COMPASSION BY THE BOOK

Client: Kurtis Griess, Ian Littman

Team: Kira Combs, Anastasia Shpurik, Andrew Towe

Start Date: May 13, 2013

End Date: June 19, 2013

Contents

Introduction

Our client, Kurtis Griess, is the Executive Director of the non-profit organization Compassion by the Book. Mr. Griess started this company when he began to sell his own textbooks. He learned from his peers that most people merely hold on to their textbooks and most of them are never used again. Using this vision, he decided to start this organization that would put the profits of selling these would-be untouched books to a good use.

As aforementioned, Compassion by the Book takes donations of gently used textbooks and sells them on a third-party website. The funds raised from these sales support causes such as the Red Cross, Habitat for Humanity, or other non-profit organizations. Currently, in order to list and sell a book, volunteers must type in information about the book into a shared Google drive spreadsheet, for bookkeeping, and then insert that same information on the Half.com seller's pages. These spreadsheets are very numerous and difficult to query.

The purpose of the project was to provide volunteers of Compassion by the Book with an online tool to help it handle management of the multiple causes, organizations, schools, users and books. This tool must also have tight interconnections with the website Half.com, which is where the donated books are sold. This interconnection must be able to manage the listing, inform the user when a book has been sold and print out the shipping label. This ultimately gives Compassion by the Book a powerful tool that streamlines the process of listing, selling and shipping primarily into one site that requires significantly less interaction with the Half.com interface.

It is important for the new website to allow volunteers to log books into the database and compare prices, list items, and suspend items on Half.com. A user should also be able to print shipping labels once the item has been sold. Lastly, our site must allow for expansion and integration of recordkeeping so that our client can view statistics of schools, organizations, causes, and users.

Requirements

Functional Requirements

The goal of this project was to provide Compassion by the Book with a tool to help them manage their organizations more efficiently and sell their collected books to help these organizations raise funds for specific causes. This project entailed several tasks:

- Design a website that is capable of handling multiple organizations with multiple users, each with different levels of access.
- Design a database to keep track of all of the organizations, causes, users and books.
    - The database should also keep track of each book's value and the current status of the book (logged, listed, sold, label purchased, shipped).

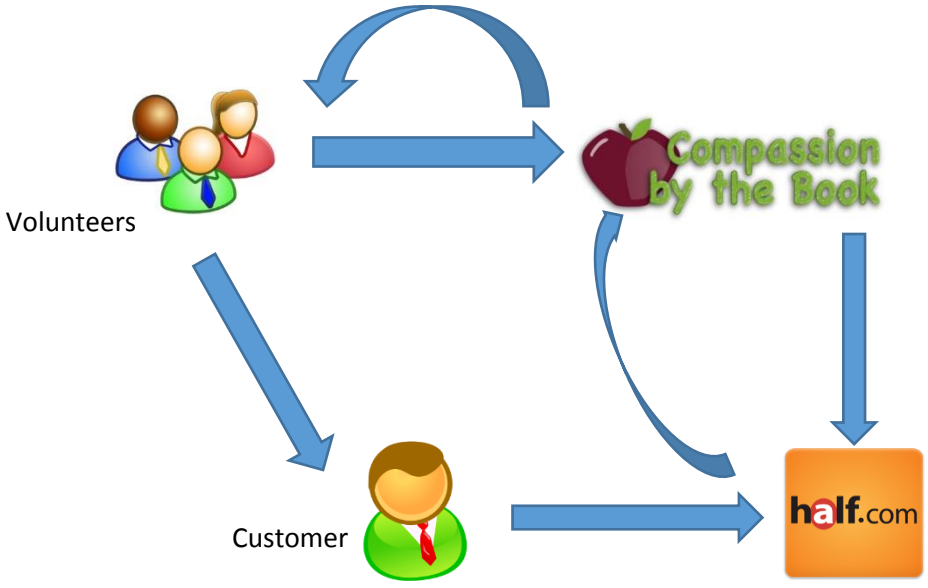
        - It should be relatively easy to query.
        - All tables must be interdependent with each other.
    - Utilize the API system from Half.com to check the value of each book, per condition; list, un-list, and manage the book; and also actively check whether the book had been sold.
    - We must be able to retrieve the weight, packing slip, and shipping label. The user should be able to print these labels from our website.
    - Have a system set up that constantly checks whether a book has been sold or not, and have some way to alert volunteers of this status.
    - Handle the difference between books that are sold online versus books that are considered free book fair books
    - Keep track of the number of books per organization

Non-Functional Requirements

- The project should allow for expansion should volunteers sell on an additional site.
- Code must be well documented and commented to allow future users to discern the functionality.
- Make a system that handles the different access levels of different user

Architecture

General Flow



The main objective of this project is to have communication between all of the above entities (the volunteers, the Compassion by the Book website, Half.com's website, and the customer) flow smoothly.

Volunteers have different statuses and thus have different abilities in regards to what they can do with a collected book.

A book may be logged on the Compassion by the Book website. Once logged, this book can be either listed or placed into a 'Free Book Fair' category, meaning that the book is not worth enough to be sold. If the book is listed, this listing can then be both edited and un-listed. As the book is sold, a shipping label and packing slip can then be printed.

When a book is 'listed' on the Compassion by the Book website, the website lists the book on Half.com. Once listed, it can be seen by perspective customers looking for that book. When a customer buys a book, the details of this purchase are sent back to our website so that the volunteers can have updated information and perform different actions. Because of this system, volunteers are able to satisfy any requirement needed to perform a transaction with a customer.

Technical Design

Website Design

Diagram of Volunteer Interface

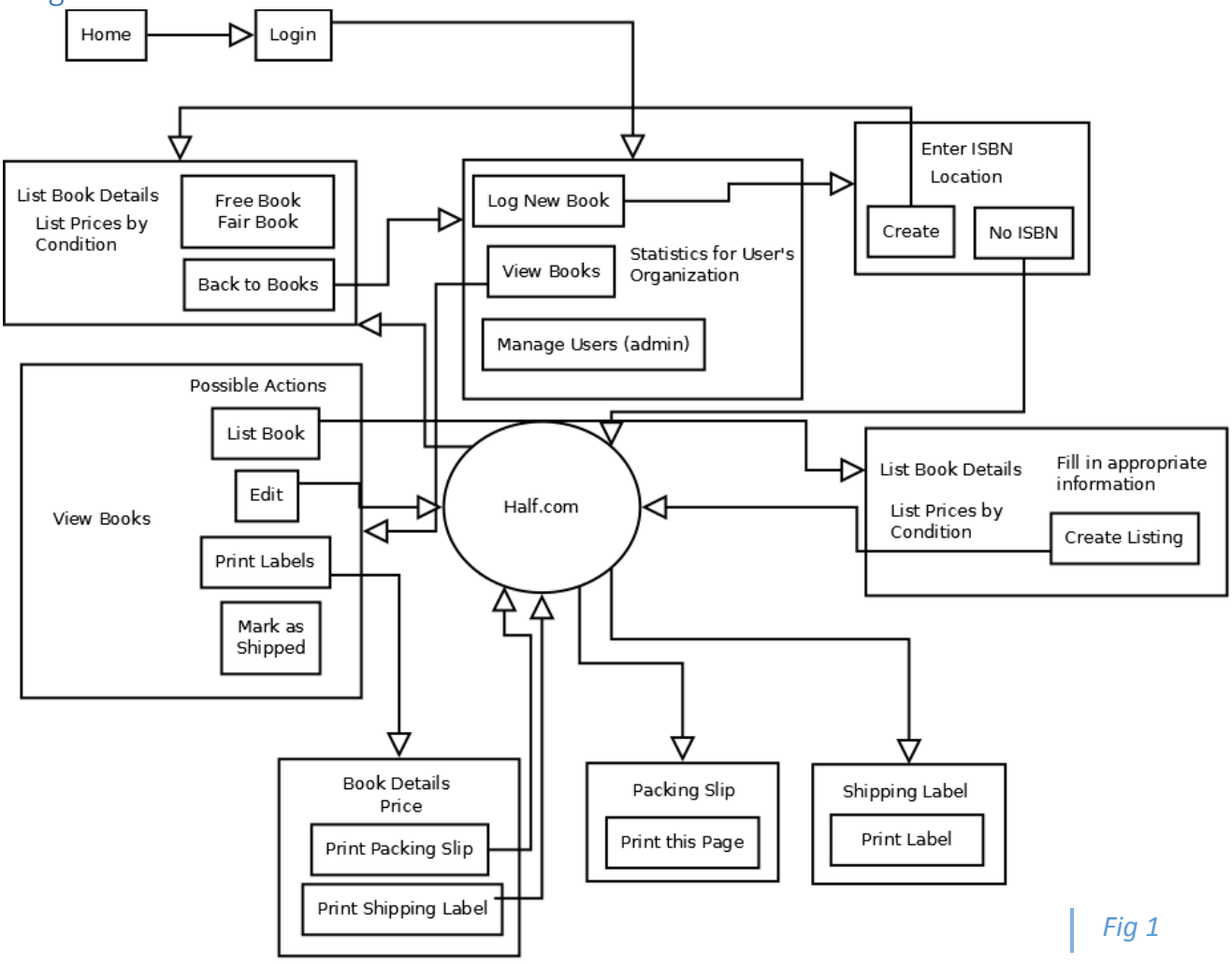


Fig 1

How the Volunteer Interface Works

Each Volunteer's abilities differ depending on the volunteer's access. Permissions are set in a tiered system, so that each level higher gains new abilities, but are still able to perform any actions from a lower tier. Details on the flow of the website for volunteers can be seen in Figure 1.

- If the volunteer is a Logger they are able to:
  - Log a book into the system
  - Look up ISBN's of a book
  - Decide whether the book is a Free Book Fair book
- If the volunteer is a Lister they are able to:
  - View all books associated with their organization
  - List a logged book on Half.com
  - Edit a currently existing listing
  - Un-list/Suspend books
- If the volunteer is a Shipper they are able to:
  - Print shipping label for a sold book
  - Print the packing slip for a sold book
  - Mark sold books as shipped
- If the volunteer is an Admin they are able to:
  - Create new users for their organization
  - Add to the number of unlogged books for their organization

Diagram of Super User Interface

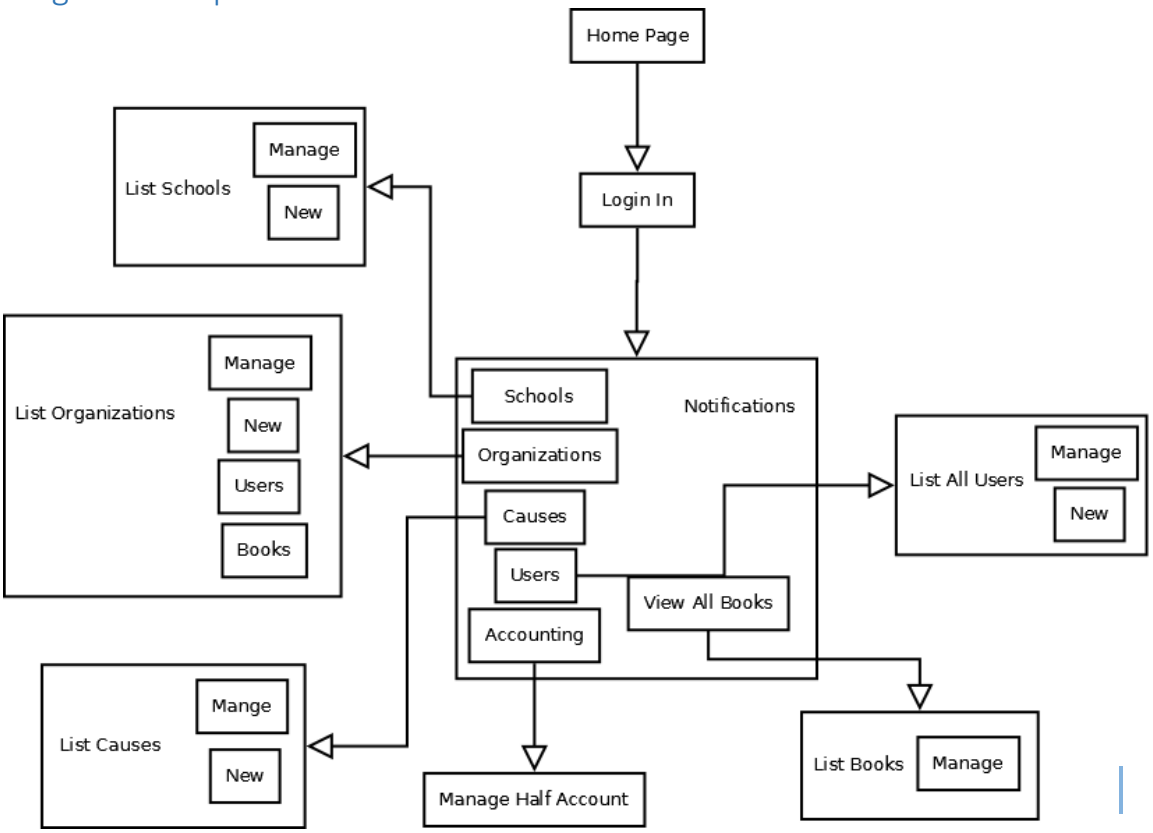


Fig 2

How the Super User Interface Works

The Super User of the website has a slightly different view and setup. The Super User is the only user with the ability and responsibility of creating new schools, causes, and organizations. The Super User also manages which causes belong to a certain organization which belongs to a certain school. The setup for these three are very interdependent.

The Super User must first create a school. Then, the Super User may create an organization for that school. Should this organization already exist, the Super User can choose from an existing organization or create a new one, but he *must* associate an organization with an already existing school. Similarly, when creating a cause, the Super User can choose from an existing cause or create a new one, but he *must* associate a cause with an already existing organization. Once these associations are made, the Super User may declare an admin, or multiple admins, for any organization.

The Super User is the only user able to handle all inventory at any instance in the listing and selling process. Finally, the Super User has the ability to see how many books have been logged, listed, sold, and shipped for any given organization, as well as viewing which users belong to which organization.

A graphical representation of the flow of the website for the Super User can be seen in Figure 2.

Relationships

Database Design

The information from the website is stored in a relational database. As can be derived from the diagram, the center for accessing the other tables in the database is organization. Organizations are related to causes by a join table in our database called organizations_schools(though not depicted in this diagram).

All users except the Super User are associated with an organizations_schools_id. A user is able to log books, and a book has only a user_id. When a book is listed, a listing is created with a book_id and a listing_id. Once this book has been sold, an order is created. This order has a listing_id and an order_id.

Organization also has a relationship with causes and material orders. While an organization can only have one cause at any given time, different organizations may share the same cause, and thus we have a second join table, called causes_organizations, to track which books were sold by a cause associated with a specific organization.

Relationships between all tables can be derived from the diagram. These relationships include many-to-many, one-to-one, and one-to-many.

The final relational diagram for our website can be viewed in Figure 3.

Entity Relationship Diagram

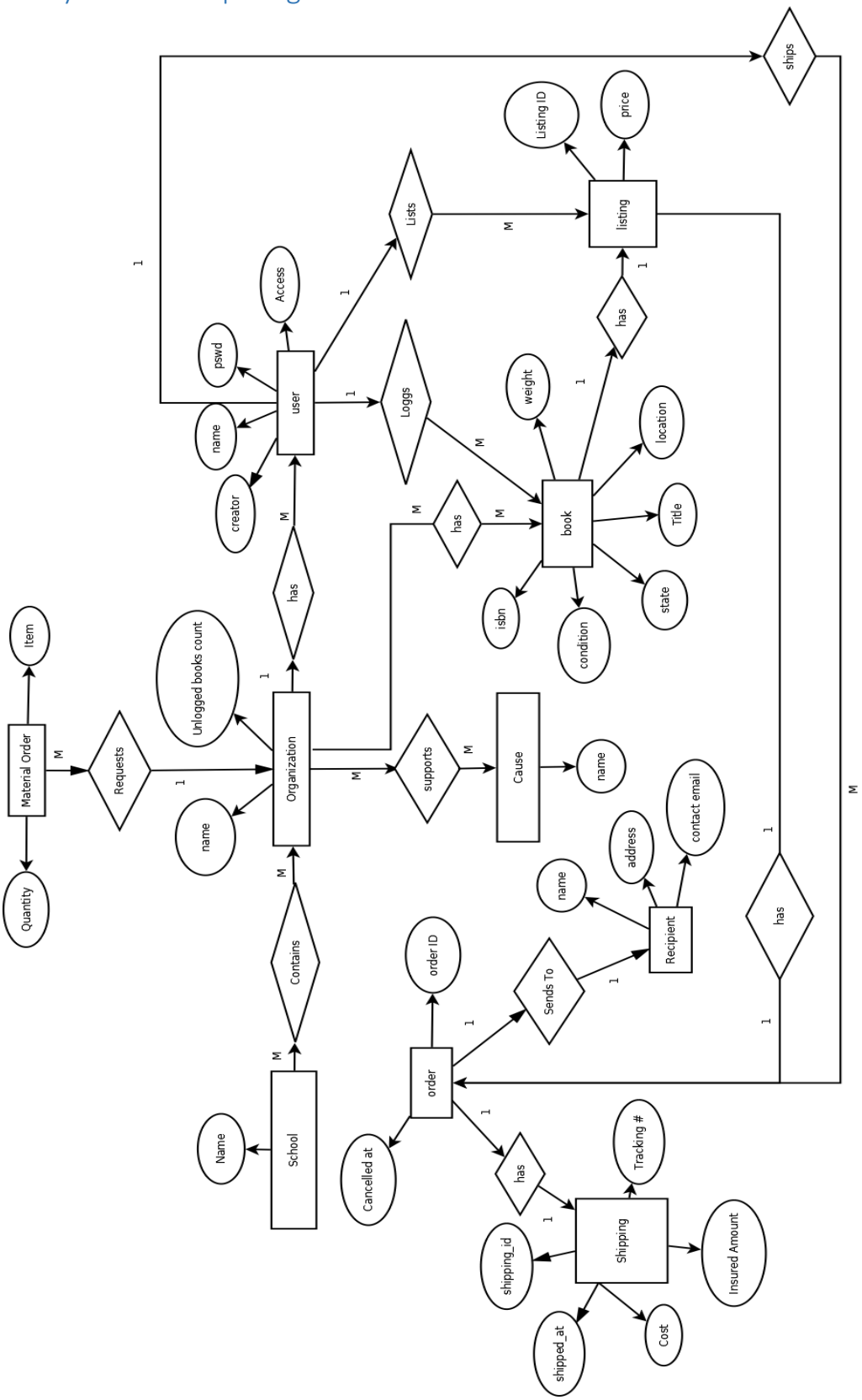


Fig 3

Design Decisions

Ruby on Rails over PHP

Two of our team members were quite familiar with Ruby, and of the two one had taken a semester-long course using Ruby on Rails. As there is a lot of documentation of Ruby on Rails and many interactive tutorials to give users the basics on how it works. As none of our group members were familiar with PHP or other web programming languages, we decided to code with Ruby on Rails to allow us more time to code rather than learn a new language.

Apache over nginX

Apache and nginX are both web servers designed to allow coding with Passenger for Ruby on Rails. We had initially tried to use NginX, as we had heard it was a better web server capable of handling more than Apache, but we had difficulty running and integrating it with rails. After enough struggle to understand this new product, we decided as a team to run Apache. It was the more common of the two, had better technical support, and we were more familiar with how it functioned.

Postgresql over Mysql

One of our team members were very familiar with the Postgresql database, and had been using it for over half of a year. Postgresql is a larger and faster database compared to Mysql, and as our website would have to be able to eventually support substantial amounts of data, we decided Postgresql would be the better choice.

Cancan over Declarative Authorization

Both of these authorizations are gems offered by ruby. CanCan is exponentially easier to understand and simpler to use. The authorization is also very customizable and easy to alter and change.

HTTP GET and POST over Soap

HTTP GET is simpler to use and is still able to communicate with the Half.com API. HTTP GET is lighter, less complicated, and easier to utilize.

Use of Twitter bootstrap

The twitter-bootstrap was actually a request we received from our client in order to alleviate some of the work required in rolling our own CSS. It is a fully customizable architecture with a pleasing interface and familiar function for most users.

Use of Mechanize Gem

While most of the communication with Half.com was done using the API, we came to realize that some of the required functions for our project were not included within the scope of the API. As such, we were forced to find an alternative, and found that the Mechanize gem was able to scrape screens that required us to log in first.

Reprint rather than purchase labels

Initially, we had attempted to use mechanize to scrape the print label screen, purchase postage, and print the label from our website. However, this became a severe issue when we found that the security

settings on Half.com prohibited us from utilizing the screen's Javascript. In order to purchase postage, we must first calculate the cost of the postage after inputting the weight. However, even when we tried to utilize the site's Javascript from our own machine, the website prohibited us from doing so. Therefore, we decided that in order to meet the requirements we would allow users to reprint the label from our site instead.

Results

Completed Work

At the conclusion of our project, users may appropriately manage users, books, organizations, causes and schools. We have tested the different views of the website by logging in as each of the different tiered users. Fully tested functionality includes:

- Loggers are capable of logging books according to their ISBN numbers and are capable of looking up a book on Half.com if no ISBN is found. Loggers are required to assign the physical location of the book so that other users may know where to locate it, but this field may be changed by any user should the book location change.

- Listers are capable of listing a book on Half.com. Condition, price, description, and comments are all properly assigned on Half.com when the listing is created are displayed on Half.com.

- Shippers are able to reprint shipping labels, if the label has been purchased by the Super User prior, and mark a book as shipped.

- Admins are able to create any user from a lower tier users, as well as another admin, but is unable to create a Super User.

- The Super User is able to create new users, schools, organizations, and causes.

    - User can view these relationships a view certain statistics for them where applicable.
    - User also has the ability to manage the Half.com account and update the tokens needed for making an API request.

- The fully functioning daemon requires the manager of the server to start it, but once started it will continuously run and update as needed.

Suggested Future Work

- Create an accounting page for handling the amount of money raised for each organization
    - This accounting page should be able to manage price depending on the cost of the shipping label, and the cost of taxes.
    - The main page should be customizable; able to choose between total funds raised or to break it down further to by cause, by school, and by organization separately.

- Create the ability for The Super User to manage multiple Half.com accounts in order to allow organizations new to Half.com to sell books without damaging the feedback of the main account should something go awry.
- Adjust the CSS to make the website match the main Compassion by the Book site more closely.
- Create an E-mail alert system that will redirect the notices received from Half.com to shippers so they are notified with a book has been sold and requires shipping.
- Gain permissions from the Half.com or Paypal site in order to allow our site access to the dynamic Javascripts.
- Convert API logic from specific forms and the logic that utilizes the Mechanize gem into the model and controller of their respective pages for code optimization and better convention.

Appendices

Main README

(This file is included within the main folder of the Project)
== README==
Software Developers: Kira Combs, Anastasia Shpurik, Andrew Towe
This README would normally document whatever steps are necessary to get the application up and running.

This project runs using the following version of rails and ruby:
* Ruby version: ruby 1.9.3p194 (2012-04-20 revision 35410) [i686-linux]
* Rails version: Rails 3.2.13

** System dependencies**
When the website uses a new gem, these gems must be included within the Gemfile in order for the project to run properly. To set up the box initially, please run the following command from the server while you are within the project folder:
sudo bundle install

As a developer's note, some gems such as Nokogiri had many dependencies that would prevent the bundle install from completing properly. If this becomes a problem, the failed install will tell you what you need to get the gem up and running.

As another note, in order to use and view this website, an server must be used. This server is variable, and just depends on the environment the user wants to work in. Our team used Apache with Passenger. Full documentation on Passenger with Apache can be found at
http://www.modrails.com/documentation/Users%20guide%20Apache.html
For quick installation of installing Ruby-on-Rails with Apache with Passenger on Ubuntu:
http://nathanhoad.net/how-to-ruby-on-rails-ubuntu-apache-with-passenger

Configuration
config/routes.rb contains information about the pages on the website. When adding a new page, the name of the page must be added to the routes.rb file in order for the website to find the correct path and function.
Most pages that are added during the course of this project are located in the app/views/pages folder. There is a README in that folder containing information about how the scripts used to communicate with Half.com.

Database creation
The database behind this application is Postgresql.
Changes can be made to the schema by generating a new migration and putting in the changes that must be made.

Run the following command to set these changes in development mode
rake db:migrate

In order to convert the database to production mode, the following command must be run:
rake RAILS_ENV=production db:schema:load
However, note that the development database must be updated before using this command. This command will also delete any data that currently exists in the production database

In order to populate the database, with the superuser data, the following command must be run:
RAILS_ENV=production bundle exec rake db:seed

The file that will set the superuser data is
/db/seeds.rb

To access the database from the developer's side, run the following command with the appropriate database name:
sudo –u postgres psql database_name
Currently, the database in production is compassionsite_prod

Database initialization
To create a new database and populate it with the currently existing migrations, run the following commands:
rake db:create
rake db:migrate

To revert the database, run the following command before editing the migration:
rake db:rollback

Should the database have to be dropped (say a migration was edited without rolling back and the schema cannot be updated), run this command:
rake db:drop

Services (job queues, cache servers, search engines, etc.)
In order to start the daemon that updates the status of listed books to sold, the following command must be executed and left continuously running:
RAILS_ENV=production bundle exec lib/daemons/check_listing_state_ctl start
The file check_listing_state.rb may also be edited should the developer desire to add more functionality to the daemon

Deployment instructions
Any changes made on the application can be put into effect by restarting the server via the command:
sudo service apache2 restart
Depending on the server, this command may be a requirement before any changes to the website can be seen.

General setup instructions

Once the superuser has been populated in the database, he has the ability to create all schools, organizations, causes, and users to begin logging, listing, and shipping books

Pages README

(This file is included within the pages folder of the Project)
==README==
The files in this folder are used to communicate with Half.com
The following explains how each file is used and what they do
*listing renders the _form page.
*_form contains the paramaters that will be posted on Half.com
*post_form takes the parameters from _form, communicates with Half.com, and lists a book on Half.com.
*search.html.erb is used solely to check prices on Half.com given an ISBN. It currently is not rendered by any page on the website
*edit_listing grabs the parameters from the database so that they can be changed and passed to manage_listing.
*manage_listing allows the user to change information about a current listing, such as price and condition, and update them in both the database and on Half.com.
*unlist_book removes a listing from Half.com completely.
*ship_book grabs information about the order via screen scraping. It then has buttons to print the packing slip and shipping label.
*packing_slip does a screen scrape to get the html of a packing slip. It then saves the html as a file in /packing_slips/. These files are unique to each order, but each start with _packing. If the packing slip file already exists, a new one will not be created, and the already existing _packing file will be rendered
*_packing is rendered by packing_slip and returns the packing slip on the website, which can be printed.
*shipping_label does a screen scrape of an already purchased shipping label to get the html of that page. It saves the html to shipping_labels/. These files have a unique shipping id, but each start with _shipping. The file is then rendered by shipping_label, which returns the actual label, which can be printed. If the shipping label file already exists, a new one will not be created, and the already existing _shipping file will be rendered
*mark_shipped changes the state of a book to shipped.
*reprint_labels gives the user the option to reprint the shipping label and packing slip. A shipping label may only be reprinted 10 times. This is an unalterable setting set forth by Paypal.

Credentials README

(this file is included in the main folder of the Project. It is called CREDENTIALSREADME)

Half.com Selling Account

In order to use this website, a valid Half.com selling account, which is also associated to a PayPal account for the shipping labels, is needed. The credentials for the Half account must be stored in the following way:

1) Log in as superuser
2) Click on Manage Half Account
3) Click Edit

4) At the bottom, fill in the username and password of the Half account.

eBay Developers Account Setup

In order to use Half's API and developer tools, an account is needed on the ebay Developers Program. On this account we can generate application keys, which are unique identifiers that tell eBay which person and application is making a call. For Half.com specifically, we can only use the **Production** keys.

Currently, the account is under Anastasia's name. This can be accessed in the following:

1) Go to https://developer.ebay.com/DevZone/account/
2) Username: ashpurik Password: Compassion13
3) You can see the production keys currently in use.
4) This is currently using the uwcompassionbtb Half account.

A new developers account can be created by the following link:

1) https://developer.ebay.com/join/

You active your account by email confirmation

2) Go to My Account – https://developer.ebay.com/DevZone/account/Default.aspx
3) Under Production Keys, click "Generate Production Keys"
4) Once you have these keys, you may go back to My Account
5) Click on Configure Settings for the Production Keys. It generates two sets of keys, you may choose either one.
6) Click on User Tokens
7) Click on Generate Token
8) It should then prompt you to sign in with your ebay/Half.com account. You may then have to confirm your identity
9) This page contains all the data you will need. Follow the first three steps as in the Half.com Selling Account. Now copy each appropriate field from the Developers page and paste it into the corresponding fields on the compassionbythebook website. You may leave compatibility as it is.
10) Take note of the Token Expiration Date. The token you entered above will not work after this date. Be sure to generate a new token when this occurs.