|  |
| --- |
| **FPT-aptech computer education** |
| eProject Document |
| [BookStore] |
|  |
| |  |  | | --- | --- | | **<Group 1>** | | | **Group Member** | <Vu Van Tiep><c00614><2164>  <Trinh Duc Trung><c00612><1610>  <Nguyen Cong Minh><c00608><1747> | | **Instructor** | <Nguyen Tuan> | | **Batch** | <C1108I> | | **Semester** | <3> | |
|  |

|  |
| --- |
| - Hanoi, <6>/<2013> - |

Table of Contents

[1. Introduction 4](#_Toc358736201)

[2. Problem Definition 4](#_Toc358736202)

[2.1. Problem Abstraction 4](#_Toc358736203)

[2.2. The Current System 4](#_Toc358736204)

[2.3. The Proposed System 5](#_Toc358736205)

[2.4. Boundaries of the System 8](#_Toc358736206)

[2.5. Development Environment 9](#_Toc358736207)

[3. Requirements and Business Flow 10](#_Toc358736208)

[3.1. Customer Requirement Specification 10](#_Toc358736209)

[3.2. Activity Diagram 11](#_Toc358736210)

[3.3. Use Case Diagram 11](#_Toc358736211)

[3.4. Use Case Specification 11](#_Toc358736212)

[3.5. Other Concerns<Optional> 12](#_Toc358736213)

[4. Design 13](#_Toc358736214)

[4.1. System Architecture 13](#_Toc358736215)

[4.2. Class Diagram 13](#_Toc358736216)

[4.3. Class Diagram Explanation 13](#_Toc358736217)

[4.4. Sequence Diagram (Optional) 13](#_Toc358736218)

[4.5. Collaboration Diagram (Optional) 13](#_Toc358736219)

[4.6. State Diagram (Optional) 13](#_Toc358736220)

[4.7. Entity Relationship Diagram 13](#_Toc358736221)

[4.8. Database Design 13](#_Toc358736222)

[4.9. Algorithms (optional) 14](#_Toc358736223)

[4.10. Others (optional) 14](#_Toc358736224)

[5. System Prototype 15](#_Toc358736225)

[6. Management and Project Planning 16](#_Toc358736226)

[6.1. Management Approach 16](#_Toc358736227)

[6.2. Project Plan 16](#_Toc358736228)

[6.3. Task Sheet 16](#_Toc358736229)

[6.4. Meeting Minutes (Optional) 16](#_Toc358736230)

[7. Checklists 17](#_Toc358736231)

[7.1. Check List of Validation 17](#_Toc358736232)

[7.2. Submission Checklist 17](#_Toc358736233)

[8. Screenshots 17](#_Toc358736234)

[9. Coding Convention 17](#_Toc358736235)

[10. Other Concerns<Optional> 17](#_Toc358736236)

[Appendix 18](#_Toc358736237)

[Glossary [Optional] 18](#_Toc358736238)

[References [Optional] 18](#_Toc358736239)

[Others<Optional> 18](#_Toc358736240)

# Introduction

* Book Store is general book stores .His book stores span across entire India with his daily bussiness running into thousands of dollar .Shradha Book Store boasts to be one of the finest collectors of books in India. Shradha Book Store contain books on a lot of different genres.They provide book travel,children,fiction,cookery,technical subjects,philosophy,sports,autobiographies,spirituality and so on.
* With the recent advent of Internet revolution,Everyone are developing e-commerce Web sites for different resaons: to reach new customers,to stay abreast with their competitors,and to meet their customers’ expectations and needs.This has resulted in more and more people buying commodities through the Internet.
* Hence,to cope up the dynamism in the market, he(Shradha) has now decided to launch a Book Store Online .This Web site would enable the customer to search his/her favorite book and shop for it.

*>*

# Problem Definition

## Problem Abstraction

Our client is a retail dealer for the books and sells the books at his shop

Shradha General Book Stores . He sells different types of books of different

publications of stories, school related text books, note books, essay writing books,

some of books for the competitions like General Knowledge, Essay Writing, and also

novels, books to improve vocabulary, for learning other languages, etc and also he

sells some of the book related materials like covers, pens, pencils, etc .. Also the

slam books, Files, etc . are available at his shop.

## The Current System

Actually the client deals directly with some of the publications for purchasing the books, and for some of the publications he purchases the books through the other

dealers. Also some of the dealers and the book sellers purchase the books from our

client. The customers either visit the dealer or call him for purchasing. Similarly is

the case with the client.

Because of no automation in the system, the orders placed get delayed, due to

which the customer dissatisfaction happens. Also in this competitive world where the

number of publications and the dealers are increasing, it becomes difficult to retain

the customers.

In order to retain the customers, he has placed some of the discount schemes

such as if the customer places an order in bulk then that customer can get the

discount based on the quantity of orders. Also he has provided a home delivery

which will be free, provided the distance is < 3 KMs and for the distance > 3 KMs,

the charges are as per the delivery charges as per the service chosen.

## The Proposed System

To bring the Automation in the system and to cope up in this competitive world, our client now want to incorporate an online sale system where the customers can directly interact and place their orders through online itself. So he has approached us for developing the website. He wants the following list to be maintained and

implemented for the website.

**Non-Financial :**

* He wants the website to be navigated through different pages like contact information, About Us (information about the dealer),Products (Books,etc),FAQs, Order Tracking, Buying a product, Feedback and Queries and the ADMIN.
* For the Products there should be information categorized based on the categories like Books, Magazines, Files, CD s, DVD s, Utilities like Pen, Pencil, Paper,Sketches, etc. and the sub categories based on the Publisher, type of the Products (like for books novel, Magazine, etc ., for CD s readable or rewritable, etc and similarly is the case with the other products)
* Also for the products like books, CD s, DVD etc . the details like the manufacturer, price, and the type of the product (like for CD s it will be readable or re writable) should be indicated
* For the Books like Novels, Preparation books like learning vocabulary, General Knowledge, etc the brief description is to be included (like What is the book all about and for what it is used, Author Name, Publisher Name, Release Date,Version, etc .)
* The products to be assigned with a unique seven digit code, which will be as follows :

1. The Category of the Products (like Book, Magazine, Files, CD s, DVD s, etc utilities etc ) are to be assigned with the Unique 2 digit alpha numeric code (first being Alphabet and the second one being the Numeric digit)
2. The subcategory of the Products are to be assigned with a unique 5 Digit Code (First three Digits being the first three acronyms of the Manufacturer and the remaining two being the Numeric digit) so as to separate the different products being manufactured by a single manufacturer (like if a manufacturer Ab Cd manufactures products X and Y , then the Code for X can be AC001 and for Y it can be AC002 , where AC0 is representing the manufacturer, AC being the first two acronyms of the manufacturer and the digit zero being added as the third one, as the manufacturer is having only two acronyms, and the remaining two will be the numeric order, which can be in the order you maintain the list of the products of that manufacturer).

* The details of the Products should be easily inserted, updated and deleted.
* The admin alone will be able to insert, update or delete a product. So there should be a login for the Admin.
* When one places the order, the order number is to be auto-generated and the code will be a unique 8 digit code, which will depend on the serial order of the receipt of the booking (order placed).
* Also one can track the order through the Order Tracking link in the Navigation bar. Also in this page alone one can cancel or update an order based on the conditions applicable.
* And One can provide a feedback and also can post their queries through Online through the Link Feedback and Queries in the navigation bar.
* In the Feedback and Queries and the Order Form (available when a product is chosen for placing an order i.e., Buy a Product page) page the Email id will be taken for further conversation.
* When an Order is to be Placed, i.e., in the order form, the following Message is to be displayed Apart from the Product Charges, the Delivery charges will be Nil, if <3 KMs and if >3KMs is as per the Delivery charges based on the type of the delivery chosen.
* As per the orders received, the customer details is to be maintained so as to track the details of the orders placed by that customer and as to know the status of the order.
* The payment is of three types:

1. Payment before delivery (Note : where the customer need to send a Demand Draft of the total bill so as to receive the order)
2. Payment after Delivery (Note : Payment to be done at the time of receipt of the product). It can be done in two ways, either payment through cash or payment through cheque.
3. Online Payment (Note : This payment is through the Credit Card, and the order will be dispatched only after the payment is being processed by the respective bank)

* The FAQ s Should consist the Following :

1. Will there be any charges for the delivery apart from the products ordered?
2. How many days will it take to deliver the order? (Ans : Depends on the Order,Quantity and Stock and as well on the payment type chosen, as if the payment option chosen is payment before delivery, then the order will only be dispatched after the receipt of the payment.)
3. What if the order delivered is not in proper condition? (Ans : If the order delivered is improper, it will be replace with the other)
4. Can an Order be cancelled? If yes, will there be any charges? (Ans : Yes, but if cancelled with in 24 Hrs. there will be no charges, but if cancelled after 24 Hrs. and if the Order is being dispatched the Delivery charges for the to and fro is to be bared by the one who has ordered).
5. How can the payment be made?

**Financial :**

The charges for the delivery will be as per the distance and the quantity ordered

and will be only the delivery charges and the total price of the products ordered. In

case of any cancellation, the charges will be the delivery charges (to and fro) alone.

Also the delivery charges are applicable only if the distance is >3 KMs.

## Boundaries of the System

Depending on the decision taken by the team,following are the requirements based on which the Web site needs to be developed:

**Functional Requirements:**

1. The Admin should alone be able to update or insert or delete the details of the

products and the order and as well that of the FAQ s.

2. The admin should alone be able to reply to the feedback and Queries that are

posted by the customer.

3. Auto generation of the unique 8 digit Order Number and based on the series

of the order received.

4. The products to be assigned a unique seven digit code based on the

manufacturer and the category of the product and as well based on the serial

order of entry of that particular product.

5. The feedback and query page to collect the feedback and queries of the

customer.

6. The Order tracking page, where one can track the status of the order and as

well cancel or update an order

7. Creation, maintenance and updating database, which contains

* Details of the Products.
* Details of the Manufacturers.
* Details of the stock available.
* Details Orders (included order details and the Delivery details).
* Details of the Payments based on the Order and Products and the
* Distance of delivery.

8. The billing for the order is to be calculated taking into account the Order

quantity, type of the product, and the Distance of the Delivery.

## Development Environment

*<* Following current scenario,we will develope a Web site named ”[www.shradhabookstores.com](http://www.shradhabookstores.com)” using ASP.NET and MS SQL Server 2008 & MS SQL Server 2012.The Web site authenticates the existing user by his user name and password.The Web site has a registration page where a new user can enter his personal details in the given fields.The user details get stored in the database and the database containig details of existing users gets updated.

After the user authenticated,the list of available books along is displayed.The Web site contains a form that allows the user to purchase the selected book by accepting the details such as Shipping Address,etc.For any queries and suggestions,the user can interact with the Web site coordinator through the given email address.

**Hardware Requirements:**

|  |  |  |
| --- | --- | --- |
| No. | Items | Description |
| 1 | CPU | Pentium IV CPU 2.80 GHz or better |
| 2 | RAM | 512 MB of RAM or better |
| 3 | HDD | Minimum 4 GB |

**Software Requirements:**

|  |  |  |
| --- | --- | --- |
| No. | Items | Description |
| 1 | Operating System | Windows 7 or 8 |
| 2 | Database | SQL Server 2008 |
| 3 | Browser | Firefox or Chrome |
| 4 | Other software | Windows Installer 3.0 or later, Internet Information Services (IIS) 7.0 or later, Visual Studio 2010, Microsoft .NET Framework version 3.5, Ajax Extender For Visual Studio 2010 |

**Scope of the Work (in brief):**

* A GUI containing text boxes, buttons, drop-down list boxes, labels and hyperlinks, which provide a user-friendly interface to the user to work with the web site.
* Allow customer to add, update and delete items in the shopping cart.
* Allow customer make payment through 3 types: payment before delivery, payment after delivery, online payment.
* Generation of invoice (bill) after successful purchase of a book.

# Requirements and Business Flow

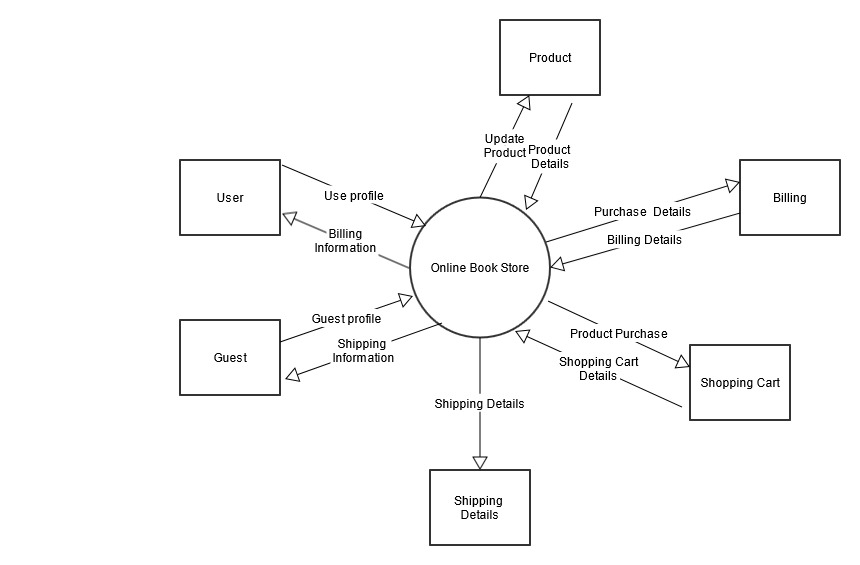
## Customer Requirement Specification

* This Web site aims at providing the customer with searching a book for their favorite category and subsequently shopping for it.The Web site would also enable the customer to place a request for a book.Additionally,the Web site should provide the following factors to the customers for better usability:
* Flexible and easy scanning and selecting books from the Web site
* Consistent layout of the Web site
* Returning to different parts of the site after successfully adding an item to the shopping basket.
* Minimal and effective security notifications
* Simple navigation features from the home page to other web pages
* Provide three payment method delivery

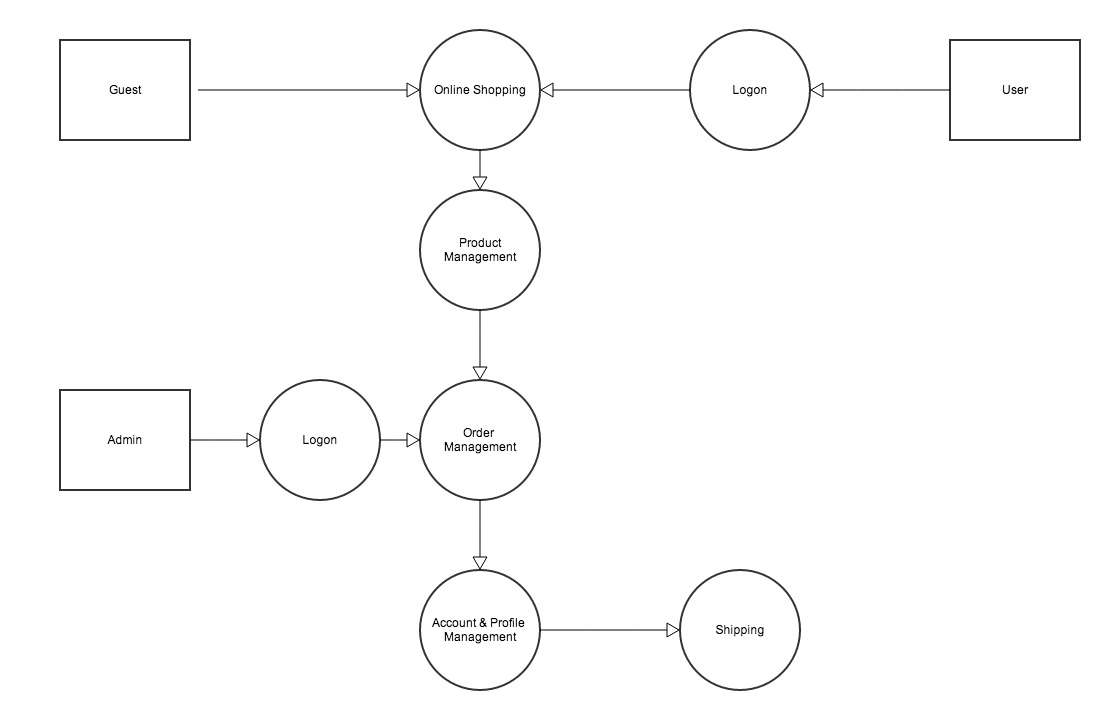
***Inputs provided by the Client:***

* Inputs fot the Web-based application
* Outputs form the Web-based application
* Processed involved in the Web-based application
* Expected purchasing order

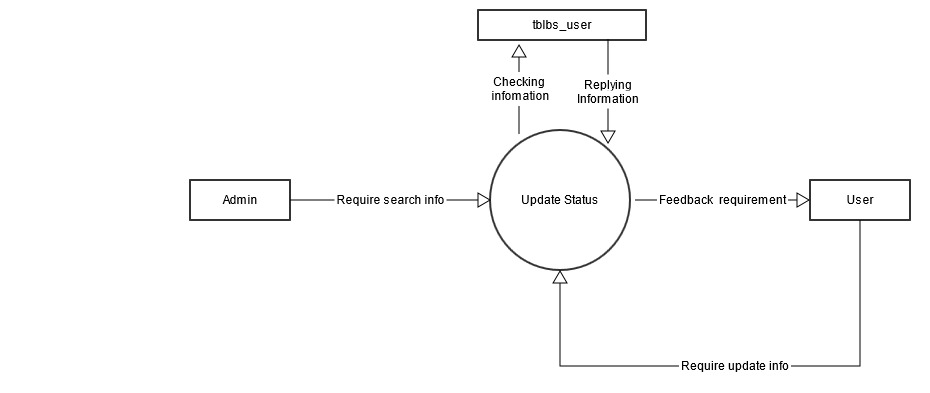
## Activity Diagram



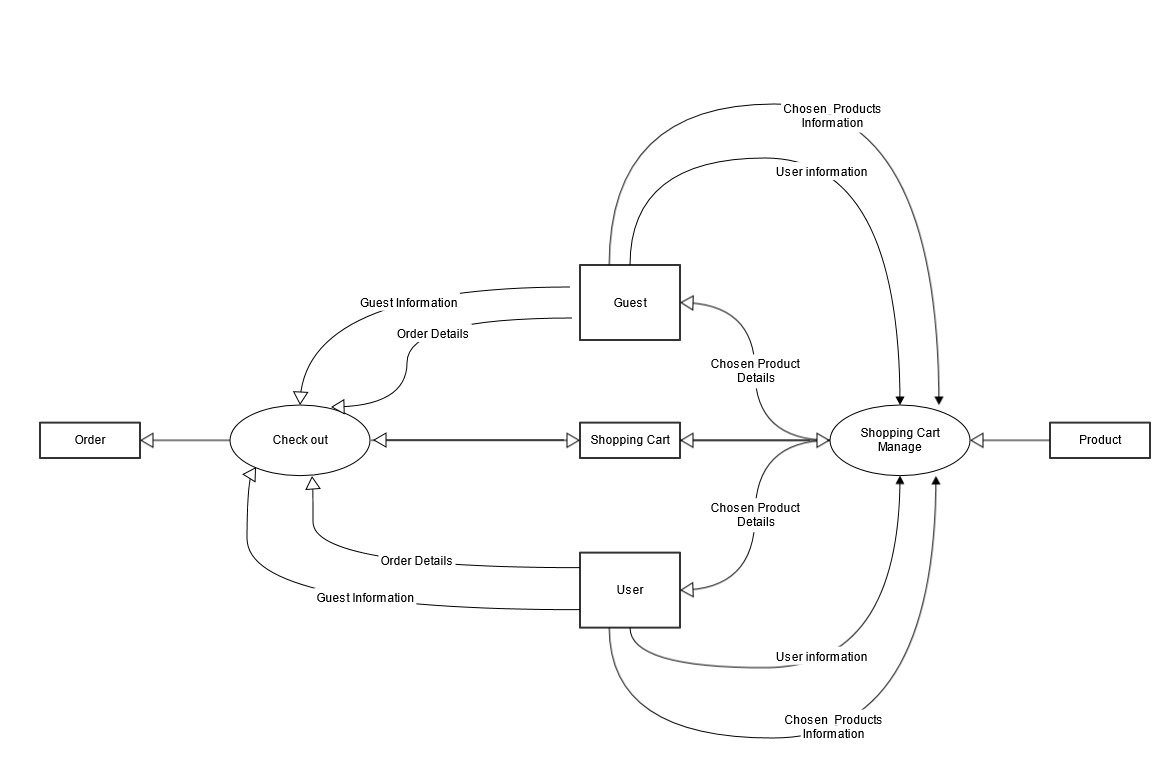
3.2.1 Context Diagram – Online Book Store



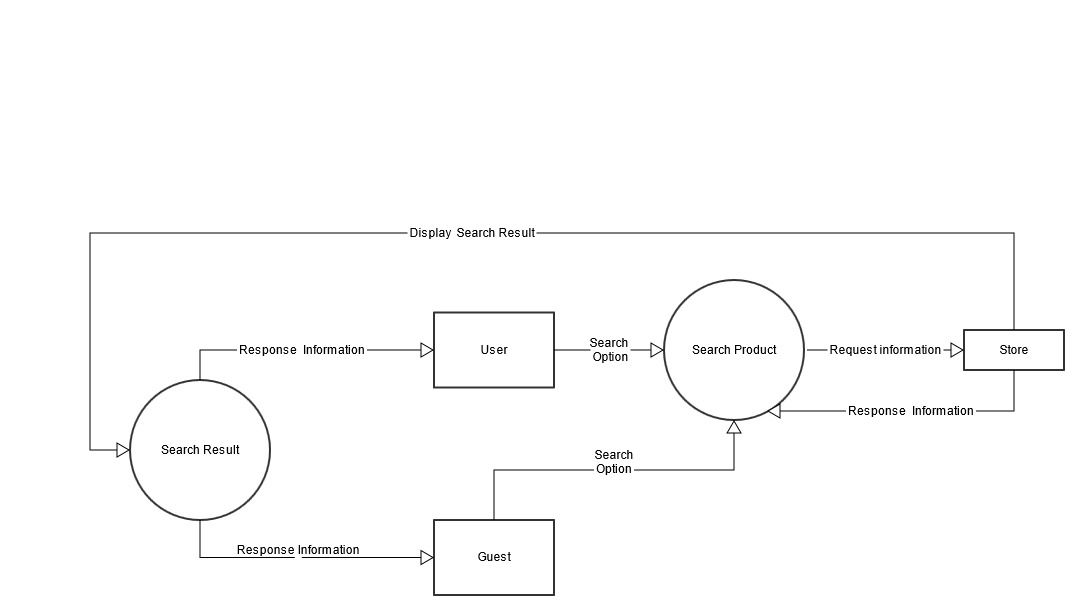
* + 1. DFD Level 1 Diagram – Online Book Store



* + 1. DFD Level 1 Diagram : User Account

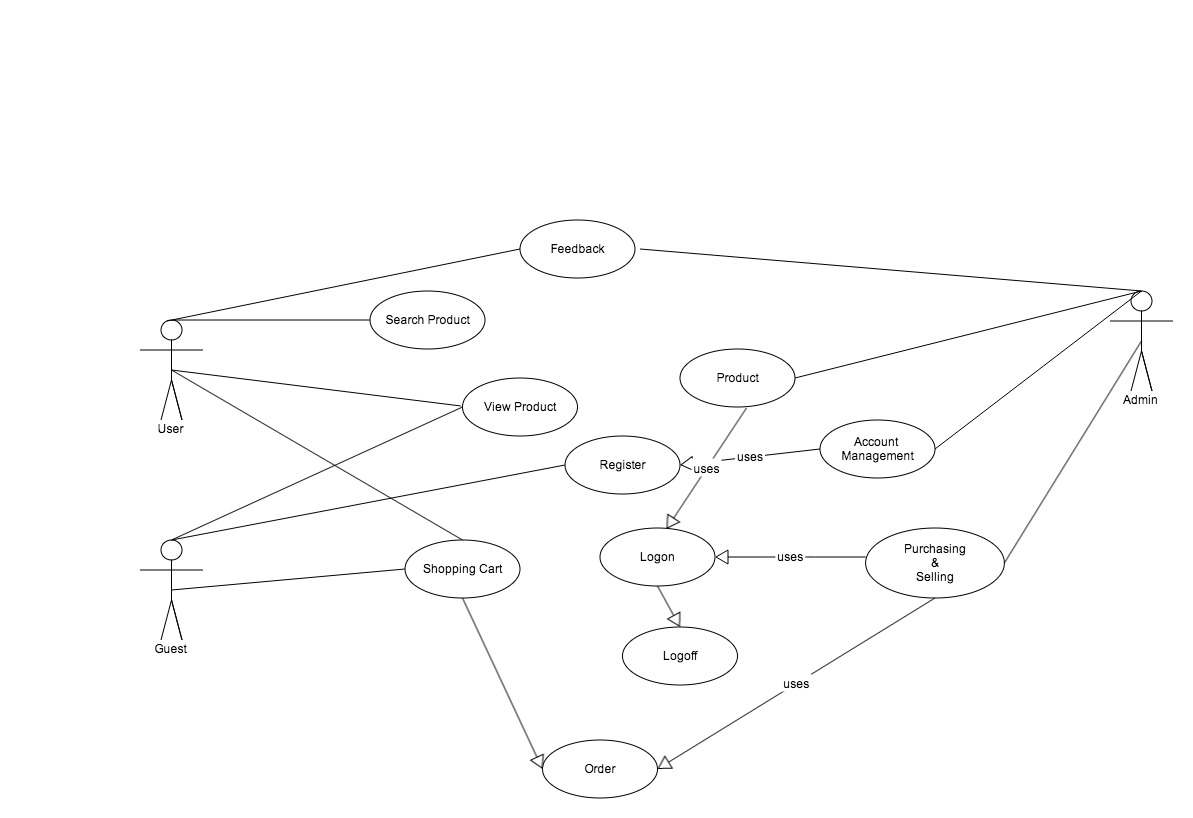


* + 1. DFD Level 1 Diagram : Check out

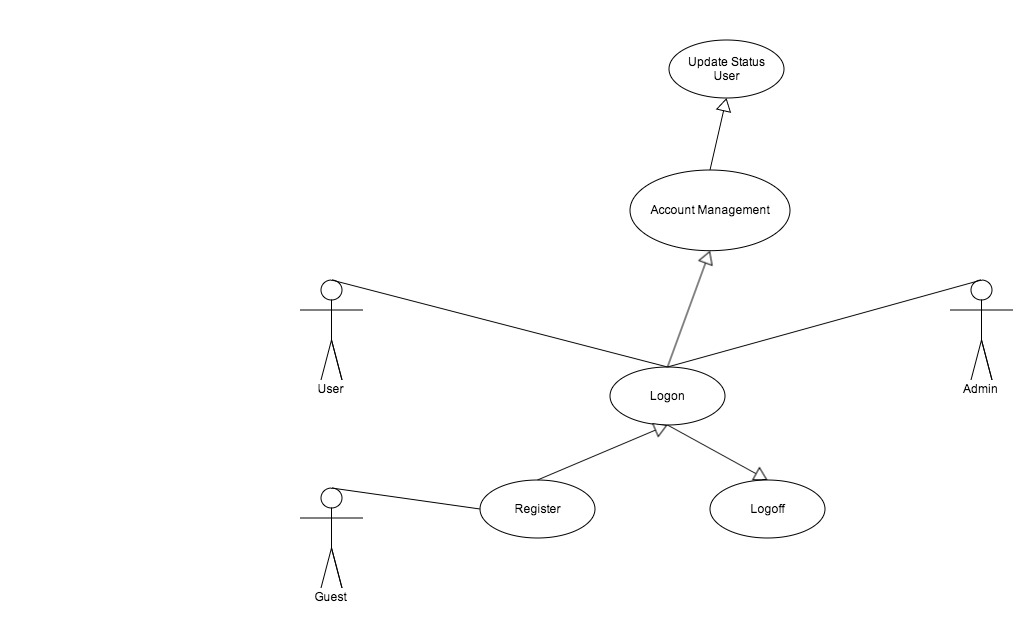


3.2.6 DFD Search

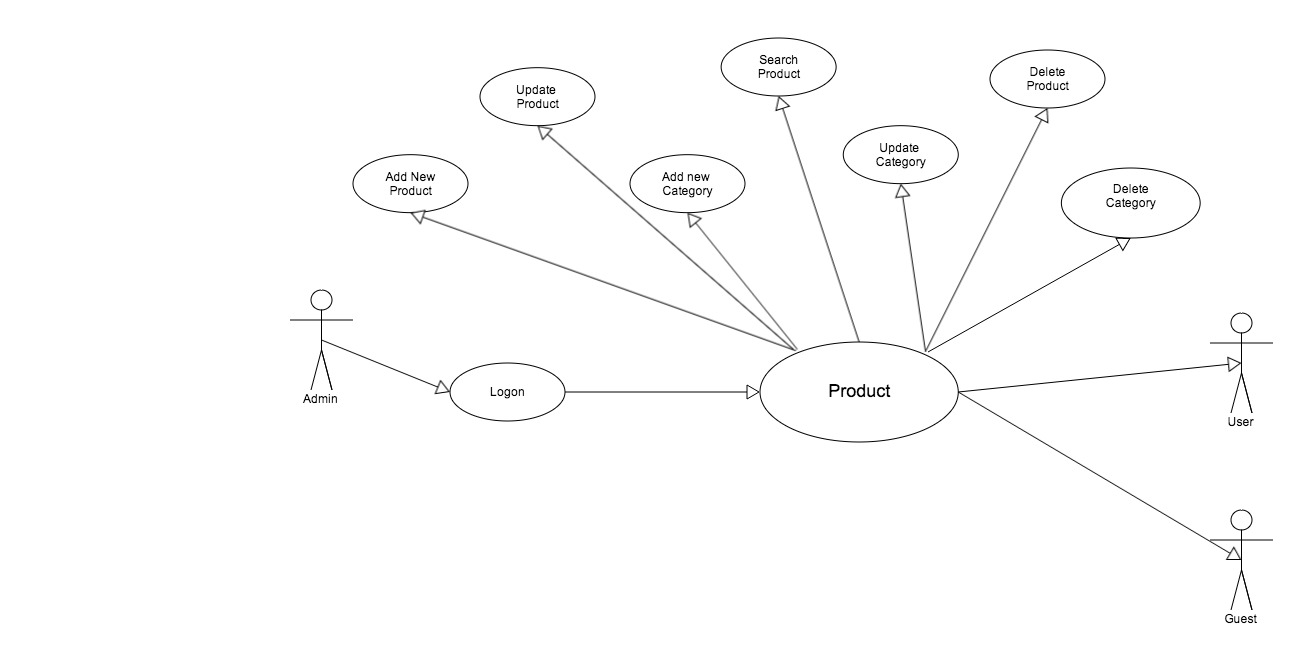
## Use Case Diagram



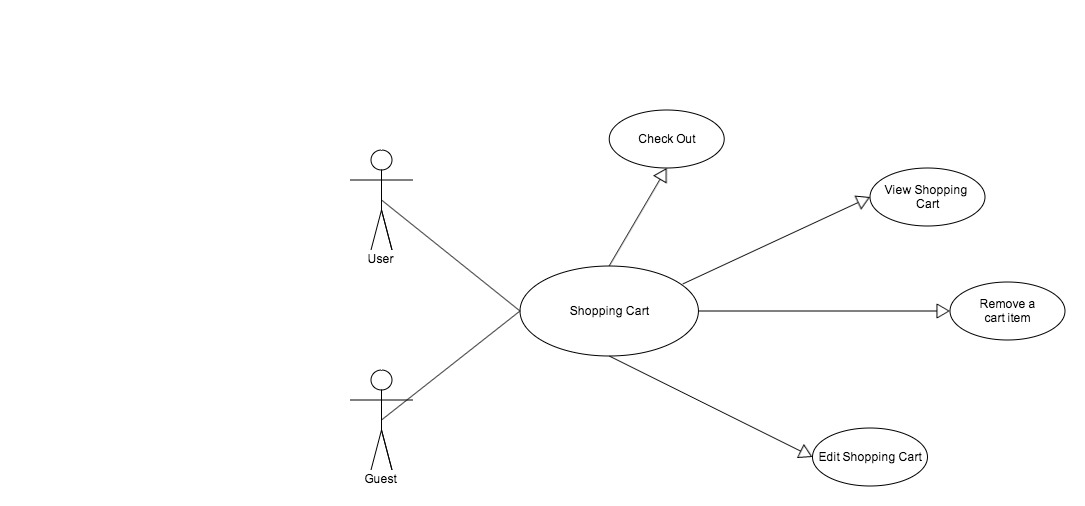
3.3.1 Online Book Store Use Case



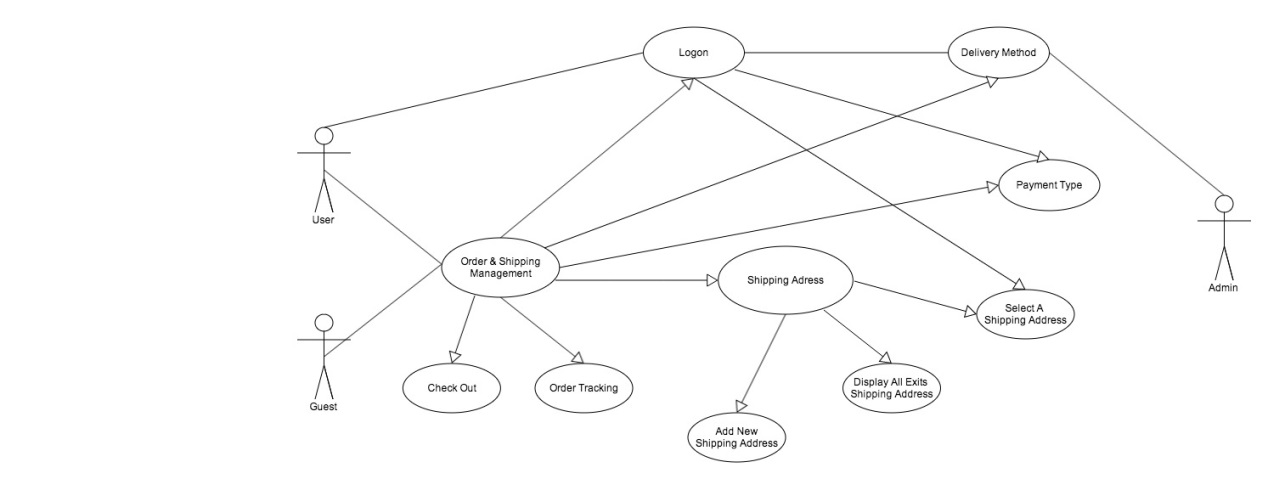
3.3.2 Account Use Case



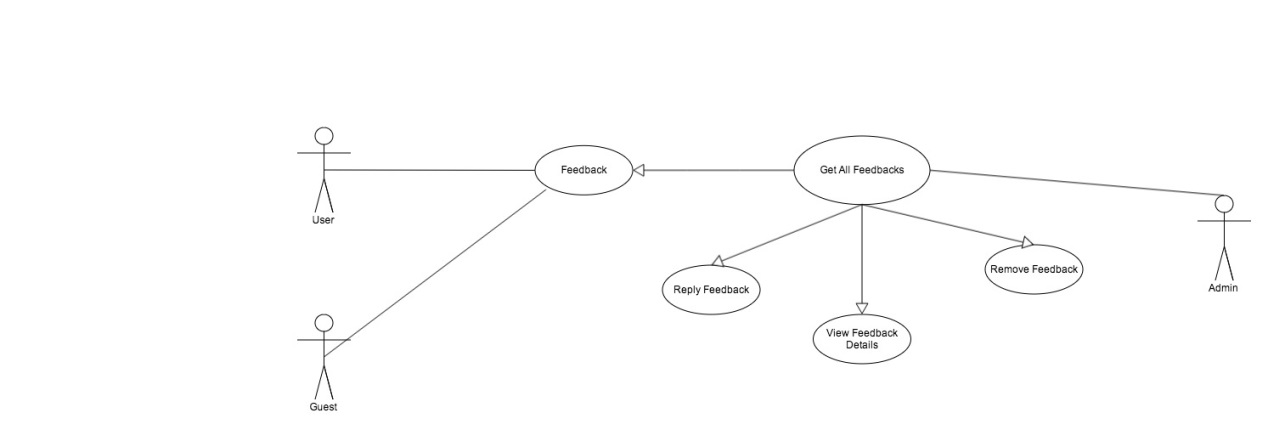
3.3.3 Product Use Case



3.3.4 Shopping Cart Use Case



3.3.5 Buying & Selling Use Case



3.3.6 Feedback Use Case

## Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC01 | **Use-case Version** | 1.0 |
| **Use-case Name** | Shopping Cart | | |
| **Author** | Vu Van Tiep | | |
| **Date** | 24/05/2013 | **Priority** | Important |
| **Actor:**  User  Guest  **Summary:**  User/Guest view and edit shopping cart information.  **Goal:**  Allow user/guest view and edit shopping cart information  **Triggers**    **Preconditions:**   * User/Guest has been authenticated. * User/Guest has bought some products.   **Post Conditions:**  - User ‘s shopping cart is saved.  **Main Success Scenario:**  1. User/Guest login to system.  2. User/Guest add some products to their shopping cart  3. User/Guest view shopping cart.  4. User/Guest edit some information in their shopping cart.  5. User/Guest accept changes that they have made.  **Alternative Scenario:**  Return to main success scenario.  **Exceptions:**   1. User/Guest not login  * Redirect to login page * Authenticate user and redirect to shopping cart page.  1. User/Guest has no product in their shopping cart  * Show message inform to the user.   **Relationships: (List the relationships that use case relates to)**  Login  **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC02 | **Use-case Version** | 1.0 |
| **Use-case Name** | Online Payment | | |
| **Author** | Nguyen Cong Minh | | |
| **Date** | 25/05/2013 | **Priority** | Important |
| **Actor:**  User  Guest  **Summary:**  User/Guest selects one of exists credit card or enter a new card.  **Goal:**  Allow user/guest to select one of exists credit card or enter a new card.  **Triggers**    **Preconditions:**   * User/Guest has been authenticated. * User/Guest has bought some products.   **Post Conditions:**  - Selected credit card information is saved.  **Main Success Scenario:**  1. User/Guest logins to system.  2. User/Guest adds some products to their shopping cart  3. User/Guest views shopping cart.  4. User/Guest edits some information in their shopping cart.  5. User accepts changes that they have made.  6. User/Guest check out.  7. User/Guest chooses one of existing shipping address from their address book or enter a new shipping address.  8. User/Guest chooses one of given delivery method.  9. User/Guest chooses online payment as payment type.  10. User/Guest chooses one of exists credit card or enter a new card.  **Alternative Scenario:**  Return to main success scenario.  **Exceptions:**   1. User/Guest not login  * Redirect to login page * Authenticate user and redirect to payment type page.  1. User has no product in their shopping cart  * Show message inform to the user.   **Relationships: (List the relationships that use case relates to)**   * Login. * Shopping Cart. * Shipping Address. * Delivery Method. * Payment Type.   **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC03 | **Use-case Version** | 1.0 |
| **Use-case Name** | Payment Type | | |
| **Author** | Trinh Duc Trung | | |
| **Date** | 24/05/2013 | **Priority** | Important |
| **Actor:**  User  Guest  **Summary:**  User/Guest selects one of three payment type (before delivery, after delivery, online payment)  **Goal:**  Allow user to select one of three payment type.  **Triggers**    **Preconditions:**   * User has been authenticated.   **Post Conditions:**  - The choosen payment type is selected to deliver user’s products.  **Main Success Scenario:**  1. User/Guest login to system.  2. User/Guest add some products to their shopping cart  3. User/Guest view shopping cart.  4. User/Guest edit some information in their shopping cart.  5. User/Guest accept changes that they have made.  6. User/Guest check out.  7. User/Guest choose one of existing shipping address from their address book or enter a new shipping address.  8. User/Guest choose one of given delivery method.  9. User/Guest choose one of three payment type.  **Alternative Scenario:**  Return to main success scenario.  **Exceptions:**   1. User,guest not login  * Redirect to login page * Authenticate user and redirect to payment type page.  1. User,guest has no product in their shopping cart  * Show message inform to the user.   **Relationships: (List the relationships that use case relates to)**   * Login. * Shopping Cart. * Shipping Address. * Delivery Method.   **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC04 | **Use-case Version** | 1.0 |
| **Use-case Name** | View Home Page & Master Page | | |
| **Author** | Vu Van Tiep | | |
| **Date** | 16/05/2013 | **Priority** | Important |
| **Actor:**  User  Guest  Administrator  **Summary:**  User,guest (included : guest,customer,administrator) view website’s main information about Shradha Online Bookstore  **Goal:**  Allows users,guest can view primary information that is displayed on home page  **Triggers:**  Display main information such as: top best seller products,top newest products,feature products..,category  **Preconditions:**  User has logged in system  **Post Conditions:**  Validation to login system successfully.  **Main Success Scenario:**  1. Input email id & password.  2. Validation.  3. Display warning message : “Login successful”.  4. System authorize user to customize home page    **Alternative Scenario:**  If email id & password are incorrect , system display message “Email or password is incorrect” and focus on email id text box.  **Exceptions:**  **Relationships: (List the relationships that use case relates to)**  Login  View Product Details  **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC05 | **Use-case Version** | 1.0 |
| **Use-case Name** | Statistic | | |
| **Author** | All Member | | |
| **Date** | 03/06/2013 | **Priority** | High |
| **Actor:**  Product Owner  **Summary:**  Statistic Order in period , The best seller product in month , Total Money in period  **Goal:**  Allow PA can manager order  **Triggers**    **Preconditions:**  **Post Conditions:**   * The statistic saved and delivered to sys admin   **Main Success Scenario:**  1. Users links to ContactUs site.  2. Users enters some information about them  3. User inputs statistic message.  4. User sends query to system  **Alternative Scenario:**  Return to main success scenario.  **Exceptions:**  User not input any message or information   * Show warning message * Redirect to ContactUs page   **Relationships: (List the relationships that use case relates to)**  **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC06 | **Use-case Version** | 1.0 |
| **Use-case Name** | View Product Details | | |
| **Author** | Trinh Duc Trung | | |
| **Date** | 06/06/2013 | **Priority** | High |
| **Actor:**  Guest  User  Administrator  **Summary:**  User (included : guest,customer,administrator) view details of products on Shradha Online Bookstore  **Goal:**  Allows users/guest can view information of product that he/she wants to buy  **Triggers:**  Display product information such as: Product Name , Price , Manufacture , Stock , Description and more.  **Preconditions:**  User/Guest has logged in system  **Post Conditions:**  Validation to login system successfully.  **Main Success Scenario:**   1. User click on product image or product name to select a product. 2. Request ProductDetails page 3. System respond product details     **Alternative Scenario:**  If product information has not been updated , system display “This product is updating . Not available now “  **Exceptions:**  **Relationships: (List the relationships that use case relates to)**  Login  Advanced Search  View Home Page  View Shopping Cart  **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC07 | **Use-case Version** | 1.0 |
| **Use-case Name** | Advanced Search | | |
| **Author** | Nguyen Cong Minh | | |
| **Date** | 10/06/2013 | **Priority** | High |
| **Actor:**  Guest  User  System Administrator  **Summary:**  Improve search funtions  **Goal:**  Increase search result  **Triggers**  **Preconditions:**  **Post Conditions:**  **Main Success Scenario:**   1. Enter product name. 2. Select search condition in combo-boxes to request a search result. 3. Click search button. 4. System display result search.   **Alternative Scenario:**  Back to Main Success Scenario  **Exceptions:**  **Relationships: (List the relationships that use case relates to)**    **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC08 | **Use-case Version** | 1.0 |
| **Use-case Name** | Delivery Method | | |
| **Author** | Nguyen Cong Minh | | |
| **Date** | 28/05/2013 | **Priority** | High |
| **Actor:**  User  **Summary:**  User selects one of given delivery method.  **Goal:**  Allow user to selects one of given delivery method.  **Triggers**    **Preconditions:**   * User has been authenticated. * User has bought some products. * User has choose the shipping address.   **Post Conditions:**  - The delivery method is selected to deliver customer’s products.  **Main Success Scenario:**  1. User login to system.  2. User add some products to their shopping cart  3. User view shopping cart.  4. User edit some information in their shopping cart.  5. User accept changes that they have made.  6. User check out.  7. User choose one of existing shipping address from their address book or enter a new shipping address.  **Alternative Scenario:**  Return to main success scenario.  **Exceptions:**   1. User not login  * Redirect to login page * Authenticate user and redirect to delivery method page.  1. Customer has no product in their shopping cart  * Show message inform to the user.   **Relationships: (List the relationships that use case relates to)**   1. Login. 2. Shopping Cart. 3. Shipping Address.   **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC09 | **Use-case Version** | 1.0 |
| **Use-case Name** | Shipping Address. | | |
| **Author** | Vu Van Tiep | | |
| **Date** | 20/05/2013 | **Priority** | High |
| **Actor:**  User  **Summary:**  User view and select one of existing shipping address from their address book or enter new shipping address.  **Goal:**  Allow user choose shipping address.  **Triggers**    **Preconditions:**   * User has been authenticated. * User has bought some products.   **Post Conditions:**   * The new shipping address is saved. * The choosen shipping address is selected to deliver customer’s products.   **Main Success Scenario:**  User choose one of existing shipping address from their address book or enter a new shipping address  **Alternative Scenario:**  Return to main success scenario.  **Exceptions:**   1. User not login  * Redirect to login page * Authenticate user and redirect to shipping address page.  1. Customer has no product in their shopping cart  * Show message inform to the user.   **Relationships: (List the relationships that use case relates to)**   1. Login 2. Shopping cart.   **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use-case No.** | UC10 | **Use-case Version** | 1.0 |
| **Use-case Name** | Add/Update/Delete Product | | |
| **Author** | Trinh Duc Trung | | |
| **Date** | 07/06/2013 | **Priority** | High |
| **Actor:**  Administrator.  **Summary:**  Administrator delete selected products.  **Goal:**  Allows administrator to deletes information of products  **Triggers:**  Display product information such as: Product Name , Price , Manufacture , Stock , Description and more.  **Preconditions:**  User has logged in system as Administrator.  **Post Conditions:**  The selected products are added,updated, deleted.  **Main Success Scenario:**   1. User logins to system as Administrator. 2. User chooses one product to delete information. 3. User clicks delete button and confirm that decision.     **Alternative Scenario:**  Back to Main Success Scenario  **Exceptions:**   1. User not login.  * Redirect to login page and authenticate the user. * If user is administrator, redirect to admin page.   **Relationships: (List the relationships that use case relates to)**  Login    **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC11 | **Use-case Version** | 1.0 |
| **Use-case Name** | Login | | |
| **Author** | Vu Van Tiep | | |
| **Date** | 06/06/2013 | **Priority** | Normal |
| **Actor:**  User  **Summary:**  Login to system.  **Goal:**  Allows user can sign in website to buy product or do another actions.  **Triggers**    **Preconditions:**  User had an account.  **Post Conditions:**    **Main Success Scenario:**   1. Enter email address. 2. Enter password. 3. Click login button. 4. System checks information. 5. Redirect to recent page (If user is Administrator ,redirect to admin page)   **Alternative Scenario:**  System validation: Incorrect email id or wrong password,system display warning message to alert user retry or register if don’t have any account)  **Exceptions:**  **Relationships: (List the relationships that use case relates to)**    **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC12 | **Use-case Version** | 1.0 |
| **Use-case Name** | Register | | |
| **Author** | Vu Van Tiep | | |
| **Date** | 06/06/2013 | **Priority** | Normal |
| **Actor:**  Guest  **Summary:**  Guest sign up an account on Shradha Online Bookstore  **Goal:**  Allows guest to become a User  **Triggers**  **Preconditions:**  **Post Conditions:**  **Main Success Scenario:**   1. Enter firstname. 2. Enter lastname 3. Enter email address. 4. Enter Telephone 5. Enter Fax 6. Enter First Adress 7. Enter City 8. Enter Postcode 9. Enter Country 10. Enter Region 11. Enter password. 12. Re – Enter password. 13. Insert new customer on database.   **Alternative Scenario:**  System checks information :Invalid - system display warning message to alert user re-type.  **Exceptions:**  There is an account has been logged in website on current browser.User will log out that account to sign up a new account.  **Relationships: (List the relationships that use case relates to)**  **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC13 | **Use-case Version** | 1.0 |
| **Use-case Name** | Forgot password | | |
| **Author** | Nguyen Cong Minh | | |
| **Date** | 29/05/2013 | **Priority** | Normal |
| **Actor:**  User  **Summary:**  User forgot password  **Goal:**  Allow user can get back their password to log in system to buy product.  **Triggers**    **Preconditions:**  User has an account to loged in system.  **Post Conditions:**    **Main Success Scenario:**   1. Enter email address. 2. Re – Enter email address. 3. Click get password button. 4. System checks information. 5. System display message. 6. Update new password on database.   **Alternative Scenario:**  System checks information: email not exists ,system show warning message “ Your e-mail address is not exist in our system,try again“(Or redirect to Register page)  **Exceptions:**  **Relationships: (List the relationships that use case relates to)**  **Business Rules:** | | | |

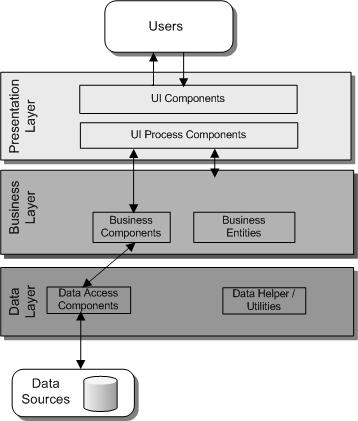
|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC14 | **Use-case Version** | 1.0 |
| **Use-case Name** | Change password | | |
| **Author** | Trinh Duc Trung | | |
| **Date** | 01/06/2013 | **Priority** | Normal |
| **Actor:**  User  **Summary:**  User wants to change password  **Goal:**  Allows user to change password.  **Triggers**    **Preconditions:**  User have an account and loged in system.  **Post Conditions:**    **Main Success Scenario:**   1. Enter new password. 2. Re – Enter new password. 3. Enter old password. 4. Click change password button. 5. System checks information. 6. System display result.   **Alternative Scenario:**  System checks information : Invalid : email id is incorrect ,password doesn’t match with confirm password , system display warning message “Email id is not correct” or “Password must match with confirm password”  **Exceptions:**  **Relationships: (List the relationships that use case relates to)**    **Business Rules:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE SPECIFICATION** | | | |
| **Use-case No.** | UC15 | **Use-case Version** | 1.0 |
| **Use-case Name** | Feedback. | | |
| **Author** | Vu Van Tiep | | |
| **Date** | 06/06/2013 | **Priority** | Normal |
| **Actor:**  Guest  User  **Summary:**  User inputs some feedback query and sends to website owner.  **Goal:**  Allow user can contact with administrator of website.  **Triggers**    **Preconditions:**  **Post Conditions:**   * The feedback is saved and delivered to sys admin   **Main Success Scenario:**  1. User/Guest links to ContactUs site.  2. User/Guest enters some information about them  3. User/Guest inputs feedback message.  4. User/Guest sends query to system  **Alternative Scenario:**  Return to main success scenario.  **Exceptions:**  User not input any message or information   * Show warning message * Redirect to ContactUs page   **Relationships: (List the relationships that use case relates to)**  **Business Rules:** | | | |

## Other Concerns<Optional>

# Design

## System Architecture



**3.5 *: Book Store Architecture***

## Class Diagram

**

*4.2.1: Databse Diagram*

## Class Diagram Explanation

In this section,we want toexplain about the class diagram above. We separate our Web-based application into three tiers.

|  |  |
| --- | --- |
| ***Class Name*** | ***Description*** |
| SqlDataHelper | Connection to SQL Server 2005 |
| CodeGenerator | Generate Product Code operations |
| SqlCodeGeneratorDataProvider | Provides Code Generated from database |
| Category | Category entity:attributes,properties |
| CategoryManager | Operations for Category entity |
| SqlCategoryDataProvider | Provides Category data from database |
| Product | Product entity:attributes,properties |
| ProductManager | Operations for Product entity |
| SqlProductDataProvider | Provides Product data from database |
| Order | Order entity:attributes,properties |
| OrderManager | Operations for Order entity |
| SqlOrderDataProvider | Provides Order data from database |
| ShoppingCart | ShoppingCart entity:attributes,properties |
| ShoppingCartManager | Operations for ShoppingCart entity |
| SqlShoppingCartDataProvider | Provides ShoppingCart data from database |
| Customer | Customer entity:attributes,properties |
| CustomerManager | Operations for Customer entity |
| SqlCustomerDataProvider | Provides Customer data from database |

## Sequence Diagram (Optional)

*<for important and complex interactions, protocols or algorithms, sequence diagrams should be drawn for clearing the details and supporting the system implementation. This section is optional>*

## Collaboration Diagram (Optional)

*<for important and complex interactions, collaboration diagrams should be drawn for clearing the details and supporting the system implementation. This section is optional>*

## State Diagram (Optional)

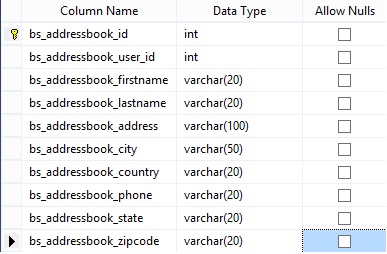
*<put all state diagrams here>*

## Entity Relationship Diagram

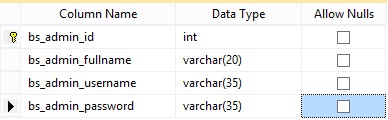
**

## Database Design

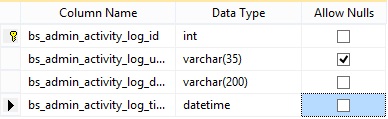
**Table : Adress Book**

****

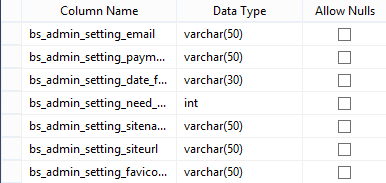
**Table : Admin**

****

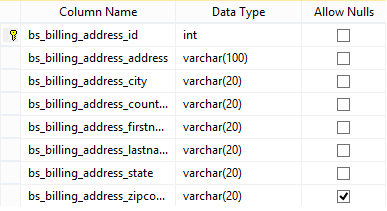
**Table : Admin Activity Log**

****

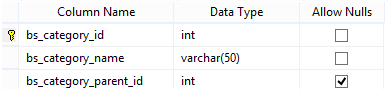
**Table : Admin Setting**

****

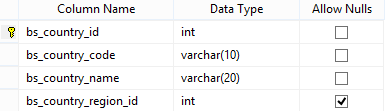
**Table : Billing Address**

****

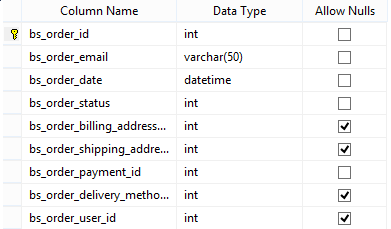
**Table : Category**

****

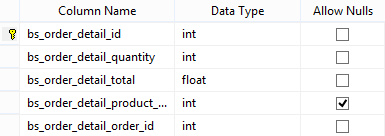
**Table : Country**

****

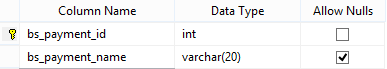
**Table : Order**

****

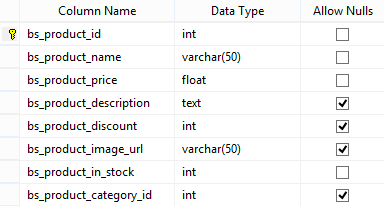
**Table : Order Details**

****

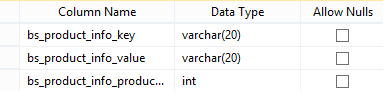
**Table : Payment**

****

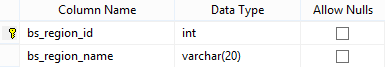
**Table : Product**

****

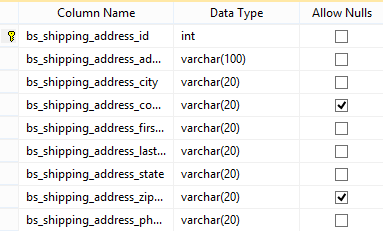
**Table : Product Info**

****

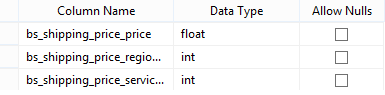
**Table : Region**

****

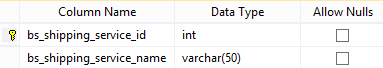
**Table : Shipping Address**

****

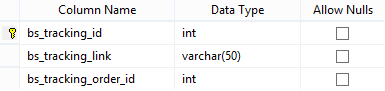
**Table : Shipping Price**

****

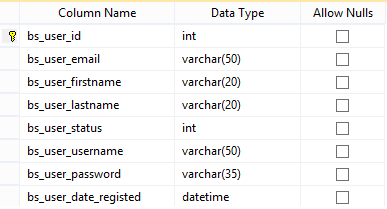
**Table : Shipping Service**

****

**Table : Tracking**

****

**Table : User**

****

## Algorithms (optional)



*4.9.1: Login Flow Chart*



4.9.2: Search Flow Chart



4.9.3: Register Flow Chart



4.9.4: View Product Details Flow Chart



* + 1. : Shopping Cart Flow Chart



4.9.6: Check Out Flow Chart



4.9.7: Add Product Flow Chart



4.9.8: Delete Product Flow Chart



4.9.10: Update Product Flow Chart

## Others (optional)

*<Any design concerns or diagrams can be put here>*

# System Prototype

# Management and Project Planning

## Management Approach

*<* Scrum has been used to develop complex products since the early 1990s. This section describes how to use Scrum to build our application. Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. The role of Scrum is to surface the relative efficacy of your development practices so that you can improve upon them while providing a framework within which complex products can be developed.

The Scrum framework consists of a set of Scrum Teams and their associated roles: Time-Boxes, Artifacts, and Rules.

* Scrum Teams are designed to optimize flexibility and productivity; to this end, they are self-organizing, they are cross-functional, and they work in iterations. Each Scrum Team has three roles: 1) the ScrumMaster, who is responsible for ensuring the process is understood and followed; 2) the Product Owner, who is responsible for maximizing the value of the work that the Scrum Team does; and 3) the Team, which does the work. The Team consists of developers with all the skills to turn the Product Owner's requirements into a potentially releasable piece of the product by the end of the Sprint.
* Scrum employs time boxes to create regularity. Elements of Scrum that are timeboxed include the Release Planning Meeting, the Sprint Planning Meeting, the Sprint, the Daily Scrum, the Sprint Review, and the Sprint Retrospective. The heart of Scrum is a Sprint, which is an iteration of one month or less that is of consistent length throughout a development effort. All Sprints use the same Scrum framework, and all Sprints deliver an increment of the final product that is potentially releasable. One Sprint starts immediately after the other.
* Scrum employs four principal artifacts. The Product Backlog is a prioritized list of everything that might be needed in the product. The Sprint Backlog is a list of tasks to turn the Product Backlog for one Sprint into an increment of potentially shippable product. A burndown is a measure of remaining backlog over time. A Release Burndown measures remaining Product Backlog across the time of a release plan. A Sprint Burndown measures remaining Sprint Backlog items across the time of a Sprint.

Rules bind together Scrum's time-boxes, roles, and artifacts. Its rules are described throughout the body of this document. For example, it is a Scrum rule that only Team members - the people committed to turning the Product Backlog into an increment - can talk during a Daily Scrum.

*.>*

## Project Plan

*<****6.2.1 The Rules of Scrum***

* The ScrumMaster assures everyone related to the project (Chickens and Pigs) follow the rules! The rules, which have been successfully used on thousands of projects, hold Scrum together and let everyone know how to play. Disputing rules waste time but they can be changed.
* Changes to rules originate solely from the Team and are discussed during the Sprint Retrospective Meeting. Rules can only be changed when the ScrumMaster determines that everyone involved understands Scrum enough to "skillfully and mindfully make such changes."

*6.2.2 Sprint Planning Meeting*

The Sprint Planning Meeting is time-boxed into two consecutive four hour segments. In Segment 1, the Team selects from the Product Backlog. In Segment 2, the Team prepares the Sprint.

#### Attendees

The Product Owner must be available for both meeting segments and is responsible for the Product Backlog (suggestions can come from the Team). If this is the first meeting then the Product Owner also prepares the business case, funding, contractual agreement, vision and stake holder buy-in.

The ScrumMaster takes over for the Product Owner if the Product Backlog or Product Owner is not available.

The Team, in segment 1, analyses the Product Backlog refining as many large-grained high priority items as possible in the four hours allotted. Further analysis takes place in the Sprint.

The Team commits to the analyzed items from the Product Backlog they believe can be turned into an "increment of potentially shippable product functionality" in a single Sprint. The increment is demonstrated to the Product Owner and Stakeholders at the Sprint Review Meeting held at the end of the Sprint.

The Team, in segment 2, completely on their own, plans how to turn the selected items into an "increment of potentially shippable product functionality." The Team can ask for help but all other attendees may not offer help or suggestions. The Sprint Backlog, created during this segment, can be incomplete but must be clear enough that the team can commit to the work for the first part of the sprint.

Others are allowed but dismissed immediately after providing additional business domain, technology domain and advice. No chickens as observers.

6.2.3 Daily Scrum Meeting

The Daily Scrum Meeting is time-boxed to 15 minutes (no matter how many team members) and is always held at the same place and time (preferably in the morning). In the meeting, the Team reports their status to each other (not the ScrumMaster).

**Attendees**

The Daily Scrum Meeting is time-boxed to 15 minutes (no matter how many team members) and is always held at the same place and time (preferably in the morning). In the meeting, the Team reports their status to each other (not the ScrumMaster).

Attendees

Team members must attend the meeting. When a member can not be present, they should attend by phone/internet or have another member report on their behalf. Team members who are late must immediately pay the ScrumMaster $1.00.

The team member immediately to the left of the ScrumMaster starts the meeting by answering the following:

What have you Done[[1]](#footnote-1) since the last Daily Scrum Meeting regarding this project?

What will you do between now and the next Daily Scrum Meeting regarding this project?

What impedes you from performing your work as effectively as possible?

Reporting continues counter clockwise with only ONE person talking at any one time. Upon hearing anything of interest or a request for assistance, any Team members can arrange to get interested parties together after the Daily Scrum Meeting.

The ScrumMaster assures answers are kept to the point: no issues, design problems or gossip allowed. The ScrumMaster may limit chicken attendance and assures everyone conforms to the rules. Non conformance means Chickens can be removed from the meeting and Team members from the Team. The ScrumMaster, after the meeting, works to remove any impediments.

Chickens stand on the peripherals the Team and may not talk, make faces or be obtrusive in any way during or after the meeting.

**6.2.4 *Sprint (Max 30 Consecutive Calendar Days[[2]](#footnote-2))***

The Sprint is 30 consecutive calendar days where the team turns the Sprint Backlog into an "increment of potentially shippable product."

**Attendees**

The Team has committed, in the Sprint Planning Meeting, to the Sprint Backlog. Corresponding Product Backlog items are frozen until the end of the Sprint. The Team is 100% self managed but can seek outside advice, help, information, direction, comments and support. However, NO ONE from the outside may offer any advice, help, information, direction, comments or support.

If the Team has overcommitted, they may work with the Product Owner  
to remove Sprint Backlog items. If the Sprint "looses its value," the  
ScrumMaster can abnormally terminate the Sprint. Time permitting,  
the Team can commit to more Product Backlog consulting with the  
Product Owner.

During the Sprint, Team member's administration responsibilities are:

Attend the Daily Scrum.

Update the Sprint Backlog by adding new tasks when conceived and for each task, update time estimates.

***6.2.5 Sprint Review Meeting (4 Hours)***

The Sprint Review Meeting is time-boxed to four hours. In this meeting, the Team demonstrates, to the Product Owner and Stakeholders, ONLY the functionality that is considered Done. Artifacts that aren't functionality can not be presented as work product but can be used to demonstrate functionality.

**Attendees**

The ScrumMaster should determine the number of people attending the Sprint Review Meeting and make appropriate accommodations.

A Team member starts the Sprint Review Meeting by presenting the Sprint goal, Product Backlog committed to and Product Backlog completed. Different Team members then discuss what did and did not go well during the Sprint.

2b) Team members then present, preferably from their workstation and against closest production server (usually QA), functionality that is done. Questions from Stakeholders are answered and noted.

3) Stakeholders, between presentations, can voice any comments,  
observations or criticisms regarding functionality just presented.

3b) After all presentations, Stakeholders are polled (one at a time) to get their impressions, desired changes and priority of changes (if any). Stakeholders can identify and request

A) functionality not delivered and/or

B) functionality not delivered as expected and/or

C) new functionality to be added to the Product Backlog.

The Product Owner discusses with Stakeholders and the Team potential rearrangement of the Product Backlog.

Finally, the ScrumMaster announces the place and time of next Sprint Review Meeting.

*6.2.6 Sprint Restrospective ( 3 Hours)*

The Sprint Retrospective Meeting is time-boxed to three hours and allows the team to reflect on the last Sprint.

**Attendees**

Team members start the meeting by answering the following:

What went well during the last Sprint?

What could be potentially improved on in the next Sprint?

Answers are written down by the ScrumMaster. The Team prioritizes potential improvements to discuss.

The ScrumMaster facilitates the Team's search for ways to improve on how Scrum can work for them. Actionable items are added to the Product Backlog (and so next Sprint) as "high-priority non-functional action items."

The Product Owner may (optionally) attend the meeting.

*>*

## Task Sheet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Ref No. | | Date of Preparation of Activity Plan: | | | |
| No. | Task | Actual Start Date | Actual Days | Team Members Names | Status |
| 01 | Create process document | 23/5 | 2 | Vu Van Tiep | Complete |
| 02 | Problem Definition | 23/5 | 5 | All team | Complete |
| 03 | Analysis CRS | 23/5 | 7 | All team | Complete |
| 04 | Write Checklist of Validations | 25/5 | 1 | All team | Complete |
| 06 | Architecture and Design | 25/5 | 5 | All team | Complete |
| 07 | Write Use Case | 25/5 | 3 | All team | Complete |
| 08 | Write DFD | 26/5 | 3 | All team | Complete |
| 09 | Flow Chart | 26/5 | 3 | All team | Complete |
| 10 | Entity Relationship Diagram (ERD) | 26/5 | 5 | Vu Van Tiep | Complete |
| 11 | Database Design | 26/05 | 5 | All team | Complete |
| 12 | Database Stored Procedure | 27/05 | 7 | All team | Complete |
| 13 | Write Coding Standards | 23/05 | 5 | Nguyen Cong Minh | Complete |
| 14 | Prototype | 23/05 | 5 | All team | Complete |
| 15 | Homepage | 27/05 | 3 | Vu Van Tiep | Complete |
| 16 | Master Page | 27/05 | 3 | Vu Van Tiep | Complete |
| 17 | CSS&Theme | 27/05 | 5 | Vu Van Tiep | Complete |
| 18 | Admin Site Master Page | 27/05 | 3 | Trinh Duc Trung | Complete |
| 19 | Login Page | 28/05 | 3 | Vu Van Tiep | Complete |
| 20 | Register Page | 28/05 | 3 | Vu Van Tiep | Complete |
| 21 | Customer Pages | 30/05 | 5 | Nguyen Cong Minh | Complete |
| 22 | Search Web User Controls | 30/05 | 5 | Nguyen Cong Minh | Complete |
| 23 | Product Pages | 28/05 | 7 | Trinh Duc Trung | Complete |
| 24 | Category Pages | 28/05 | 7 | All Team | Complete |
| 25 | Order Pages | 29/05 | 7 | Nguyen Cong Minh | Complete |
| 26 | Shopping Cart | 06/06 | 3 | Vu Van Tiep | Complete |
| 27 | Best Seller | 04/06 | 1 | Trinh duc Trung | Complete |
| 28 | Feature Product | 04/06 | 1 | Trinh duc Trung | Complete |
| 29 | Paging | 05/06 | 1 | Nguyen Cong Minh | Complete |
| 30 | Checkout Step Pages | 30/05 | 5 | All team | Complete |
| 31 | Feedback Pages | 07/06 | 1 | Vu Van Tiep | Complete |
| 32 | FAQs Pages | 29/05 | 1 | Trinh Duc Trung | Complete |
| 33 | About Us,Condition of Use,Privacy Notice | 03/06 | 1 | Nguyen Cong Minh | Complete |
| 34 | Validations | 06/06 | 4 | Vu Van Tiep | Complete |
| 35 | Testing | 08/06 | 7 | Trinh Duc Trung | Complete |
| 36 | Debugging | 14/06 | 7 | All team | Complete |
| 37 | Installation Guide | 13/06 | 1 | All Team | Complete |
| 38 | User Manual | 12/06 | 3 | All Team | Complete |
| 39 | Screenshots | 10/06 | 1 | Vu Van Tiep | Complete |
| 40 | Documentation | 11/06 | 20 | Nguyen Cong Minh | Complete |

## Meeting Minutes (Optional)

*<Put all minutes of your team meetings here>*

# Checklists

## Check List of Validation

*<*

|  |  |
| --- | --- |
| **Option** | Validated |
| Can a new user who gets registered,enter the Web site after logging ? | Yes |
| Can users find products which they want to buy ? | Yes |
| Do all the links navigate to the correct Web pages ? | Yes |
| Does the Web sites’s functionality resolve the client problem,and satisfy his needs ? | Yes |
| Has the hardware and software been correctly chosen ? | Yes |

*>*

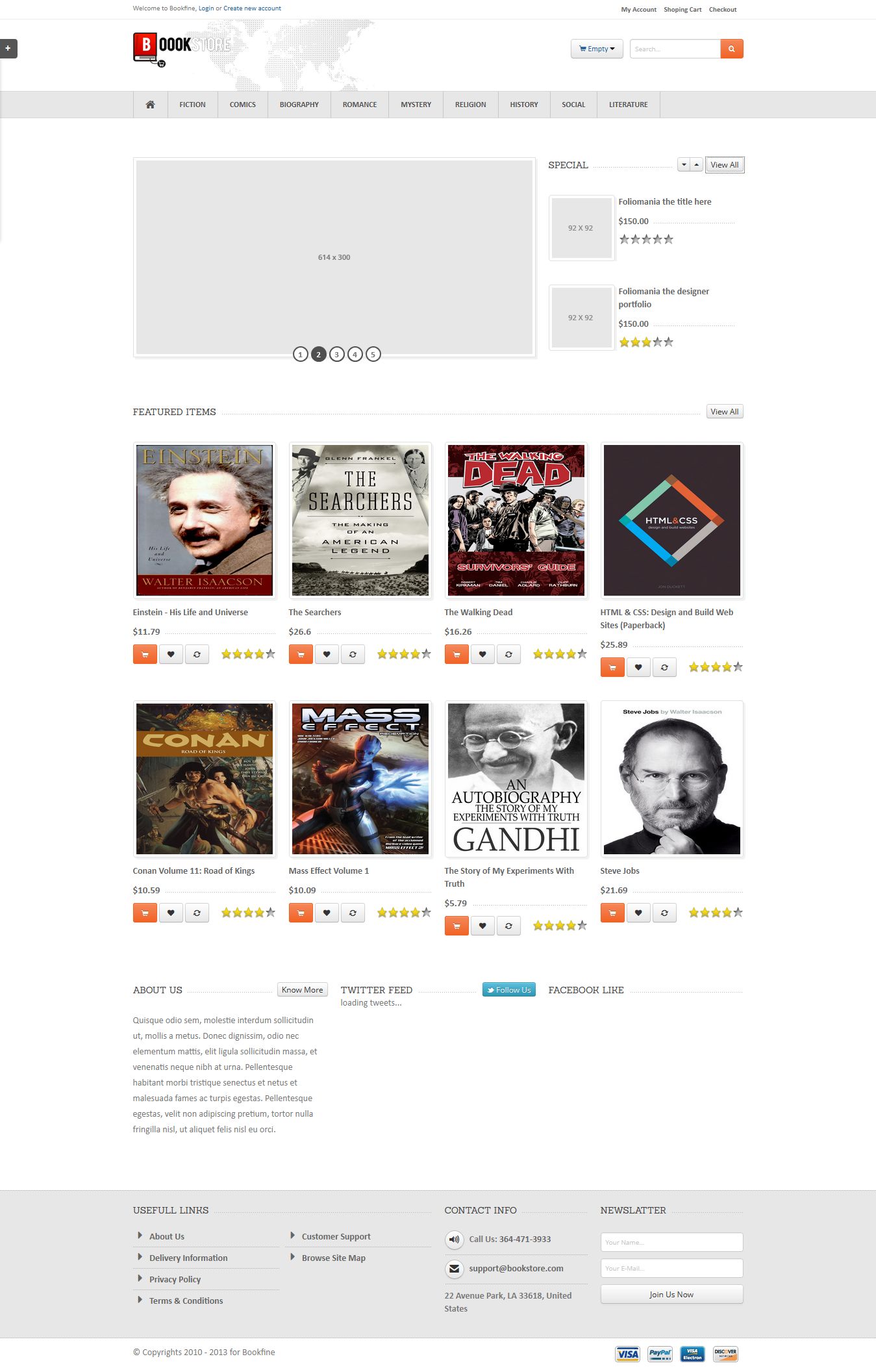
## Submission Checklist

*<*

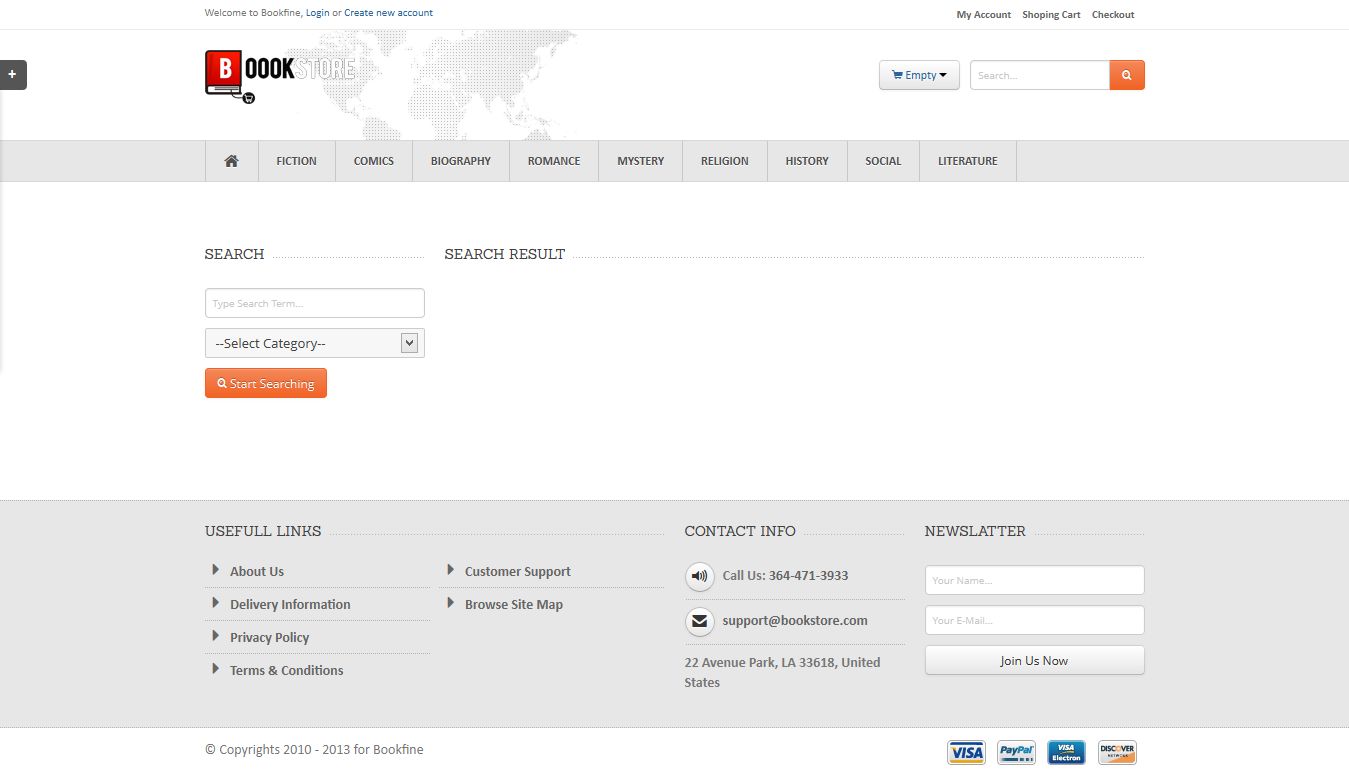
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Particulars** | **Yes** | **No** | **NA** | **Comments** |
| 1 | Are the users able to enter the Web site after validation is performed on the User‘s EmailID and Password | Yes |  |  |  |
| 2 | Are the users able to enter the Web site after getting registered ? | Yes |  |  |  |
| 3 | Do all the Web page contents are devoid of spelling mistakes ? | Yes |  |  |  |
| 4 | Is the user able to purchase the book which has been selected ? | Yes |  |  |  |
| 5 | Is the Web site user-friendly ? | Yes |  |  | Tested on Firefox, IE 8, Google Chrome |

*>*

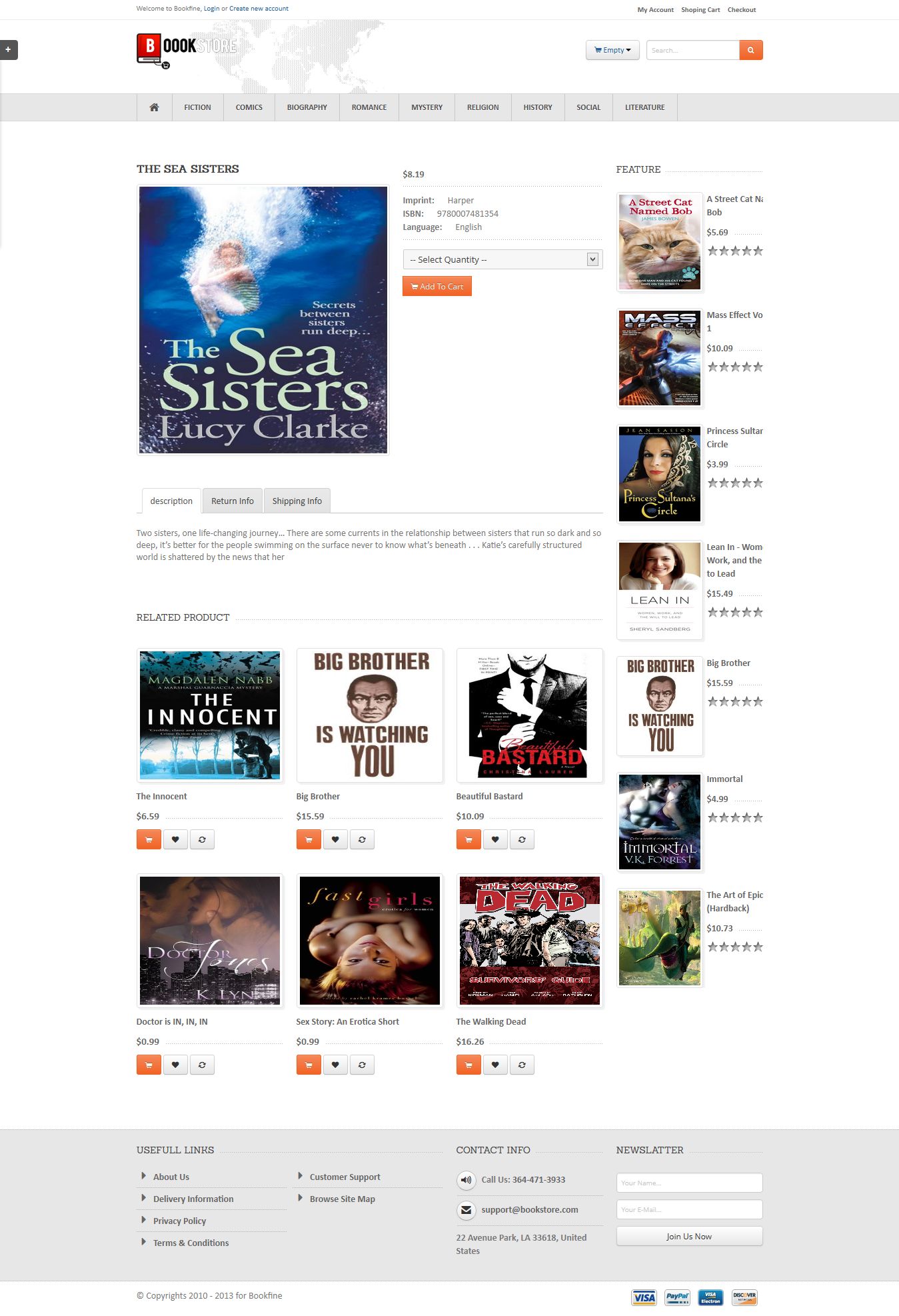
# Screenshots

**

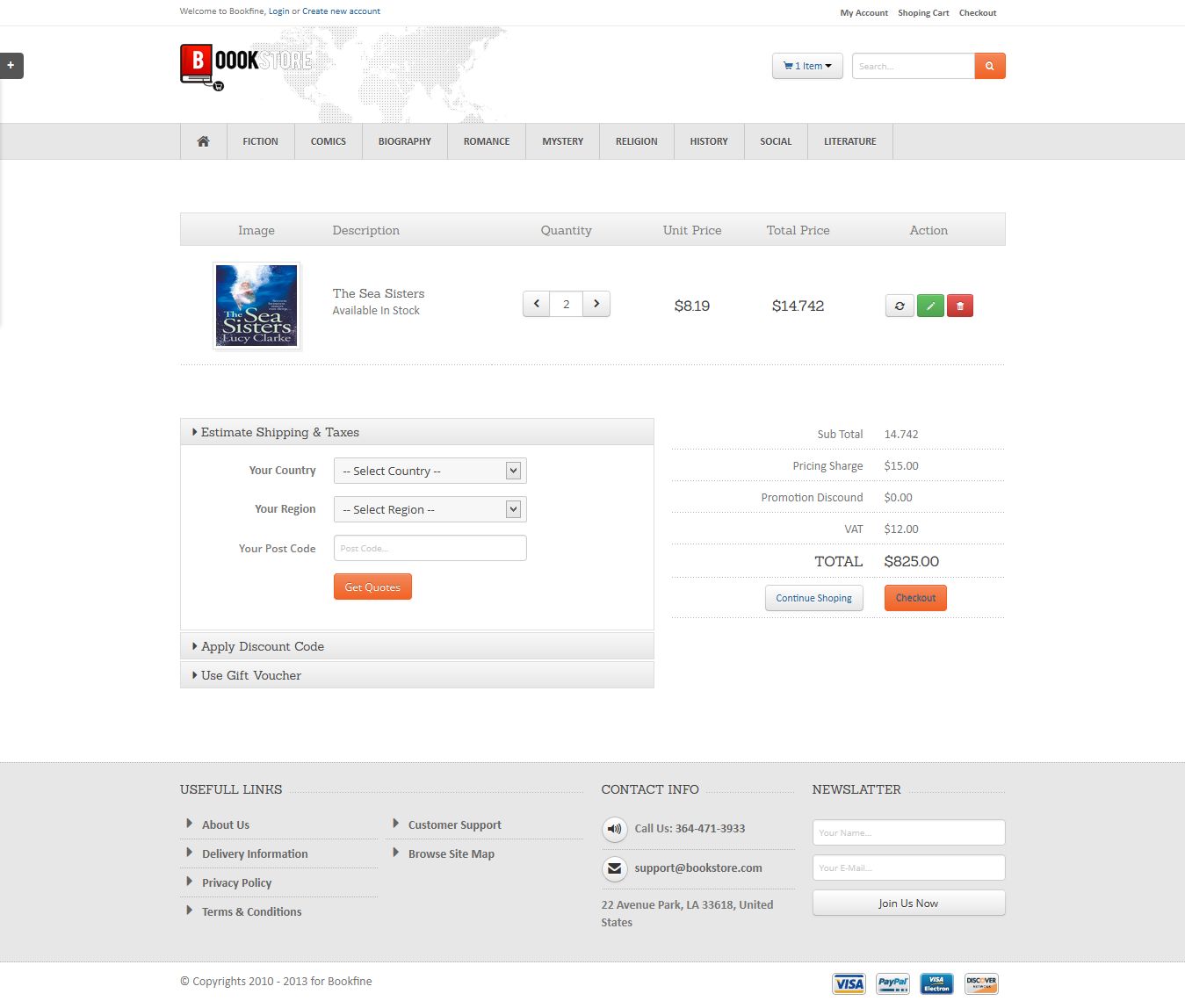
**Screenshot – *Home Page***

******

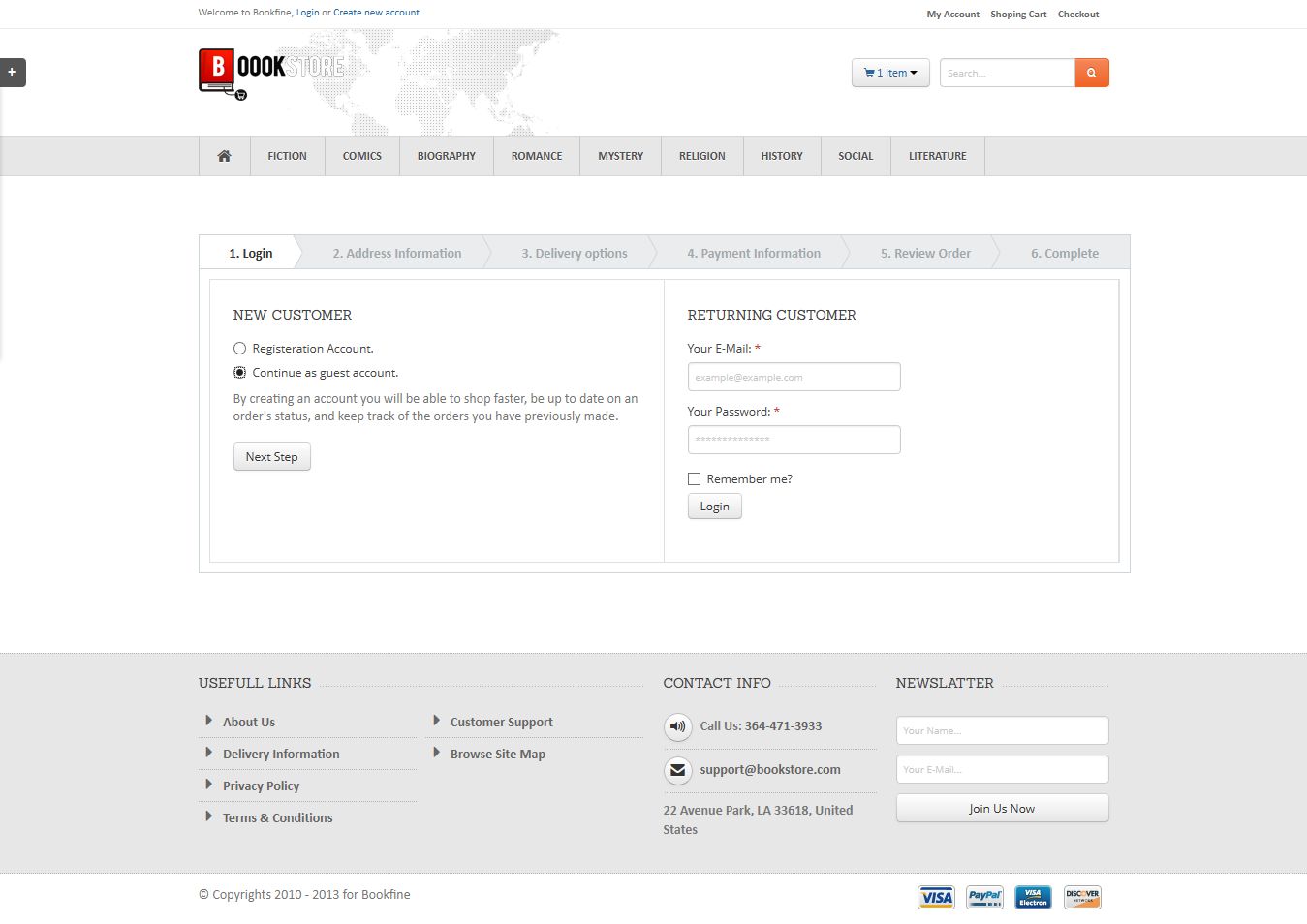
**Screenshot – *Search***

**

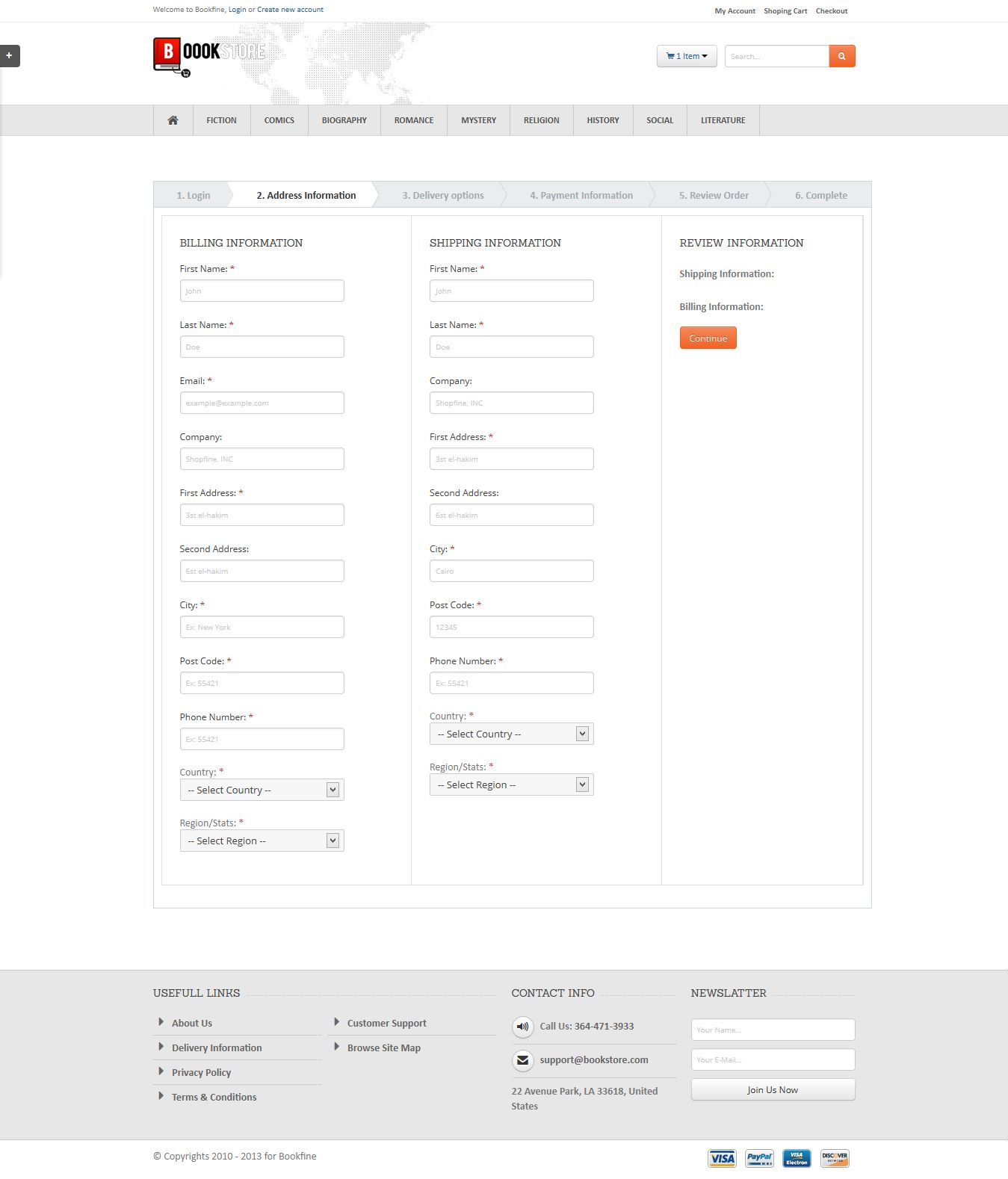
**Screenshot – *Product Details***

**

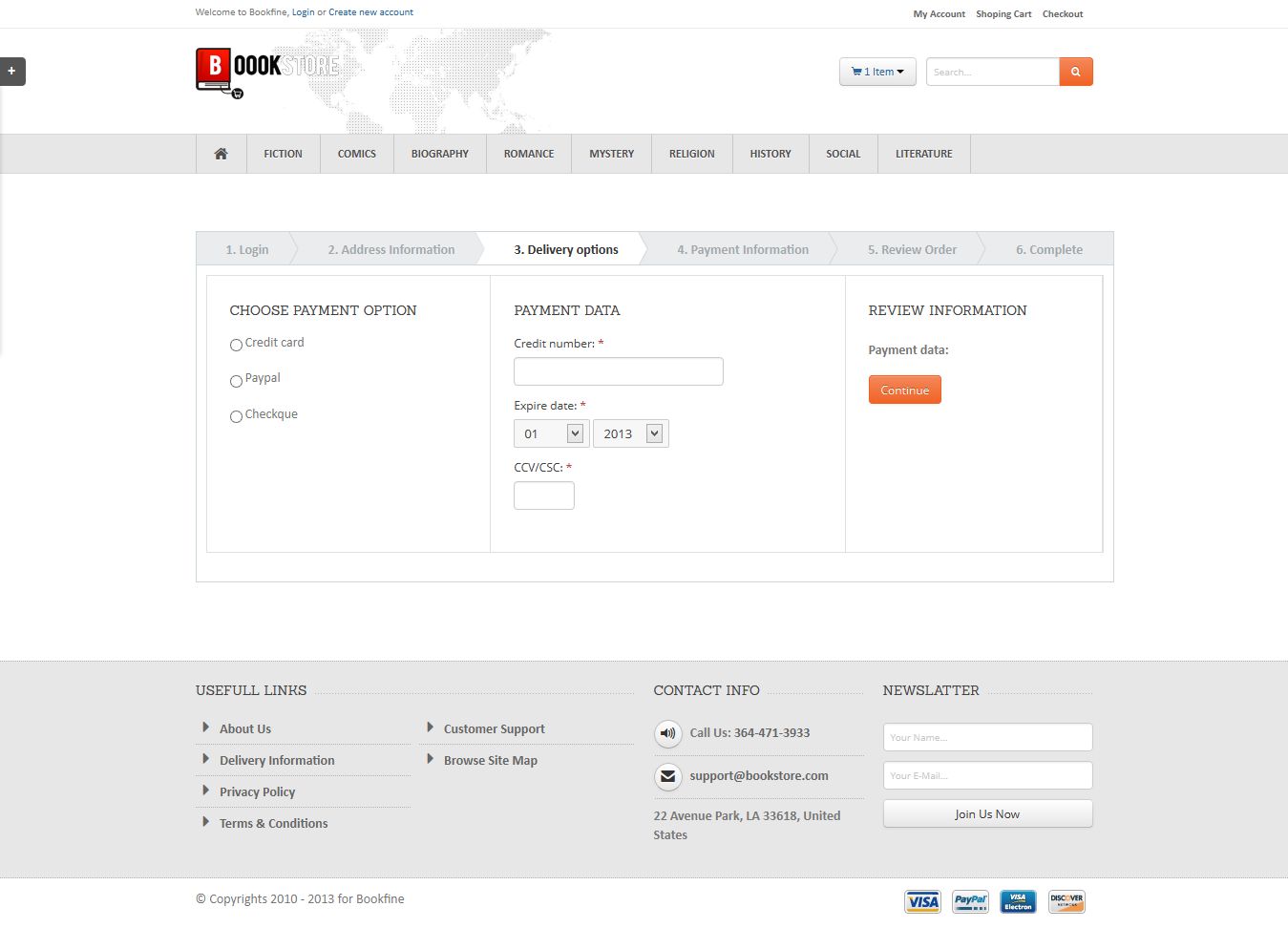
**Screenshot – Shopping Cart**

**

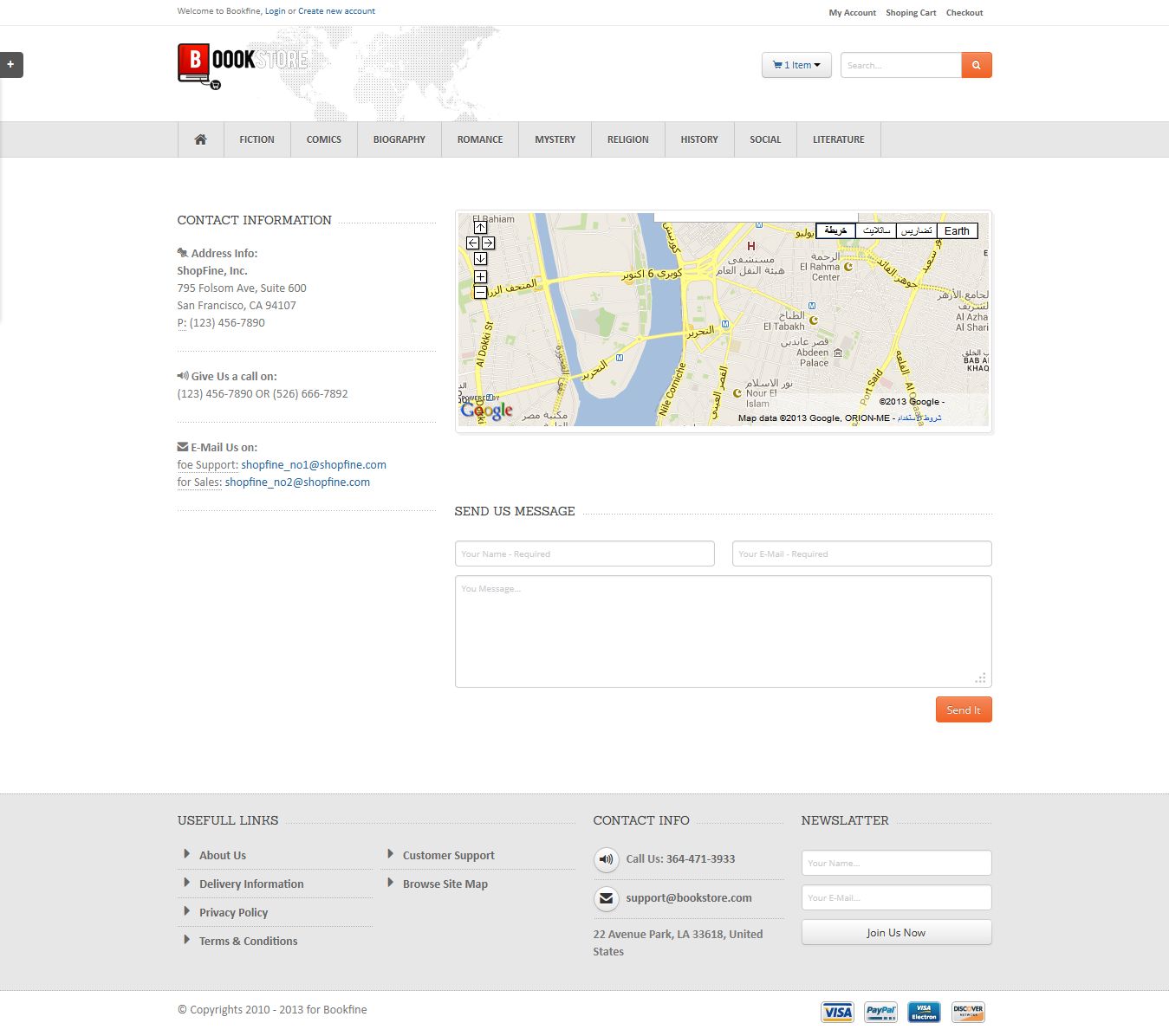
**Screenshot - *Checkout Step 1/6***

****

**Screenshot - *Checkout Step 2/6***

**

**Screenshot - *Checkout Step 3/6***

**

**Screenshot - *Feedback***

# Coding Convention

***9.1 Objective***

This section requires or recommends certain practices for developing programs in the C# language. The objective of this coding standard is to have a positive effect on

* Avoidance of errors/bugs, especially the hard-to-find ones.
* Maintainability, by promoting some proven design principles.
* Maintainability, by requiring or recommending a certain unity of style.
* Performance, by dissuading wasteful practices.

***9.2 Scope***

* This standard pertains to the use of the C# language. With very few exceptions, it does not discuss the use of the .NET class libraries. Certain items that deserve attention have been identified, but have not been included in this section because treatment in separate documents appears more appropriate.
* This standard does not include rules or recommendations on how to **layout brackets, braces, and code in general.**

***9.3 Rules:***

**9.3.1 Name Convetions and Standards:**

Note :

The terms Pascal Casing and Camel Casing are used throughout this document.

*Pascal Casing* - First character of all words are Upper Case and other characters are lower case.

Example: BackColor

*Camel Casing* - First character of all words, except the first word are Upper Case and other characters are lower case.

Example: backColor

1. ***Use Pascal casing for Class names***

public class **HelloWorld**

{

...

}

1. ***Use Pascal casing for Method names***

void **SayHello**(string name)

{

...

}

1. ***Use Camel casing for variables and method parameters***

Example:

int *totalCount* = 0;

void SayHello(string name)

{

string *fullMessage* = "Hello " + name;

...

}

1. ***Use the prefix “I” with Camel Casing for interfaces ( Example: IEntity )***
2. ***Do not use Hungarian notation to name variables.***

In earlier days most of the programmers liked it - having the data type as a prefix for the variable name and using m\_ as prefix for member variables. Eg:

string m\_sName;

int nAge;

However, in .NET coding standards, this is not recommended. Usage of data type and m\_ to represent member variables should not be used. All variables should use camel casing.

|  |
| --- |
| Some programmers still prefer to use the prefix **m\_** to represent member variables, since there is no other easy way to identify a member variable. |

1. ***Use Meaningful, descriptive words to name variables. Do not use abbreviations.***
2. ***Do not use single character variable names like i, n, s etc. Use names like index,temp.***
3. ***Do not use underscores (\_) for local variable names***.
4. ***All member variables must be prefixed with underscore (\_) so that they can be identified from other local variables.***
5. ***Do not use variable names that resemble keywords.***
6. ***Prefix boolean variables, properties and methods with “is” or similar prefixes.***

Ex: private bool \_isFinished

1. ***Namespace names should follow the standard pattern***

<company name>.<product name>.<top level module>.<bottom level module>

1. ***Use appropriate prefix for the UI elements so that you can identify them from the rest of the variables.***

There are 2 different approaches recommended here.

* 1. Use a common prefix ( ui\_ ) for all UI elements. This will help you group all of the UI elements together and easy to access all of them from the intellisense.
  2. Use appropriate prefix for each of the ui element. A brief list is given below. Since .NET has given several controls, you may have to arrive at a complete list of standard prefixes for each of the controls (including third party controls) you are using.

|  |  |
| --- | --- |
| **Control** | **Prefix** |
| Label | lbl |
| TextBox | txt |
| DataGrid | dtg |
| Button | btn |
| ImageButton | imb |
| Hyperlink | hlk |
| DropDownList | ddl |
| ListBox | lst |
| DataList | dtl |
| Repeater | rep |
| Checkbox | chk |
| CheckBoxList | cbl |
| RadioButton | rdo |
| RadioButtonList | rbl |
| Image | img |
| Panel | pnl |
| PlaceHolder | phd |
| Table | tbl |
| Validators | val |

1. ***File name should match with class name.***

For example, for the class HelloWorld, the file name should be helloworld.cs (or, helloworld.vb)

1. ***Use Pascal Case for file names.***

**9.3.2 Indentation and Spacing**

1. Use TAB for indentation. Do not use SPACES. Define the Tab size as 4.
2. Comments should be in the same level as the code (use the same level of indentation).
3. Curly braces ( {} ) should be in the same level as the code outside the braces.

1. Use one blank line to separate logical groups of code
2. There should be one and only one single blank line between each method inside the class.
3. The curly braces should be on a separate line and not in the same line as if, for etc.
4. Use a single space before and after each operator and brackets.
5. Use #region to group related pieces of code together. If you use proper grouping using #region, the page should like this when all definitions are collapsed.
6. Keep private member variables, properties and methods in the top of the file and public members in the bottom.

**9.3.3 Good Programming practies:**

1. Avoid writing very long methods. A method should typically have 1~25 lines of code. If a method has more than 25 lines of code, you must consider re factoring into separate methods.
2. Method name should tell what it does. Do not use mis-leading names. If the method name is obvious, there is no need of documentation explaining what the method does.
3. A method should do only 'one job'. Do not combine more than one job in a single method, even if those jobs are very small.
4. Use the c# or VB.NET specific types (aliases), rather than the types defined in System namespace.
5. Always watch for unexpected values. For example, if you are using a parameter with 2 possible values, never assume that if one is not matching then the only possibility is the other value.
6. Do not hardcode numbers. Use constants instead. Declare constant in the top of the file and use it in your code.
7. However, using constants are also not recommended. You should use the constants in the config file or database so that you can change it later. Declare them as constants only if you are sure this value will never need to be changed.
8. Do not hardcode strings. Use resource files.
9. Convert strings to lowercase or upper case before comparing. This will ensure the string will match even if the string being compared has a different case.
10. Use String.Empty instead of “”
11. Avoid using member variables. Declare local variables wherever necessary and pass it to other methods instead of sharing a member variable between methods. If you share a member variable between methods, it will be difficult to track which method changed the value and when.
12. Use enum wherever required. Do not use numbers or strings to indicate discrete values.
13. Do not make the member variables public or protected. Keep them private and expose public/protected Properties.
14. The event handler should not contain the code to perform the required action. Rather call another method from the event handler.
15. Do not programmatically click a button to execute the same action you have written in the button click event. Rather, call the same method which is called by the button click event handler.
16. Never hardcode a path or drive name in code. Get the application path programmatically and use relative path.
17. Never assume that your code will run from drive "C:". You may never know, some users may run it from network or from a "Z:".
18. In the application start up, do some kind of "self check" and ensure all required files and dependancies are available in the expected locations. Check for database connection in start up, if required. Give a friendly message to the user in case of any problems.
19. If the required configuration file is not found, application should be able to create one with default values.
20. If a wrong value found in the configuration file, application should throw an error or give a message and also should tell the user what are the correct values.
21. Error messages should help the user to solve the problem. Never give error messages like "Error in Application", "There is an error" etc. Instead give specific messages like "Failed to update database. Please make sure the login id and password are correct."
22. When displaying error messages, in addition to telling what is wrong, the message should also tell what should the user do to solve the problem. Instead of message like "Failed to update database.", suggest what should the user do: "Failed to update database. Please make sure the login id and password are correct."
23. Show short and friendly message to the user. But log the actual error with all possible information. This will help a lot in diagnosing problems.
24. Do not have more than one class in a single file.
25. Have your own templates for each of the file types in Visual Studio. You can include your company name, copy right information etc in the templateAvoid having very large files. If a single file has more than 1000 lines of code, it is a good candidate for refactoring. Split them logically into two or more classes.
26. Avoid public methods and properties, unless they really need to be accessed from outside the class. Use “internal” if they are accessed only within the same assembly.
27. Avoid passing too many parameters to a method. If you have more than 4~5 parameters, it is a good candidate to define a class or structure.
28. If you have a method returning a collection, return an empty collection instead of null, if you have no data to return. For example, if you have a method returning an ArrayList, always return a valid ArrayList. If you have no items to return, then return a valid ArrayList with 0 items. This will make it easy for the calling application to just check for the “count” rather than doing an additional check for “null”.
29. Use the AssemblyInfo file to fill information like version number, description, company name, copyright notice etc.
30. Logically organize all your files within appropriate folders. Use 2 level folder hierarchies. You can have up to 10 folders in the root folder and each folder can have up to 5 sub folders. If you have too many folders than cannot be accommodated with the above mentioned 2 level hierarchy, you may need re factoring into multiple assemblies.
31. Make sure you have a good logging class which can be configured to log errors, warning or traces. If you configure to log errors, it should only log errors. But if you configure to log traces, it should record all (errors, warnings and trace). Your log class should be written such a way that in future you can change it easily to log to Windows Event Log, SQL Server, or Email to administrator or to a File etc without any change in any other part of the application. Use the log class extensively throughout the code to record errors, warning and even trace messages that can help you trouble shoot a problem.
32. If you are opening database connections, sockets, file stream etc, always close them in the finally block. This will ensure that even if an exception occurs after opening the connection, it will be safely closed in the finally block.
33. Declare variables as close as possible to where it is first used. Use one variable declaration per line.
34. Use StringBuilder class instead of String when you have to manipulate string objects in a loop. The String object works in weird way in .NET. Each time you append a string, it is actually discarding the old string object and recreating a new object, which is a relatively expensive operations.

**9.3.4 Architecture:**

1. Always use multi layer (N-Tier) architecture.
2. Never access database from the UI pages. Always have a data layer class which performs all the database related tasks. This will help you support or migrate to another database back end easily.
3. Use try-catch in your data layer to catch all database exceptions. This exception handler should record all exceptions from the database. The details recorded should include the name of the command being executed, stored proc name, parameters, connection string used etc. After recording the exception, it could be re thrown so that another layer in the application can catch it and take appropriate action.
4. Separate your application into multiple assemblies. Group all independent utility classes into a separate class library. All your database related files can be in another class library.

**9.3.5 ASP.Net**

1. Do not use session variables throughout the code. Use session variables only within the classes and expose methods to access the value stored in the session variables. A class can access the session using System.Web.HttpCOntext.Current.Session
2. Do not store large objects in session. Storing large objects in session may consume lot of server memory depending on the number of users.
3. Always use style sheet to control the look and feel of the pages. Never specify font name and font size in any of the pages. Use appropriate style class. This will help you to change the UI of your application easily in future. Also, if you like to support customizing the UI for each customer, it is just a matter of developing another style sheet for them

**9.3.6 Comments:**

Good and meaningful comments make code more maintainable. However,

* Do not write comments for every line of code and every variable declared.
* Use // or /// for comments. Avoid using /\* … \*/
* Write comments wherever required. But good readable code will require very less comments. If all variables and method names are meaningful, that would make the code very readable and will not need many comments.
* Do not write comments if the code is easily understandable without comment. The drawback of having lot of comments is, if you change the code and forget to change the comment, it will lead to more confusion.
* Fewer lines of comments will make the code more elegant. But if the code is not clean/readable and there are less comments, that is worse.
* If you have to use some complex or weird logic for any reason, document it very well with sufficient comments.
* If you initialize a numeric variable to a special number other than 0, -1 etc, document the reason for choosing that value.
* The bottom line is, write clean, readable code such a way that it doesn't need any comments to understand.
* Perform spelling check on comments and also make sure proper grammar and punctuation is used.

**9.3.7 Exception Handling**

* Never do a 'catch exception and do nothing'. If you hide an exception, you will never know if the exception happened or not. Lot of developers uses this handy method to ignore non significant errors. You should always try to avoid exceptions by checking all the error conditions programmatically. In any case, catching an exception and doing nothing is not allowed. In the worst case, you should log the exception and proceed.
* In case of exceptions, give a friendly message to the user, but log the actual error with all possible details about the error, including the time it occurred, method and class name etc.
* Always catch only the specific exception, not generic exception.
* No need to catch the general exception in all your methods. Leave it open and let the application crash. This will help you find most of the errors during development cycle. You can have an application level (thread level) error handler where you can handle all general exceptions. In case of an 'unexpected general error', this error handler should catch the exception and should log the error in addition to giving a friendly message to the user before closing the application, or allowing the user to 'ignore and proceed'.
* When you re throw an exception, use the throw statement without specifying the original exception. This way, the original call stack is preserved.
* Do not write try-catch in all your methods. Use it only if there is a possibility that a specific exception may occur and it cannot be prevented by any other means. For example, if you want to insert a record if it does not already exists in database, you should try to select record using the key. Some developers try to insert a record without checking if it already exists. If an exception occurs, they will assume that the record already exists. This is strictly not allowed. You should always explicitly check for errors rather than waiting for exceptions to occur. On the other hand, you should always use exception handlers while you communicate with external systems like network, hardware devices etc. Such systems are subject to failure anytime and error checking is not usually reliable. In those cases, you should use exception handlers and try to recover from error.
* Do not write very large try-catch blocks. If required, write separate try-catch for each task you perform and enclose only the specific piece of code inside the try-catch. This will help you find which piece of code generated the exception and you can give specific error message to the user.
* Write your own custom exception classes if required in your application. Do not derive your custom exceptions from the base class SystemException. Instead, inherit from ApplicationException.

# Other Concerns<Optional>

*<If you have any other information you want to add to this document, place it here. This could include thoughts on the eProject, improvements, etc.>*

# Appendix

## Glossary [Optional]

***Chicken*** - Interested but not committed or accountable (not a Pig).

***Daily Scrum Meeting*** - Short meeting where Team members present their status: impediments noted by ScrumMaster.

***Done*** - Complete, as mutually agreed to by all parties, and conforming to an organizations standards, conventions and guidelines.

***Estimated Work Remaining*** - The estimated total hours remaining on any one task which is updated at the end of every Sprint Backlog working day.

***Increment*** - Product functionality developed during a Sprint.

***Increment of Potentially Shippable Product Functionality*** - A complete developed increment that contains all parts of a completed product (Note: Does not include Product Backlog Items of current Sprint).

***Iteration*** - A project cycle which in Scrum is 30 sequential calendar days: a Sprint.

***Pig*** - A Team member, Product Owner, ScrumMaster or other Stakeholder committed and accountable to the project.

***Product Owner*** - Represents all Stakeholders and manages Product Backlog to maximize project value.

***Scrum*** - A mechanism in the game of rugby for getting an out-of-play ball back into play.

***ScrumMaster*** - Person responsible for the Scrum process, its correct implementation and to maximize on its benefits.

***Sprint*** - 30 sequential calendar days (time-boxed) in which a Team attempts to turn the Sprint Backlog into an "increment of potentially shippable product functionality."

Stakeholder - People interested in the project because they funded it, will use it or will be affected by it.

***Team*** - Cross-functional group of self managed people who develop product during Sprints.

***Time-boxed*** - Period of time, within a meeting or event, that may not be exceeded.

## References [Optional]

*"Agile Retrospectives: Making Good Teams Great,"Esther Derby and Diana Larsen, Pragmatic Bookshelf, 2006.*

*"User Stories Applied: For Agile Software Development," Mike Cohn, Addison-Wesley, 2004. "Writing Effective Use Cases,"Alistair Cockburn, Addison-Wesley, 2000.*

*“Schwaber, K. 2004, Agile Project Management with Scrum: Scrum Rules, Microsoft Professional”*

## Others<Optional>

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)