrained to MIT, students already trust the sellers, even if they are strangers to them. This would not be the same if the local market extended beyond MIT. Instead, students are more concerned about coordinating a meeting time and place. For simplicity, instead of prioritizing offers based on a combination of geographical and social distance, we decided to prioritize using each feature sequentially. That is, two competing offers are first compared with respect to their location, and only if this comparison yields no winner are they compared based on the social distance between the buyer and the seller. If the offers are match in these two aspects, then they are sorted based on the listings' creation date (older listings first). Note that offer prioritization is likely only useful when the competing offers are both using automatic pricing. Otherwise, it is unlikely that two offers will have the exact same price, not to mention sellers that use fixed prices and manually undercut the other offers, rendering any prioritization (arguably) pointless.

In order to retrieve location information, we allow users to enter a book's location when they sell books through BooksPicker. We assume the book's location is the same as theirs, so we update the user's address to the location of their last submitted listing. This is clearly not foolproof, but it works well in most cases. We considered using the MIT directory to retrieve a user's location automatically. We chose not to do this due to the added complexity and because it is not extensible to other schools. While our work was limited to MIT, we focused on building a solution that is easily extensible to other schools that face similar problems.

We considered using Google Maps to facilitate location input by allowing users to search for an address or simply by click on their location on a map. However, for simplicity, we restricted location information to a few predetermined locations, including all undergraduate dorms and common MIT buildings such as the Student Center.

The metric used to sort offers based on location is the distance between the locations. While this is the intuitive metric, we believe it might be too specific. For example, according to students, all locations within a 5 minute walk from the user's dorm should probably we weighed the same; a minute more or less does not really make a difference in their decisions. In contrast, the difference between a 2-minute walk to a neighboring dorm and buying the book from the same dorm is actually *more* than the 2-minute walk by itself, since it requires the overhead of getting dressed and leaving the dorm (and the overhead is particularly large during the winter). However, these difference do not matter much because of the fact that these numbers are used in an ordinal fashion, as opposed to a cardinal one.

To gather social information about the users, we rely on Facebook. In fact, this is one of the main reasons why we decided to require users to log in with Facebook in the first place. When a user logs in with Facebook, we can obtain a list of the user's friends, allowing us to build a subgraph of the entire social graph. To determine the social distance between two users, we consider both the shortest path between them and the number of mutual friends they have. More specifically, the social distance metric is given by:

$$\text{socialDistance} = \min(\text{shortestPathLength}, 6) - \min(\text{mutualFriends}, 100)/100,$$

with the distance set to 6 if there is no path connecting the two users. Thus, the social distance can vary from 0 to 6. Thus, the possible values can be in the $[0, 1.99]$ interval, starting from 0 and increasing in steps of 0.01, or in the $\{3, 4, 5, 6\}$ set. This approach was chosen because it is simple and follows intuition: the fewer degrees of separation between two people and the more mutual friends, the shorter the social

distance.

Notice that for these methods to work properly, the potential buyer needs to be logged in so we can access their location and social information. This is a problem since buyers have no incentive to log in when searching. They only have an incentive to log in when buying a local offer or when selling books, since the system requires it. To aleviate this problem, we keep users logged in for a week after their last login.

3.2.5 Crowd-sourcing Textbook Information

Since the book list information we get from MIT and The COOP might be incorrect or incomplete, we attempt to crowdsource it to some extent. When a user posts a book for sale and adds a class number to the listing, the book appears in every subsequent search result for that class. It is quite likely that the crowdsourced book information relates to books that were used in past offerings of the class (e.g., old editions). Putting them in the search results is both useful for the buyer, who can choose an old edition as an alternative, as well as the seller, who is more likely to sell their old edition book.

However, in order to avoid misleading the buyer into buying a useless book, the search result must be accompanied with a cautionary note, as shown in Figure 3-7.

3.2.6 Integration with Other Services

In order to improve our reach and inspire trust in students, we attempted to integrate with multiple other MIT online services. In the past, BooksPicker had been integrated with CoursePicker, a tool that helps students choose their classes for the semester. When a student chooses their classes on CoursePicker, they can click on a link that brings them to BooksPicker. The link contains their class information, so we directly display all the books they need for their classes.

In addition to CoursePicker, we collaborated with APO to bring their book listings to BooksPicker. We worked closely with them on defining an HTTP-based API that allowed BooksPicker to access their market data on the fly. When a user wants to

buy an offer available at the APO Book Exchange, we redirect them to their website, where they can complete the purchase. Again, BooksPicker never handles any money and serves only to connect buyers and sellers for free.

We also attempted to integrate with the course catalog, Stellar (a course management system used by professors and students throughout the semester), and Classmates (a student-run service for students to meet other students in their classes). Despite having started conversations about collaboration with these services months in advance, we were unable to successfully integrate due to policy and time constraints.

Chapter 4

Implementation

This chapter describes how BooksPicker was implemented, making emphasis on the important design decisions made. Throughout the implementation of the system, our main focus was on these three areas: usability, performance, and extensibility.

It is worth noting that even though not all descriptions in this chapter pertain to the local market implementation, which is the focus of this thesis, all the work described here was performed within the project timeframe and contributed to improving the service as a whole.

4.1 Client-Side

The BooksPicker client-side is implemented using Google Web Toolkit (GWT). GWT is an open-source platform that enables developers to create browser-based applications by writing code entirely in Java. The GWT compiler takes the Java source and generates optimized browser-specific Javascript. GWT supports a subset of the Java API, as well as providing its own client API, including UI widgets, an event handling mechanism, and wrappers around the HTML document object model (DOM) and other common resources available in Javascript.

4.1.1 Model-View-Presenter Pattern

Our client-side implementation makes heavy use of the Model-View-Presenter (MVP) pattern. This approach enables us to highly decouple our codebase, making it easily extensible and testable.

In the MVP pattern, the presenter contains most of the logic in the application. Contrary to the controller in the Model-View-Controller (MVC) pattern, the presenter is responsible for both reacting to user actions performed through the view *and* driving the view itself as the model changes. In our particular setting, this means the presenter handles view transitions, remote-procedure-calls (RPCs) to the server to fetch and update the model, and handles events sourced from the UI widgets within its view.

In our particular implementation, each presenter defines a specficic `Display` interface, which is able to present whatever model the presenter is presenting. A `View` then implements the `Display` interface in whichever way it wants. The actual `Views` are kept as 'dumb' as possible, knowing nothing about the model they are displaying. Likewise, the presenter does not have any knowledge about the `View` itself. For instance, the `Presenter` does not care if the view shows text in a `TextBox` or a `TextArea`, as long as it contains a text container, or in our case, an object that implements the `HasValue<String>` interface.

As an example, a `SellerPagePresenter` is responsible for driving the main seller page (shown in Figure 3-8). While this presenter handles the overall logic of that page, some tasks are delegated to child presenters, such as a `OfferManagementPresenter` or `EditOfferPresenter`, thus creating a hierarchy of presenters. To go down the hierarchy, presenters create sub-presenters to handle smaller tasks. Conversely, to go up the hierarchy, we use an event-based mechanism. In our example, a custom `EditOfferEvent` is fired by the `OfferManagementPresenter` when the user clicks on 'Edit'. The event is then handled by its parent presenter, `SellerPagePresenter`, which in turn gives the `EditOfferPresenter` the `Offer` to edit and hands over control to it. Note that not all user-generated events must map to a custom event. For

example, when a user clicks on 'Activate', the `OfferManagementPresenter` handles the user event directly by using an RPC to update the offer's status and refreshing the view upon success.

In order to communicate custom events across presenters, we use a shared event bus. The event bus is basically a registry of handlers that are subscribed to specific types of events. When an event is fired, each handler is dispatched in succession. The event bus is initially set up at the top level (in the application's `EntryPoint`) and then passed down the hierarchy as needed. Using the event bus allows us to easily add new functionality into the client, since all that needs to be done is subscribe new listeners to the event bus so they respond to the intended event.

One of the most important reasons behind our decision to follow this implementation strategy was testing. Through the MVP pattern, we completely decouple the presenters from UI elements. Thus, we can test the application logic using mock objects that implement the `Display` interface, all without having to initialize any UI libraries. This permits the tests to run in the order of milliseconds as opposed to seconds.

4.1.2 URL and History support

To make our service more usable, we attempt to make it feel like an application instead of a regular website. Thus, we perform AJAX requests instead of regular HTTP requests, thereby minimizing network traffic and allowing us to provide a faster and seamless interface to the user, in much the same way as Gmail does.

However, even though everything happens in Javascript, we do want to maintain the URL history in the browser. This is necessary for two main reasons: usability and integration. By maintaining the history, we can allow users to go back and forward just like they would in a regular website. For instance, if they search multiple times, they can easily go back to their previous results instead of re-typing the search query.

In terms of integration, it was important to completely represent the state of the application within the URL, so that referrers could send traffic to a page containing relevant information, as well as to allow users to easily share a link with others. For

example, CoursePicker may refer a user to

`http://bookspicker.com/#search?bundle=6.002,6.003&q=6.002,6.003,`

in which case the user's bundle would be pre-populated with the books for 6.002 and 6.003, and the search results view would be showing the book's details. This URL-to-state mapping not only proved itself useful for integration, but also when promoting the service through MIT mailing lists. That is, we could respond to students asking for books with a URL that directly showed them the best offers for the book they were looking for.

In order to implement history support, we introduce a hidden iframe in the HTML and use GWT's history stack to push and pop strings as events occur. However, in order to prevent the browser from sending out an HTTP request when the URL changes, the history can only contain URL *tokens*, which means all strings are preceded by a hash (#). This is exactly how Gmail works, where history is supported, and for example, labels are mapped in the following way:

`mail.google.com/mail/#label/<label-name>.`

Again with performance in mind, we do keep some hidden state in the client so that we can save requests to the server whenever possible. For example, after a user performs a search or builds a bundle, his search state is kept intact even if he browses away to another part of the application. Thus, when he returns to 'Search' after browsing the 'FAQ', all their work will still be there.

Implementing history support in this way has a few drawbacks though. For instance, the history model becomes counter-intuitive when it comes to the user's bundle. After adding a book, the book is removed when the user clicks 'Back' on the browser, which is unlike a regular shopping cart. There are also performance issues. When the server receives a request to serve a page, it ignores any tokens attached to it (every character appearing after a hash). Thus, we need to send multiple requests to the server to serve pages whose URL contain application state information: one

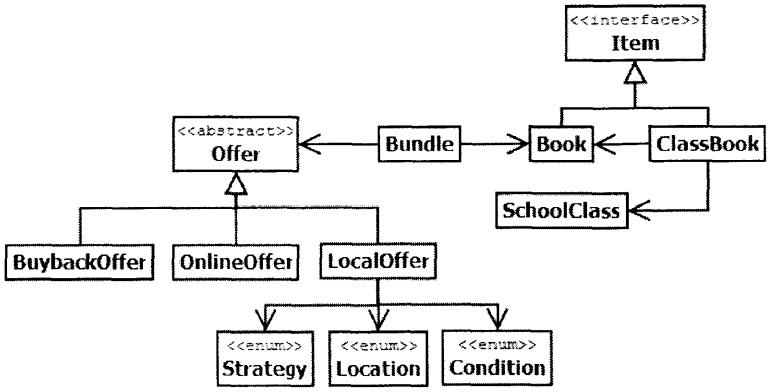

Figure 4-1: Data structures module dependency diagram.

request to load the client and one (or more) requests to load any extra information needed by it. This situation only occurs when loading a URL with state information. In the future, we can address this issue by using a servlet `Filter` to rewrite URLs.

4.1.3 Code Splitting

With performance in mind, we make use of code splitting. Instead of creating a single Javascript file, we separate the code into smaller chunks contained in separate Javascript files. These smaller chunks of code are then fetched dynamically in an as-needed basis and injected directly into the DOM.

4.2 Server-Side

On the server-side, our service lives in a dedicated Ubuntu Lucid server running Apache Tomcat version 6.0. We use a MySQL database as our persistence layer.

4.2.1 Data Structures

The data structures used in our application are straightforward. Figure 4-1 shows a module dependency of the core data structures we use. A short description of the important data structures is included below:

Book A `Book` contains all the necessary information about a book, including its list price and edition, whenever available. The information used to populate a `Book` is taken directly from Amazon.

SchoolClass A `SchoolClass` contains information about a class at MIT, including the class code and its title, and it is specific to a semester or `Term`. More importantly, a `SchoolClass` includes a list of joint classes, which we can use to retrieve book lists whenever there is none available for the current class. Information about classes at MIT is taken from MIT's course catalog, made available through the Data Warehouse.

ClassBook A `ClassBook` serves as a mapping from a class to a book, but contains extra information pertaining to the mapping itself, such as necessity, the source of the mapping, and any notes. We currently have book list information from both the Data Warehouse and The COOP. In the future, more information will be included in a `ClassBook`, such as the students' perceived usefulness of the book with relation to the class (see Section 6.1.1).

OnlineOffer An `OnlineOffer` represents an offer from an external website, whether it is Amazon or the APO BookExchange. It contains information about the book's condition, price, and buy URL.

LocalOffer A `LocalOffer` represents an offer created by a student selling a book through BooksPicker. Apart from all the information about the offer itself, including the book, the class it is used in, pricing method, location, and comments, the structure also contains a fair amount of metadata regarding the offer's performance. For instance, we track the amount of time the offer has been on the market in each of the three pricing possibilities: fixed price and auto-pricing using each of the two strategies. We also track the number of times the offer is shown to users, and the number of times we fail to show it due to the automatic price being unavailable (due to lack of information to calculate it). Lastly, we keep information about whether the offer is currently sold or

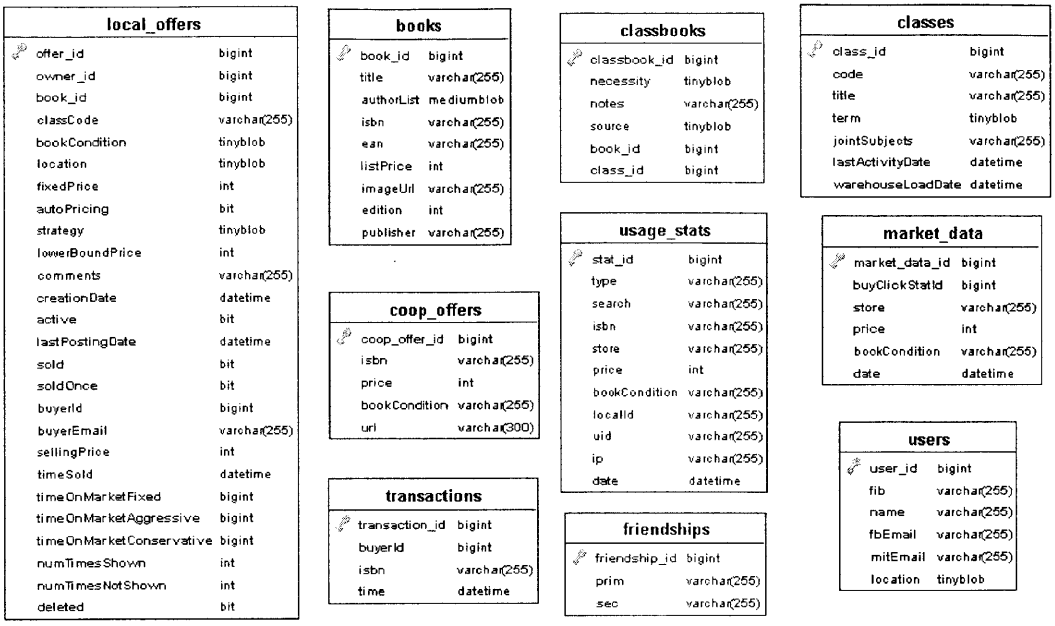

Figure 4-2: Database schema.

not, and whether is has ever sold (since a user can re-activate a sold offer if the transaction does not complete successfully).

Bundle A `Bundle` represents a bundle of books, but it also contains information about the offers available for the books in the bundle.

4.2.2 Database Layer

To make database persistence simple, we use Hibernate as our object-relational mapping (ORM) agent. The database connections are managed by the open-source C3P0 connection pool. Depending on the server load, we maintain anywhere from 5 to 30 connections with the database. In addition, we use a prepared statement cache for all database queries, so that the database does not need to re-generate the query plan every time. The SQL queries we execute are usually all of the same form, so the cache size is kept small (10 statements), yet it is still effective.

Our database schema, showed in Figure 4-2, is largely similar to the data structures described in the previous section. There a few extra tables in the schema though,

some of which are described below:

friendships

This table contains friendship relationships obtained from Facebook.

coop_offers

This table contains The COOP's prices for books needed by classes at MIT. Since The COOP provides no programmatic access to their prices like Amazon or the other stores do, we cannot fetch the prices on the fly and therefore need to manually copy them and store them in this table.

transactions

Every time a user buys a BooksPicker offer, a `transaction` record is inserted in this table. We can then use this information to implement the anti-abuse mechanism mentioned in Section 3.1.1. Note that this table might seem redundant, since the information we need to enforce the anti-abuse mechanism can be inferred by looking at the local offers table itself (`buyerId`). However, in the case in which the local offer sells, gets re-activated, and then sells again to a different user, all within 6 hours, the local offers table would not have the necessary information. This situation is unlikely to happen, but since it is a possibility, we prefer to opt for correctness and keep track of transactions on a separate table.

usage_stats

This table is used to track how the application is being used. We track 8 different types of statistics: class search, book search, buy link click, anti-abuse, offer personalization, and prioritization based on location, social distance, and creation date. The first three stats let us measure how much user activity is happening on the site. We log an anti-abuse stat every time the anti-abuse mechanism is used to prevent a user from buying an offer. The offer personalization stat is logged whenever a BooksPicker offer's seller

name is changed from the generic "MIT student" to something more personal. And finally, every time two BooksPicker offers are prioritized by something other than price, we log the corresponding stat to denote so.

market_data

We use this table to save what the competing offers were at the time when a user clicks on a buy link. This data is later used to analyze what users prefer to buy. For instance, we can estimate how much premium a student is willing to pay in order to get a non-international edition or to avoid shipping delays.

Chapter 5

Evaluation

This chapter describes the evaluations that were conducted. The first two subsections present data obtained from usage tracking. Unless otherwise noted, the data presented in this Chapter corresponds to the the 4-week period centered at registration day of each semester. For the current semester, Fall 2010, the data period starts on August 24 and ends on September 21. The last section presents user feedback requested from our users in the form of a survey.

5.1 Overall Usage and Popularity

This section evaluates BooksPicker's performance in terms of usage.

5.1.1 Traffic

We obtained traffic information from Google Analytics regarding three measures: pageviews, visits, and absolute unique visitors. Figure 5-1 shows graphs for the three metrics, including this semester's data as well as the past two semesters'.

Notice that every semester there is a big spike in traffic. In the previous two semesters the spike has ocurred on registration day, while for this semester, Fall 2010, it occurred 3 days prior. The spikes in traffic are due to email advertising. Sending emails to the dorm mailing lists is by far the most effective way to advertise to

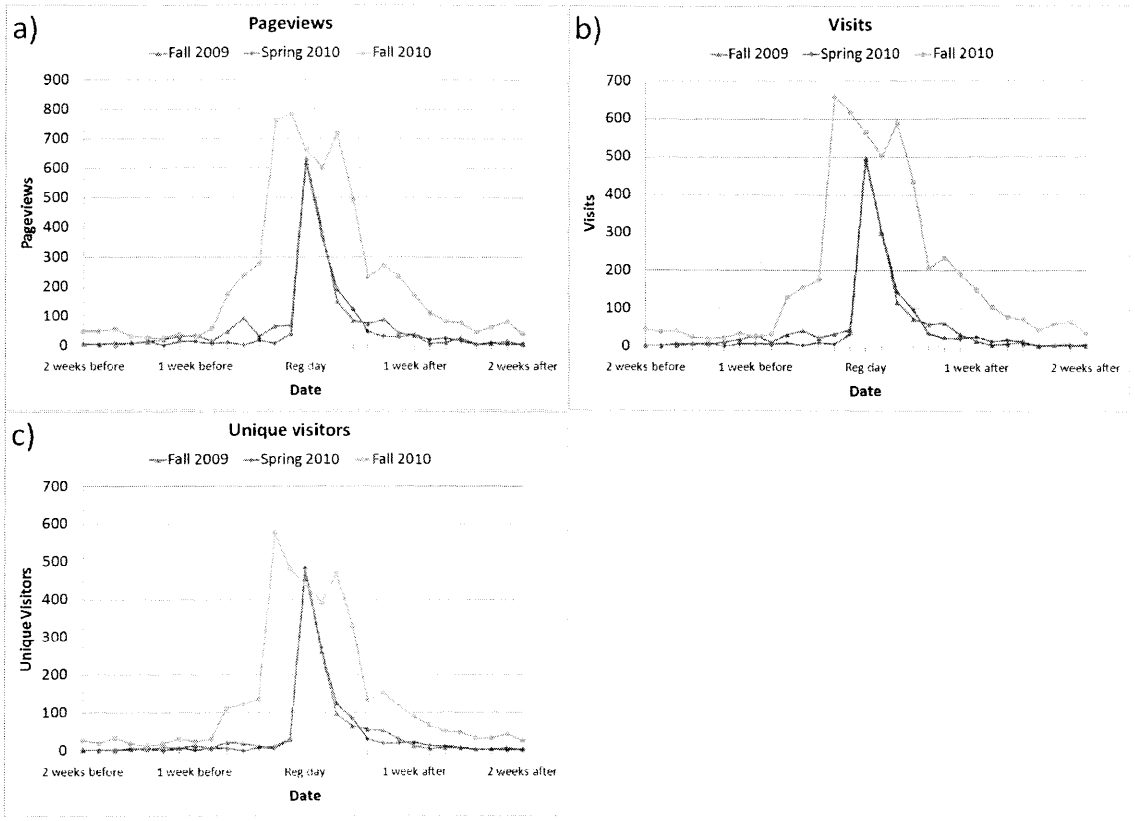

Figure 5-1: Number of (a) pageviews, (b) visits, and (c) unique visitors on Books-Picker.

undergrads at MIT. In the past, we have sent our advertising emails on registration day, but this semester we started advertising earlier. Instead of a single mass email, we sent two: one three days before registration day and one two days after. In addition, we handed flyers during the freshmen registration day, which takes place five days before regular registration day. These three events have very clear corresponding surges in traffic.

Another point worth mentioning is the difference in retention across the three semesters. Note that in the previous two semesters, traffic has died down rather quickly after the main traffic spike. On the day after the spike, pageviews decrease by almost 50%. However, pageviews did not decrease by nearly as much this semester. This could be due to a combination of timing and usability.

Although the graphs clearly show this semester did way better in terms of traffic,

Table 5.1: Total pageviews, visits, and unique visitors.

Semester	Pageviews	Visits	Unique Visitors
Fall 2009	1975	1468	977
Spring 2010	1673	1325	955
Fall 2010	6483	5358	2714

the totals, shown in Table 5.1, are even more telling. Altogether, BooksPicker gathered about three times as much traffic during the Fall 2010 than it did in each of the previous semesters. Considering MIT's undergraduate population is of roughly 4,000 students, BooksPicker attracted about 70% of the population.

The traffic source breakdown is shown in Figure 5-2. As expected, most of the traffic is direct, which corresponds to users typing the address directly on the browser or clicking on an email link or bookmark. Just under 20% of the traffic comes from referring sites. The top referring sites are Facebook and CoursePicker, which account for 14% and 3% of the overall traffic, respectively. The Facebook traffic is mostly due to advertising. Surprisingly, about 9% of the overall traffic comes from Google. However, all the keywords driving the Google traffic are very targeted (e.g., "book picker mit" or "mit bookspicker"). That is, users were specifically looking for BooksPicker as opposed to stumbling across it while generically searching for books. This suggests that students were trying to access the service without knowing the URL, possibly after seeing a poster or having heard about it through a friend.

5.1.2 Transactions

The number of book purchases that were facilitated by BooksPicker over the last three semesters is shown in Figure 5-3, broken down by vendor. Looking at the totals, we can clearly see that this semester was a great success for BooksPicker, since its sales were over six times those of the previous semesters'. However, the local sales platform accounted for just over 11% of the total sales, which is considerably below expectations.

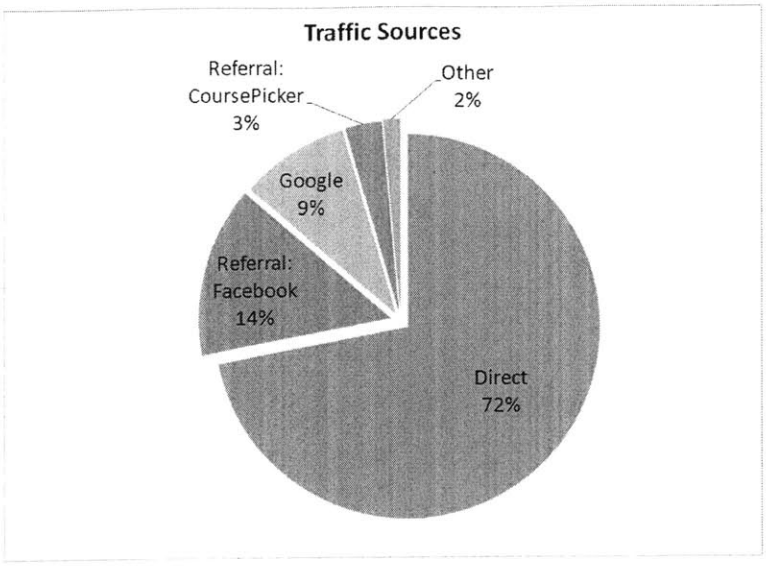

Figure 5-2: Sources of traffic to BooksPicker.

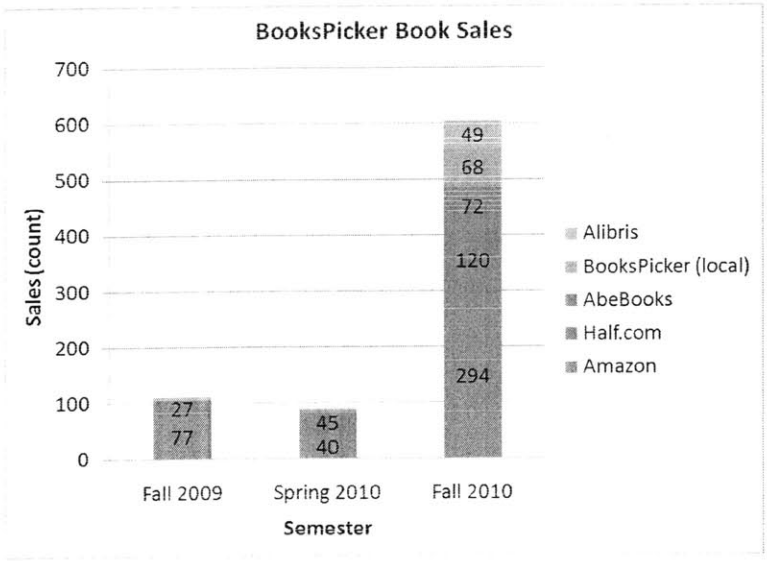

Figure 5-3: Number of books sold through BooksPicker in the last three semesters, broken down by vendor. Note that AbeBooks and BooksPicker (local) were not available in the Fall 2009 and Spring 2010 semesters. Also, note that while BooksPicker includes links to the APO Book Exchange and The COOP, we are unable to track how many book sales occur due to the traffic we provide them.

Table 5.2: Total books sold in three measured local markets. The number of books sold through mailing lists is an estimate based on the number of emails sent to mailing lists regarding books.

Local Market	Books sold
APO Book Exchange	740
Dorm mailing lists	175
BooksPicker	68

Considering local sales separately, the APO Book Exchange has clear dominion of the market. Table 5.2 reflects the number of books sold in three measured markets this semester. The APO Book Exchange leads by far, selling 740 of the 1203 books that were on sale[1]. BooksPicker accounted for about 7% of the books exchanged locally this semester (out of those that were accounted for by us). It is worth noting that the APO Book Exchange has been running for multiple years and has already built up a reputation, while this was BooksPicker's first year in the local market area. It is unclear how much of the non-local market is controlled by BooksPicker.

The number of books sold through mailing lists shown on Table 5.2 is an estimate based on the book-related email traffic seen in the dorm mailing lists. We subscribed to all the dorm mailing lists and monitored them manually to identify book-related emails. This was also used as a marketing strategy, whereby we would reply to such emails and encourage students to sell their books through BooksPicker instead. A total of 263 book-related emails were sent. Based on the estimate that about a third of those were merely notifications of a book being sold, we estimate that about 175 transactions took place. It is worth noting that two dorms (McCormick and Next House) out of eleven accounted for 252 out of the 263 book-related emails. It is a well-known unwritten rule that dorm mailing lists are not to be used for buying and selling books or other items. The practice is frowned upon by students but not enforced. It seems that once a few of these emails are sent to a mailing list, a tipping point is reached and all other students shamelessly send their own emails as well.

[1]Data is not official. It was obtained through their website by browsing their book inventory at different times.

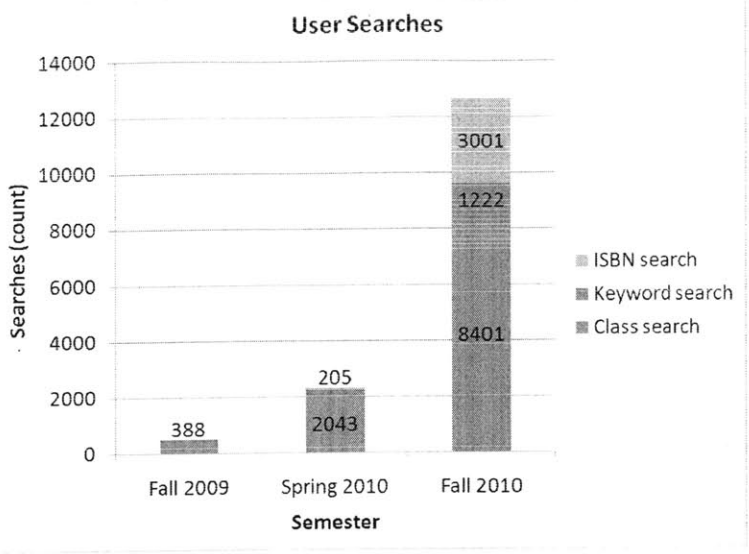

Figure 5-4: Number of searches done on BooksPicker in the last three semesters, broken down by search type.

5.1.3 Searching

To better understand how students use the site, we tracked the way they searched. Figure 5-4 shows the searches performed on the site, broken down by type. In terms of total searches, the current semester's performance was overwhelmingly better than the previous semesters'.

An interesting fact is that for previous semesters, users preferred keyword search over ISBN search (though it is hard to see in the Figure), but this semester there were more ISBN searches than keyword searches. We attribute this to the addition of book list data to the MIT course catalog, which included the book's information, including their ISBN number. This probably made it easier for students to copy and paste the ISBN numbers from the catalog over to BooksPicker.

In total, there were 4,785 bundle (or price) searches this semester. A total of 10,370 books in those bundles were searched for prices. The largest bundle searched for had 23 books, while the average bundle size was of 2.17. This means in general people do make use of the bundle search as intended, instead of searching for books one by one.

Table 5.3: Popularity of Facebook Pages of similar scope to BooksPicker's.

Facebook Page	Fans
MIT Class of 2009	35
BooksPicker	87
lbgt@MIT	106
LSC @ MIT	138
MIT Admissions	150
The Official Class of 2014	319
The Harvard COOP	340
MIT EECS	559
CSAIL	697
MIT 100K Entrepreneurship Competition	847

5.1.4 Facebook

By using Facebook Connect for user login, we can easily track how many users have logged in to BooksPicker. As of September 22, there were 185 registered (Facebook) users.

In addition to using Facebook for login purposes, we also set up a Facebook Page to get "fans" and create a following. Again as of September 22, there were 85 users who had "Liked" the BooksPicker Page. While this is not much, it seems like a decent start. For reference, the number of fans of other similarly scoped Pages are listed in Table 5.3.

5.2 Local Market

This section evaluates the performance of the local market provided by BooksPicker. Throughout this dicussion, the "BooksPicker" offer type refers to offers from the local BooksPicker market, whereas "Local" offers include BooksPicker offers as well as offers from the APO Book Exchange and The COOP (all local within the MIT campus). "Non-local" offers refer to all other offers (i.e., from Amazon, Alibris, eBay, or AbeBooks).

Table 5.4: Click-through rates (CTR) for different types of offers. The data in this table is from 9/4 onwards.

Offer type	Times shown	Clicks	CTR
All	120896 (est)	2368	1.96%
Non-local	99189 (est)	1566	1.58%
Local	21707 (est)	802	3.69%
BooksPicker	2220	236	10.63%

5.2.1 Market Effectiveness (For Buyers)

Transactions from online vendors far exceeded BooksPicker ones, as shown in Figure 5-3. However, we need to account for the fact that there are many more offers available from online vendors than locally on BooksPicker. Thus, we looked at the click-through rate (CTR) for each type of offer. The results are presented in Table 5.4. The results show that local offers in general had over two times the CTR as non-local ones, while the CTR for BooksPicker offers was over five times that of non-local ones.

CTR is not enough to compare the relative effectiveness of the different types of offers though. For instance, users seldom click on the more expensive offers, which are typically dominated by non-local offers. Thus, we considered the number of clicks per offer type, but restricted to those clicks where the offer type was actually an option for the user (i.e., the user made a meaningful choice). In addition, we track the number of times the user chose an offer that was *not* the cheapest one, so that we can calculate how much they are willing to pay, the "premium", to buy a particular offer type over another. Table 5.5 displays this data for three different types of offers. The first set of rows quantifies the total number of meaningful choices a user made (i.e., the number of times a user clicked on an offer of any type, provided that an offer of the type in question was among the options). The second set of rows quantifies the number of "successful" choices for each offer type (i.e., the number of times the chosen offer was of the offer type in question). We call the resulting likelihood of a choice being successful the "adjusted CTR" for that offer type. Note that these measurements are for *clicks* and not for actual sales.

Table 5.5: User behavior for different types of offers. Indented rows indicate a subset relation and the numbers in parenthesis indicate the breakdowns. The data in this table is from 9/9 onwards.

Variable	Offer type		
	BooksPicker	Local	Non-local
Clicks (choices) when type was an option	159	600	705
Offer was cheapest option	74 (47%)	214 (36%)	494 (70%)
Offer was not cheapest option	85 (53%)	386 (64%)	211 (20%)
Clicks on offer (adjusted CTR)	87 (55%)	225 (38%)	483 (69%)
Offer was cheapest option	57 (66%)	165 (73%)	334 (69%)
Offer was not cheapest option	30 (34%)	60 (27%)	149 (31%)
Average premium ($)	15.37	14.13	11.48

Notice that the adjusted CTR for BooksPicker and local offers are now considerably lower than that for non-local offers (55%, 38%, and 69%, respectively). However, they closely reflect the percentages of when the option was the cheapest option available (47%, 36%, and 70%). This correlation suggests that users tend to click on the cheaper offers, which is as expected.

What is more interesting are the clicks on offers that were not the cheapest option. Here, all three types of offers have similar percentages, around 30%. Yet the average premiums vary considerably. That is, whenever a user chooses an offer that is not the cheapest one, they are willing to pay more on average ($15.37) for a BooksPicker offer than for a non-local offer ($11.48). We must also keep in mind that this data is for clicks and not for actual transactions, so the data can be skewed by users who simply want to further "explore" offers before buying them.

5.2.2 Market Effectiveness (For Sellers)

To gauge the local market's effectiveness for sellers, we look at the probability of selling a book given that it was put on the BooksPicker market. In addition, we disect the data based on whether the book offer had class information or not, and on whether the

Table 5.6: Market effectiveness for sellers, measured by calculating the probabilities of their books being sold on the market. Here, a "sold" book is counted whenever an offer's buy link is clicked at least once, and a book is considered sold if such offer is never reposted. Indented rows indicate a subset relation and the numbers in parenthesis indicate the breakdowns. The data in this table is from 9/4 onwards and only considers offers that were on the market for over one hour.

Metric	Total (%)	"Sold" (%)	Sold (%)	P("sold")	P(sold)
Books on local market	373	60	52	16.09%	13.94%
Without class data	88 (24%)	1 (2%)	1 (2%)	1.14%	1.14%
With class data	285 (76%)	59 (98%)	51 (98%)	20.70%	17.89%
Accurate data	93 (33%)	38 (64%)	33 (65%)	40.86%	35.48%
Inaccurate data	150 (53%)	20 (34%)	18 (35%)	13.33%	12.00%

class information was accurate or not. Here, an offer's class information is accurate if the book in the offer is part of the class' book list according to BooksPicker's database.

Given the nature of our service, we cannot track the number of *real* book sales. We can only track when a buyer gets in touch with a seller, which we call a "sale" (with quotations). Also, note that since transactions may fail in real life, sellers might repost their offers. Consequently, the same offer can be "sold" more than once. Since a seller is ultimately interested in selling books and not offers, we do not double-count offers that are "sold" multiple times. Instead, we only consider offers for books that sold at least once. In addition, we assume that if the seller does *not* repost their offer, it means that the transaction complete successfully. We consider the number of "sold" and not reposted books as an estimate for the number books actually sold. Table 5.6 summarizes the results found. To minimize error, we only considered offers that were on the market for over one hour.

The results show that the probability of selling a book through the local BooksPicker market is around 15%. This is lower than expected and considerably lower when compared to the APO Book Exchange. As discussed in Section 5.1.2, the APO Book Exchange sold about 740 books out of 1203, which corresponds to a sale probability of 62%, four times better than BooksPicker's. Of course, we must keep in mind

Table 5.7: Conversion rates for BooksPicker and non-local sellers.

Seller	Conversion rate
BooksPicker	28.81%
Amazon	40.27%
Alibris	19.91%
Half	18.66%
AbeBooks	17.02%

that the prices at the APO Book Exchange are considerably lower.

While the probability of selling in general is low, books that are accurately related to a class fare much better. At a 35-40% selling probability, they approach the APO Book Exchange's 62% rate. This is expected, since most of the user searches are class-based, so a book not associated to a class is less likely to be found by the user. Indeed, only one book with no class information was sold. Even if the class information is inaccurate, the probability of selling the book while providing class information is over 15 times better.

5.2.3 Conversion Rates

Beside looking at CTR and adjusted CTR, we should compare the conversion rates between the local and non-local markets. That is, the number of completed transactions per clicks on the 'Buy Offer' link on the search results. For these purposes, we do count as conversions instances in which the buyer gets in contact with the seller, since the buyer expressed genuine interest in buying the book, even if the transaction does not go through in real life due to external reasons.

The conversion rates for BooksPicker as well as the non-local vendors are listed on Table 5.7. We cannot measure conversion rates for the APO Book Exchange or The COOP since we do not know how many of their sales were caused by our referring traffic. Note that, as expected, BooksPicker's conversion rate is higher than those of non-local sellers, with the exception of Amazon. In fact, Amazon's conversion rate is over twice that of its competitor non-local sellers. There a few

Table 5.8: Evaluation of automatic pricing option. Indented rows indicate a subset relation and the numbers in parenthesis indicate the breakdowns. The data in this table is from 9/4.

Metric	Count (%)	Avg days on market	Avg selling price ($)	Prob of being "sold" or sold
Books on market	373	N/A	N/A	N/A
Fixed price	158 (42%)	N/A	N/A	N/A
Auto price	215 (58%)	N/A	N/A	N/A
Quickie sale	148 (69%)	N/A	N/A	N/A
Money lover	67 (31%)	N/A	N/A	N/A
Books "sold"	60	2.10	41.14	16.09%
Fixed price	22 (37%)	1.71	47.45	13.92%
Auto price	38 (63%)	2.33	37.48	17.67%
Quickie sale	33 (87%)	2.37	36.82	22.30%
Money lover	5 (13%)	2.03	41.84	7.46%
Books sold (not reposted)	52	1.85	38.40	13.94%
Fixed price	21 (40%)	1.58	43.52	13.29%
Auto price	31 (60%)	2.03	34.93	14.42%
Quickie sale	27 (87%)	2.10	32.51	18.24%
Money lover	4 (13%)	1.53	51.20	5.97%

possible explanations for this: trust, the fact that Amazon is offering free shipping for students, or differences in the way Amazon does its referral tracking. However, we cannot easily prove or disprove any of these explanations.

5.2.4 Automatic Pricing

To evaluate the success of our automatic pricing option, we focus on three metrics: probability of making a sale, average selling price, and average time on market. We calculate these metrics for the entire local BooksPicker market, as well as breaking it down depending on whether the offer is using automatic pricing or not. We use the same conventions as in Table 5.6 to estimate sold books. The results are summarized in Table 5.8.

In terms of probability of making a sale, the results are as expected. The prob-

ability of selling a book after choosing the Quickie Sale automatic pricing strategy gravitates around 20%, which is considerably higher than the probability of selling with a fixed price (~13.50%). However, the results also show that the Money Lover strategy was ineffective at selling books.

These selling probabilities come with a tradeoff, of course, which is price. Note that, as expected, the average selling price for offers using the Quickie Sale strategy is about $10 lower than that of books that have a fixed price. Conversely, offers using a fixed price or the Money Lover strategy sold for higher prices.

The third measure, average time on the market before selling, has interesting results. Note that Quickie Sale, the strategy that is supposed to sell books fast (and cheap), resulted in higher average times on the market than the alternatives. We believe this discrepancy is due to the book's popularity. The more popular the book is, the faster it will sell, thus bringing the time on market average down. Indeed, after measuring the number of searches for books with fixed price and those using the Quickie Sale strategy, we found that on average, books using fixed prices were twice as popular as books using automatic pricing. A possible explanation for this fact is that due to their popularity, students have a good idea of the book's value and opt for setting a fixed price, while only resorting to automatic pricing for the less popular books. But regardless of the cause, if we assume a linear relation between popularity and time on market, this means that after accounting for popularity, Quickie Sale actually sells books faster than fixed prices.

5.2.5 Offer Prioritization

Unfortunately, there was not enough usage in the local BooksPicker market to make substantial use of the offer prioritization mechanism. There were only a few instances of different copies of the same book being sold at the same price. The mechanism only kicked in about a dozen times, and social and location information was often not available for those cases.

5.3 User Feedback

This section summarizes the results obtained from user surveys.

5.3.1 Overall Market

One of the key measures we are interested in is total money spent on books, since it is our goal to drive this number down. However, we only have data on money spent for the Spring 2010 and Fall 2010 semesters, and the book markets in the Spring and Fall semesters behave very differently. In general, there is much more book market activity during Fall semesters, resulting in more money being spent on books. It is unclear why, but a theory is that during the Spring there are more books available locally. First, freshmen who did not know each other at the beginning of the Fall semester can now trade books with each other for the Spring semester. And second, rising seniors have less people to buy their books from in the Fall semester since the older class has already graduated and left MIT. According to the surveys, the average money spent on books per person was of $183 in Fall 2010 and $171 in Spring 2010. It is unclear whether this $12 difference is significant and whether it even makes sense to compare money spent across Fall and Spring semesters.

In terms of the biggest pain when selling books, we found that pricing became less of a problem. Figure 5-5 shows the pains associated with the book-selling process. Comparing these results to Figure 2-3, we see a sharp drop in the pricing problem, which roughly drops from accounting for 34% of the responses to 24%. The other pain points remain largely the same.

5.3.2 Buying on BooksPicker

According to the survey, only 29% of BooksPicker users bought books through it. Figure 5-6 shows the students' responses as to why they did not buy books through BooksPicker. The most important reason was that BooksPicker was too expensive. Since BooksPicker already includes the major online and local book vendors, it is unclear where these students are getting cheaper prices from. We must work harder

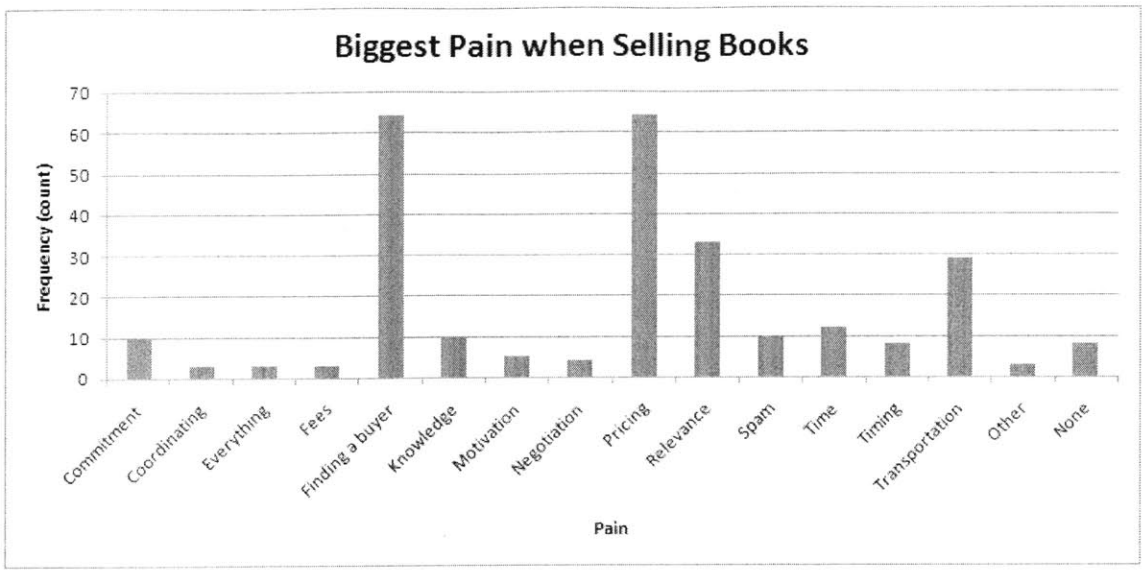

Figure 5-5: Report of students' biggest pain when selling books. To clarify: Commitment refers to finding buyers that actually follow through with their promises; Coordinating refers to setting up a meeting time and place with the buyer; Fees refers to the fees charged when selling online (e.g., though Amazon); Knowledge refers to not knowing where or how to sell; Negotiation refers to dealing with people who want to negotiate prices further; Pricing refers to both choosing a sell price and to selling the books too cheap; Relevance refers to the difficulty of selling books that are outdated or no longer needed for a particular class; Spam refers to book-related emails on mailing lists; Time refers to the process taking long to complete; Timing refers to waiting for the right time to sell books. This was an open-ended question summarized manually. Total number of responses: 269.

in getting more sellers and penetrating the local market, since typically most of the lowest prices come from there.

In Section 5.2.1, we established that users choose to buy a book other than the cheapest one about 30% of the time. Figure 5-7 shows that the book's condition was the most important factor in such decision. Yet according to the initial survey (shown in Figure 3-9) and discussions with the focus group, the book's condition does not matter much to students. It seems that despite what they claim, students are are still willing to pay a premium for a better book. As expected, the second most popular reason was students' preference to buy locally.

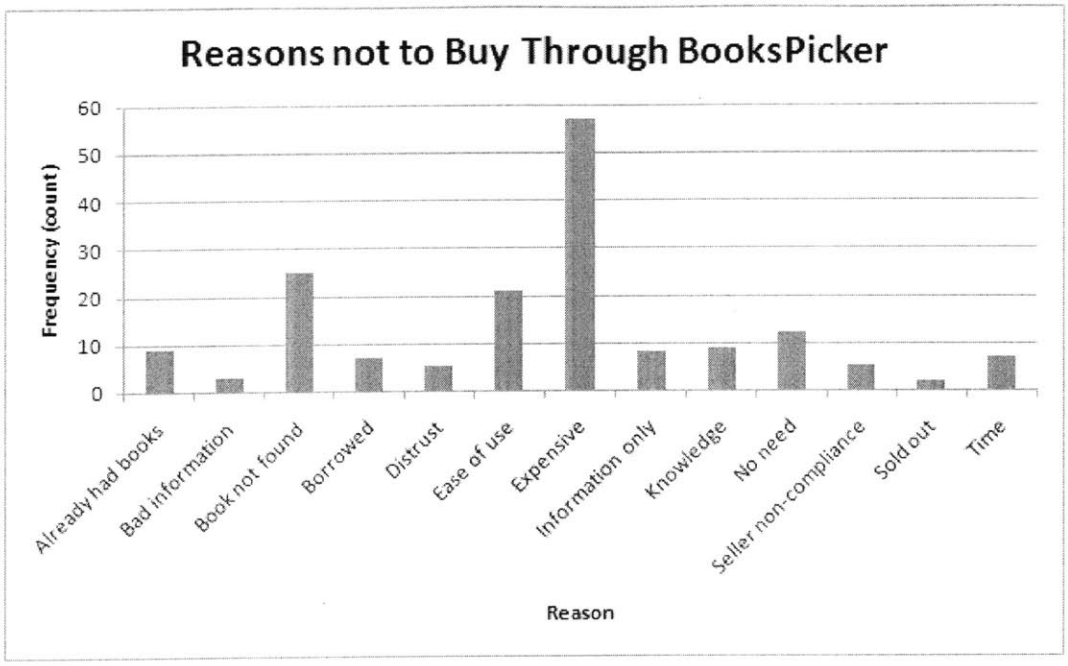

Figure 5-6: Student report on why they did not use BooksPicker to buy books. This was an open-ended question summarized manually. Total number of responses: 185.

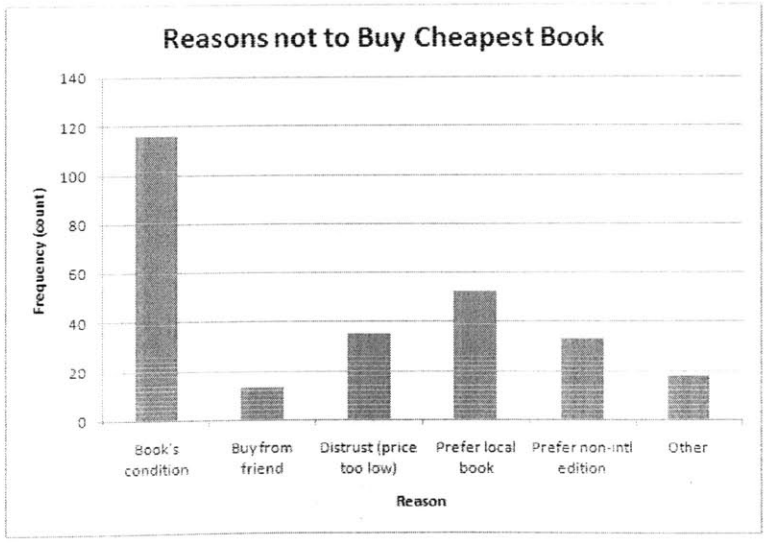

Figure 5-7: Student response on why they did not always choose the cheapest book to buy. Total number of responses: 267.

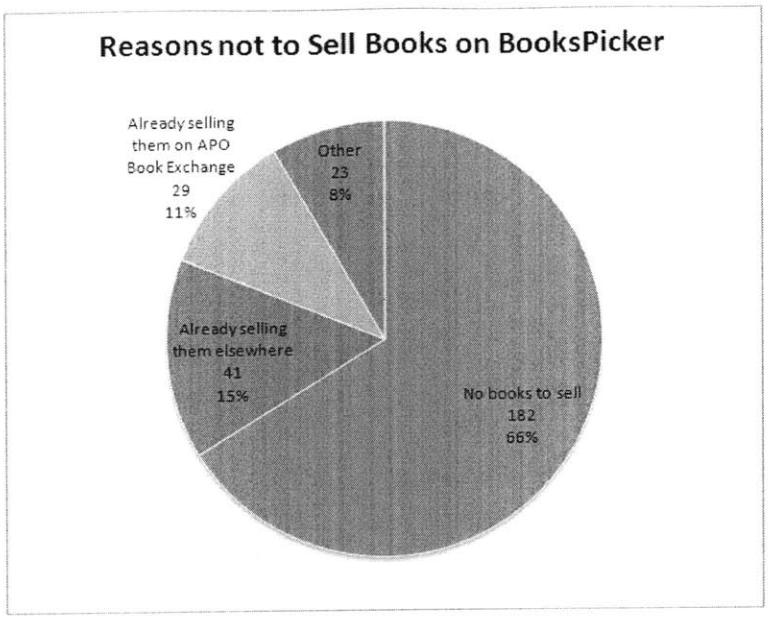

Figure 5-8: Student report on why they did not use BooksPicker to sell books. Total number of responses: 277.

5.3.3 Selling on BooksPicker

The percentage of BooksPicker users that used it to sell books is 6%, which is much lower than the 29% that used it for buying. The reasons why this is the case are shown in Figure 5-8. Note that apart from not having books to sell, students did not sell on BooksPicker because they were already selling their books elsewhere (26% altogether). That means there is still a large portion of the local market that BooksPicker can penetrate.

With regards to automatic pricing, 58% of those who tried to sell through BooksPicker tried doing so by using automatic pricing. Unfortunately the number of responses is too low to gather significant insight as to why users chose not to use it. Respones varied, but included that they did not trust it, they knew which prices to set, they preferred to round the price to the nearest dollar, or that the price was set to be too low or too high.

Chapter 6

Conclusion

BooksPicker provides an all-in-one solution for students that need to buy or sell textbooks. The principal focus of this thesis was on improving the textbook exchange system at MIT by introducing an efficient local market through the use of technology. This thesis makes three main contributions. First, it integrates a local sales system into BooksPicker, making BooksPicker the first tool that combines both local and online offers together in one place. Second, it introduces automatic pricing strategies that dynamically price a book according to current market and physical conditions in order to maximize the likelihood of making a sale at a reasonable price. Third, it provides an offer prioritization mechanism that uses social and location information to determine which offers are displayed to buyers first in order to increase the likelihood of a sale.

During the trial period of the Fall 2010 semester at MIT, we saw increased adoption of the tool relative to previous offerings. It is clear that there was a need in the market, since usage increased three-fold and number of transactions increased six-fold over the previous semesters. However, the local BooksPicker market only accounted for 11% of those transactions, leaving much room for improvement.

The automatic pricing mechanism performed reasonably well. We found that the probability of selling a book while using the Quickie Sale automatic pricing strategy is considerably higher than that of selling with a fixed price (roughly 20% to 13.5%). Moreover, the expected time on market after normalizing for the book's popularity

decreases by around half a day when using automatic pricing. And while pricing is still one of the biggest problems students face when selling books, its impact decreased by about 10 percentage points from Spring 2010 to Fall 2010.

Lastly, we learned that some of the implemented features, such as offer prioritization and anti-abuse quality control, were mostly unused. We could not gather enough data to properly evaluate their performance, as their mechanisms only kicked in about a dozen times. Further work on these features should be de-prioritized until usage increases enough to garner them enough relevance.

6.1 Future Work

BooksPicker currently only controls under 10% of the measured local textbook market. It is important that BooksPicker achieves "critical mass" in order to deliver its full potential. We believe that features like automatic pricing and offer prioritization would benefit from a more saturated market, in addition to being able to offer lower prices due to increased competition. Apart from improving marketing and advertising to increase adoption, we believe that immediate future work should be focused on the three main areas described below.

Usability

Usability is extremely important to provide a smooth experience and help increase adoption. Some enhancements that would probably be helpful include: making it easier to input books for sale (currently the user must input the book via its ISBN), allowing multiple classes to be a associated to a book on sale, allowing buyers to cancel after they have clicked to buy a book, including the data source for a class' book list, allowing users to login thourgh some mechanism other than Facebook (e.g., MIT Certificates or a traditional login system), and enabling users to sell non-book items such as TEAL clickers[1].

[1]Technology Enabled Active Learning (TEAL) is a teaching approach used at MIT to teach some introductory level physics and math courses. In TEAL classes, students may require a "clicker" to submit answers to questions posed during lecture.

Better Data

Most searches on BooksPicker are class searches, and we also found that books associated to classes sell with much higher probability than those that are not. This shows most users heavily rely on BooksPicker's data, which is why it is extremely important to provide a complete and accurate dataset. Our current data sources, MIT and The COOP still have discrepancies and incomplete data. We should continue to push MIT to improve its book list data, as well as improving our own crowdsourcing efforts through more adoption and by allowing students to input listings without having to sell a book (e.g., when they buy a book that is not currently associated to any class). Also, it may prove useful to have a system where not only can students add new listings, but they can also upvote or downvote existing listings to validate them.

Sustainability

Last but not least, a sustainability strategy must be developed to ensure the project will not die once its developers (author et al) leave MIT. We believe that first and foremost, effort should be put on making the system itself easy to maintain. This is achieved through careful code design, extensive documentation, and the automatization of most tasks (e.g., a data collection task that runs every x hours and updates book list data accordingly for the current semester). And second, we should explore the possibility of giving responsibility over maintenance to a campus-based group, such as APO.

6.1.1 Other Ideas

This section briefly describes longer-term ideas that go beyond being enhancements to current functionality. Ideas are described in no particular order.

Textbook Feedback Provide feedback on books as they relate to specific classes. The goal of such feedback is to help students decide whether they actually need to buy a book or not, regardless of its necessity according to the professor. At

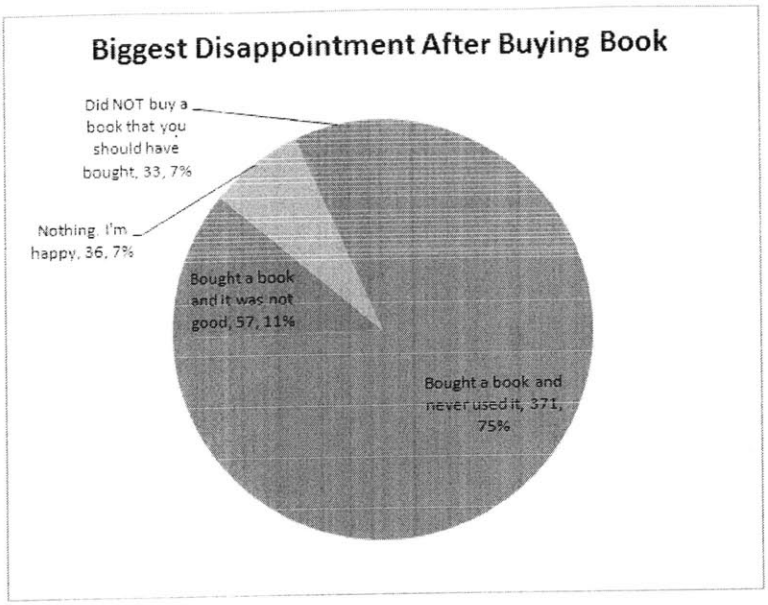

Figure 6-1: Biggest disappointment after buying books.

MIT, it is frequently the case that required books are not utilized by students. The first survey's results shown in Figure 6-1 confirm this, with the biggest disappointment after buying a book being lack of usage.

Substitute Textbooks In many cases, a student may be well off with a different version of a given book that has largely the same content, such as a previous edition or an international edition. We should automaticaly identify these similar books and offer them to students as alternatives.

Library Integration Many of the required textbooks are actually available in libraries for free. Some are only available 'on reserve,' which means students cannot check them out, but others are not. We should integrate library availability into BooksPicker's book search, so that students refrain from buying books that are available in libraries or affiliated resources such as books24x7. This is most useful when linked to the college's own libraries, but eventually it could incorporate public libraries as well.

Filters Users should be able to filter their searches based on stores they are comfortable buying from, distance (for local sales), condition, number of stores used,

and other similar factors.

Collaborative Shopping Enable students to combine their book bundles with their friends in order to make use of discounts (especially bulk discounts). Incorporating bulk discounts is challenging since most of them are given by bookstores on a case by case basis, and the buyer needs to contact them for a quote beforehand.

Browsing Allow users to browse data instead of having to search for it. This enhances discovery and opens room for some useful use cases. For example, students wanting to see which books are being sold in their dorm and neighboring dorms.

Alerts Users should be able to subscribe to an alert for a given book, so they can be notified once that book becomes available or if its price drops beyond a certain threshold.

Bibliography

[1] Abe, N., Kamba, T., A web marketing system with automatic pricing, NEC C&C Media Res. Labs, Miyazaki, Japan.

[2] "Book Exchange Information Sheet," Alpha Phi Omega, Online document, Accessed 2010 May 21, Available URL: http://web.mit.edu/apo/www/policy-guide.pdf.

[3] "Fees and Pricing," Amazon.com, Online document, Accessed 2010 May 21, Available URL: http://www.amazon.com/gp/help/customer/display.html?nodeId=1161240.

[4] "Frequently Asked Questions About 2010-2011 Financial Aid Program Adjustments," MIT Student Financial Services, Online document, Accessed 2010 May 21, Available URL: http://web.mit.edu/sfs/financial_aid/FA_FAQ_2010-11.html.

[5] Levin, S., "Belltower Books employs Bowdoin students, irks others," 2010 Jan 29, Online document, Accessed 2010 May 21, Available URL: http://orient.bowdoin.edu/orient/article.php?date=2010-01-29&section=1&id=7.

[6] MIT Data Warehouse, Accessed 2010 June, Available URL: http://ist.mit.edu/services/business/warehouse.

[7] "MIT Online Subject Evaluation Pilot," Online document, Accessed 2010 May 21, Available URL: http://web.mit.edu/subjectevaluation/.

[8] Reynolds, R., "Digital Textbook Sales in U.S. Higher Education – A Five-Year Projection," April 2010, Xplana, Available URL: http://blog.xplana.com/wp-content/uploads/2010/04/DigitalTextbooks_Report_04-19-10.pdf.

[9] The Coop Textbook Store, Accessed 2010 May 21, Available URL: http://mitcoopbooks.bncollege.com.

[10] Tucker, C., Course notes for 15.818 - Pricing, offered at MIT Sloan in Spring 2010, Online documents, Accessed 2010 July, Available URL: http://stellar.mit.edu/S/course/15/sp11/15.818/.

[11] "Undergraduate Expenses," MIT Student Financial Services, Online document, Accessed 2010 May 21, Available URL: http://web.mit.edu/sfs/afford/undergraduate_expenses.html.

[12] "Why we're better," BigWords.com, Online document, Accessed 2010 May 21, Available URL: http://www.bigwords.com/index.php?z=whywerebetter.