**MOM & POP BOOKS**

**Team Name**: Website

**Members**: Randy Halim, Stefano Onorati,

Jimmy Tang, Meng Zhao

EECS 4413 Z

**Professor**: Marin Litoiu

**Date**: April 1, 2016

# **Table of Contents**

[Implementation:](#h.on64325wmwf8) ...3

[Performance Testing Report:](#h.60hygg7xqdro) ...7

[Team member Contributions:](#h.g86emqmzpuyi) ...8

# 

# 

# 

# 

# 

# 

# **Implementation:**

* **Database:**
  + Used Derby as opposed to MySQL due to more exposure and experience with Derby in class
  + Easier to set-up and integrate in the project due to reasons above
* **Authentication:** 
  + Created table “Users” in Derby Database to store user account information
  + Used a .jspx page as a GUI interface for users to login, register and logout
    - Just one jspx used for both login and register forms for convenience and organization as opposed to two different .jspx pages
  + Stored user in session attribute to login
    - Logging out uses invalidate() method from HttpSession
  + Username of account must be unique
    - Does not allow the creation of two different accounts with same username
  + SHA-1 hashing is used to encrypt the password due to its simplicity
    - Although not as secure as other encryptions such as PBKDF2 or BCrpyt (can be cracked by brute force and dictionary attack), we decided to use it for this project’s purposes as we wanted to meet the requirements
    - **Trade-off:** chose implementation simplicity vs. implementation complexity and security
* **Books:** 
  + Created table “Books” in Derby Database to store books
  + Expanded from the provided schema to include other information for books, such as author, page count, publisher, and image filenames
  + Able to retrieve books through search
  + Main page features the top 5 rated books in the database
* **Search Functions:** 
  + Retrieve a collection of books that match a particular property
  + Two ways to search books in store:
    - Text Search
      * User-inputted text and three select options
      * Search bookstore for book title, book author, or all (both title and author)
      * If any book matches the query, then search page is populated with results, ordered by highest rated books
    - Selection Search
      * User selection of two fields
      * Able to search book categories and price ranges
      * Categories displayed exist in database, so as not to get no results
  + **Trade-off:** provided user with two simple search functions rather than one complex search function with result filters
* **Adding Reviews:** 
  + Created table “Reviews” in Derby Database to store reviews for books
  + Used a .jspx page as GUI interface for adding reviews
    - Able to access this page only via a link provided on the book.jspx page
    - Able to separate this function and keep it
  + To be able to add reviews, the user must be logged in or have account with store.
    - Cannot allow guests to add reviews since database is dependent on the username as part of the primary key.
    - Several conditions to satisfy as well:
      * Username and book id must exist in the store database
        + Review is not added if this condition fails
      * No duplicate reviews (user cannot add two reviews to the same book)
        + Review is not added if this condition fails
      * Cannot add reviews with a rating > 1 or < 5 and no comments or comments where length > 250
  + Can also retrieve average rating of a book and the count of each rating in database
    - Cannot retrieve ratings for a non-existing book\
* **Security:** 
  + Implemented SSL in Tomcat server
  + Created self-signed certificate with Java keytool and stored it in keystore
  + Modified server.xml in Tomcat folder to accept SSL connection to port 8443 (may need to change it to default port: 443)
* **Shopping Cart:**
  + Implemented a shopping cart that has both user and guest cart functions
  + Asks user to login so they can save/merge the cart and not lose it when they close the session
  + Contains buttons to clear the cart, update quantities, or remove items from cart
  + When an image is clicked in the shopping cart, it redirects to that book’s template page where users can see and add reviews.
  + Getting the total cost in both user cart and guest cart are slightly different
    - For guests, simply get the variable total from cart and output it. Add/update/remove automatically updates the total.
    - For users, the total has to be recalculated completely each time something is added/removed/updated to stay in sync with the user cart database.
  + When a user is not logged in, user is considered a guest and carts are only saved in current session
    - The cart functions are only acted locally and saved to a cart called guestCart, no derby commands are used.
  + When user is logged in, user cart is saved to database
    - If there is anything in the guest cart, it is automatically merged with the user cart if there is any. If not then the guest cart is the new user cart. Nd is then saved in the database.
    - The user cart is saved in the database and is modified using derby commands whenever user wants to update, remove or clear cart
    - Uses the user’s name and book id is primary keys for the database. User ID will be unique (can’t have the same username for 2 different users)
  + Has javascript functions to prevent some errors like SQL errors, check valid numbers and max quantity.
    - Have safeguards in the code in the event that the user has javascript disabled except the add to cart SQL error, which doesn’t do anything even if the error happens.
  + Implementation decisions
    - Creating a table called cart
      * Initial idea was to create a separate table for each user. After fiddling with the idea it was determined to be a bad idea as it would get complicated to keep track of in the end
    - Having single rows for each user with an associated book id
      * Initial idea was to have multiple of the same entry to determine quantity size. After trying to implement it this way, quickly realized can’t do a remove and update function very well so had to abandon it mid-implementation. It was just really bad implementation and optimization. Relic of it can be found in the code commented out.
* **Payment page**
  + Confirm button for the billing address
    - Change input fields to text field, and display user input
      * Instead of input text fields, it becomes text for user to view.
    - Create new button, ‘new address’
      * Upon click
        + Changes text to input field for user to input new information
        + Creates new buttons, ‘set address’ and ‘cancel’

‘Cancel’ restore the session to before ‘new address’ is clicked

‘Set address’ changes what is displayed and changes serlet content

* + - Enable a checkbox
      * When it is selected, disable input fields and button for shipping address
        + Uses Ajax to set shipping address information in servlet
      * When is unchecked, enable input fields and button for shipping address
      * Get user input for billing address and fill it to shipping address
    - Uses Ajax to call servlet and store user input to session scope
      * Each time set address is used, goes to servlet and update the related information
  + Confirm button for the shipping address
    - Change input fields to text field, and display user input
      * Instead of input text fields, it becomes text for user to view.
    - Create new button, ‘new address’
      * Upon click
        + Changes text to input field for user to input new information
        + Creates new buttons, ‘set address’ and ‘cancel’

‘Cancel’ restore the session to before ‘new address’ is clicked

‘Set address’ changes what is displayed and changes serlet content

* + - * Uses Ajax to call servlet and store what user input
    - Uses Ajax to call servlet and store user input to session scope
      * Each time set address is used, it updates the information to servlet
  + Set card button
    - Change input fields to text field, and display user input
      * Instead of input text fields, it becomes text for user to view.
    - Create new button, ‘new card’
      * Upon click
        + Changes text to input field for user to input new information
        + Creates new buttons, ‘set card’ and ‘cancel’

‘Cancel’ restore the session to before ‘new card’ is clicked

‘Set card’ changes what is displayed and changes serlet content

Create new button ‘clear’

Delete what is store for card and require user to input

* + - Get current year and set expiry date for credit card
      * Instead of letting user input expire year themselves, they can choose from a list
        + Creating the display year by getting current year and add ten year to current year and display to the list
    - Get credit card holder’s full name
      * Separate name apart and decides what will be the last and first name
        + Store the name in the database amount other information
  + Confirm order
    - Redirect current page to main page and process to handle other user’s request

# **Performance Testing Report:**

-------------TEST ONE: FEATURED BOOKS------------

BOOK p002;Leviathan;Thomas Hobbes;2008;7;Paperback;0199537283;576;Political Science;Oxford;leviathan.jpg;5.0

BOOK f006;1984;George Orwell;2008;8;Paperback;0141036141;336;Fiction;Penguin UK;1984.jpg;5.0

BOOK f003;A Storm of Swords: A Song of Ice and Fire;George R.R. Martin;2003;10;Paperback;055357342X;1216;Fiction;Bantam;stormofswords.jpg;5.0

BOOK p001;The Communist Manifesto;Karl Marx & Friedrich Engels;2008;7;Paperback;019953571X;96;Political Science;Oxford;communistmanifesto.jpg;2.0

BOOK cs001;Java By Abstraction: A Client-View Approach;Hamzeh Roumani;2010;184;Paperback;0558819834;506;Computer Science;Pearson Learning Solutions;javabyabstraction.jpg;1.0

-------------TEST ONE: PASSED------------

-------------TEST TWO: CATEGORIES------------

CATEGORY 1 Computer Science

CATEGORY 2 Engineering

CATEGORY 3 Fiction

CATEGORY 4 History

CATEGORY 5 Political Science

CATEGORY 6 Science

-------------TEST TWO: PASSED------------

-------------TEST: SEARCH BOOKS BY CATEGORY: Children's------------

-------------TEST: FAILED------------

-------------TEST: SEARCH BOOKS WITH CATEGORY Self-Help AND MIN PRICE 5------------

-------------TEST: FAILED------------

-------------TEST: SEARCH BOOKS ABOVE MIN PRICE 185------------

-------------TEST: FAILED------------

-------------TEST: ADD REVIEW FOR NON-EXISTING USERNAME------------

-------------TEST: FAILED------------

-------------TEST: ADD REVIEW WITHOUT A NUMBERED RATING------------

-------------TEST: FAILED------------

-------------TEST: GET AVERAGE RATING OF NON-EXISTING BOOK------------

-------------TEST: FAILED------------

-------------TEST: GET REVIEWS OF NON-EXISTING BOOK------------

-------------TEST: PASSED------------

# **Team Member Contributions:**

Our team worked often in scrums in EECS labs in Lassonde. Meetings occur either daily or once every two days when team members do not have other obligations. Each meetings take anywhere between an hour to 3-4 hours as we do a combination of discussion on how the project is going and helping each other in implementation problems. We each chose different parts of the project to work on and constantly update on the progress in our meetings. We also post .war files of the project as we near the project deadline as we try integrating each of our parts together in one project file. We collaborated extremely well as decisions regarding the project are made unanimously and quickly. In addition, team members all get along with the unified goal of completing the project with our best combined efforts.

Stefano Onorati worked on the main page features. He provided the search function, as users are able to search the store database via text input for book titles and authors and selection options for book categories and price ranges. He was also responsible for extending the database table for books by including other book features such as page count, image filename, and author’s name. These features are reflected in the book page that was also set up by him. He also provided the database for reviews and implemented the review feature for the book page. This was in relation to Randy’s part of implementing user login and registration to our bookstore.

Randy Halim worked on the Authentication part of the bookshop. The login, register and logout function, allowing users to authenticate and enter the bookshop as different types of users (administrator, partner, customer, visitor). Also provided the SSL configuration in order for the bookshop to operate under HTTPS. Implemented Web Services jointly with Stefano to provide Product Catalog and Order Process services, allowing partner accounts to access the bookshop and order catalog. Finally, he created UML use cases, class diagrams and sequence diagrams.

Jimmy Tang worked on the shopping cart portion of the bookstore. He takes the books from Stefano’s main page and search pages and adds it into a shopping cart when a user clicks the shopping cart icon. Also included the checkout, update, remove, and clear cart functions. Javascript was also developed with the shopping cart to try to keep user error to a minimum and to prevent errors if a user tries to input something invalid. He also created the merging of carts where if the user is not logged in, add books into cart, and then logs in, the carts would merge and the user cart that’s saved in the database is updated accordingly.

Meng Zhao worked on the payment page of the book store. When the user tries to check out, his page will be shown to user with many input fields. The user needs to fill out three tables, first one is billing address, second one is shipping address, and the last one is credit card information. Then upon the user clicking confirm for each field, it does checking for correct input type, such as numbers are used in phone, credit card number. In addition to checking fields, he uses JavaScript to changes the page’s layout and enable new inputs for user, such as. In the end when user clicks confirm order button, it connects to the database and store billing, shipping address, purchase order, and purchases order items into database.

Signed,

Randy Halim Stefano Onorati Jimmy Tang Meng Zhao