INVENTORY MANAGEMENT FOR RETAILERS

(ONLINE BOOK STORE)

1. INTRODUCTION:

The main objective of the project is to create an online book store that allows users to search and purchase a book based on title, author and subject. The selected books are displayed in a tabular format and the user can order their books online through credit card payment. The Administrator will have additional functionalities when compared to the common user.

PROJECT OVERVIEW:

There are many online book stores like Powell's, Amazon which were designed using Html. I want to develop a similar website using .NET, SQL Server.

Online Book store is an online web application where the customer can purchase books online. Through a web browser the customers can search for a book by its title or author, later can add to the shopping cart and finally purchase using credit card transaction. The user can login using his account details or new customers can set up an account very quickly. They should give the details of their name, contact number and shipping address. The user can also give feedback to a book by giving ratings on a score of five.

1.1 PURPOSE:

This is the page where the user will be navigated after a successful login. It will display all the book categories and have a search keyword option to search for the required book. It also includes some special sections titles, weekly special books. If the user would like to know details about a book he can click on the title from where he will be directed to a Book description page. It includes the notes on the book content and also a link to Amazon.com to get the book review.

2.LITERATURE SURVEY:

As e-book formats emerged and proliferated, some garnered support from major software companies, such as Adobe with its PDF format that was introduced in 1993.Unlike most other formats, PDF documents are generally tied to a particular dimension and layout, rather than adjusting dynamically to the current page, window, or other size. Different e-reader devices followed different formats, most of them accepting books in only one or a few format, thereby fragmenting the e-book market even more. Due to the exclusiveness and limited readerships of e-books, the fractured market of independent publishers and specialty authors lacked consensus regarding a standard for packaging and selling e-books.

1.2 EXISTING PROBLEM:

Vision Document will include the project's requirements and overview. It includes overview of the project, its purpose, goals, risks, constraints, and direction. It gives a listing of the main requirements and their respective Use case models to illustrate the functionality. Project Plan will detail the phases, iterations, and milestones that will comprise the project. It will include a timeline for the project and a cost estimate for completing this project. It includes the Architecture Elaboration plan will define the activities and actions that must be accomplished before the Architecture Presentation. Software Quality Assurance Plan describes the required documentation, standards and conventions test tracking and problem reporting, and tools used during the project. The plan will also identify the set of quality metrics used to assess product reliability. Demonstration of at least one executable prototype is required. Projects with a graphical user interface will include an executable prototype of the user interface.

User Module

- Only register user can login into the application.

- Those users who are not register must register first by filling the necessary attributes such as name, email, password etc. and then the account will be created by providing user name and password.
- User can read the contents of the application such as notes, books, syllabus etc.
- User can watch the videos available in application.
- Interested users can also suggest and provide study materials to admin.

Admin Module

- Admin can upload the study materials such as notes, books, syllabus, old question collection etc.
- Admin can check, verify and post the study materials that are provided by the user.

1.3 REFERENCES:

2. Lingxi Meng.Design of Online Bookstore System Based on B2C Model.Value engineering,2011,(36):101-102.

3. Yingqian Tan.Research on Evaluation Index System of Online Bookstore Based on University Students' Satisfaction.Journal of Sun Yat-sen University (Social Science Edition),2011,(4):174-184.

4. Yinghui Xu.Design and Realization of Online Used Book Sale System.Consumer Electronics,2012,(9):67-68.

5. Zhuanhong Chen,Huan Li.A typical application of jQuery in WEB query module.Electronic Technology,2013,(12
):28-32.

6. Paz J R G. Ajax and jQuery.Beginning Asp Net Mvc,2013,(3):11-13.

7. Dan Deng.The Operating Mode and Development Prospect of China 's Online Bookstore.Entrepreneur World,2013,(4):110-111.

7.1 PROBLEM STATEMENT DEFINITION:

The Online Book Store Project helped me to improve my confidence level in C# Programming. Though I have made many mistakes during the initial phase I have learnt how to use user controls, master pages, data grid, data set and other data base functionalities. Since MSE Project is done as an individual I have learnt how to manage time during the Software Life Cycle Process. I have also learned how o face tense situations and meet the deadlines. This would add as a good experience for me for my future job prospective**.**

8. IDEATION AND PROPOSED SOLUTION:

- Searching books in library is time consuming.

- The website is lagging and doesn't have mobile version

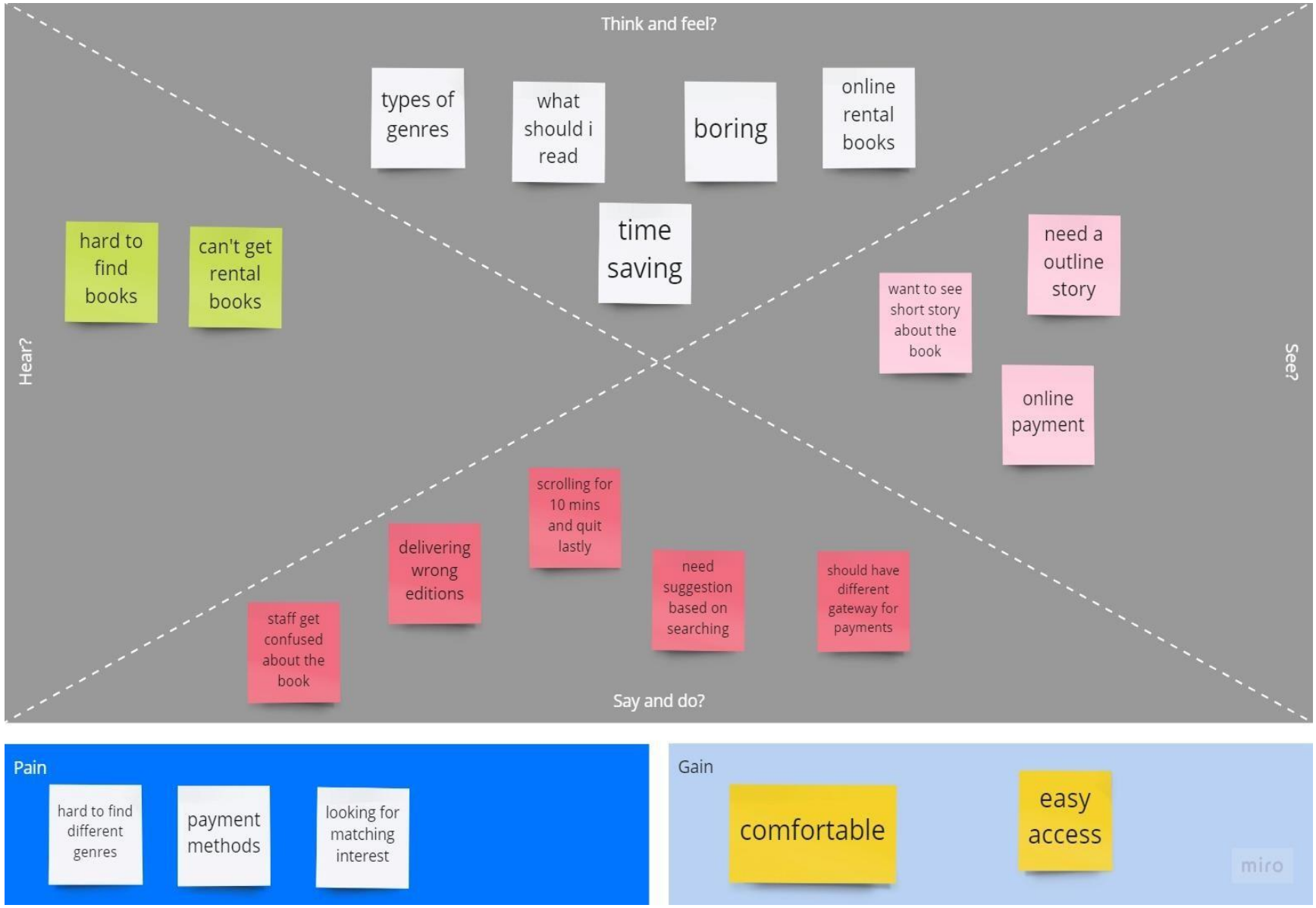
8.1 EMPATHY MAP CAVAS:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Example:

Entrepreneur online book store (buying a books)


Think and feel?
types of genres
what should i read
boring
online rental books
time saving
hard to find books
can't get rental books
need a outline story
want to see short story about the book
online payment
Hear?
See?
scrolling for 10 mins and quit lastly
delivering wrong editions
need suggestion based on searching
should have different gateway for payments
staff get confused about the book
Say and do?
Pain
hard to find different genres
payment methods
looking for matching interest
Gain
comfortable
easy access
miro

8.2 IDEATION AND BRAINSTORMING:

Step-1: Team Gathering, Collaboration and Select the Problem Statement Problem

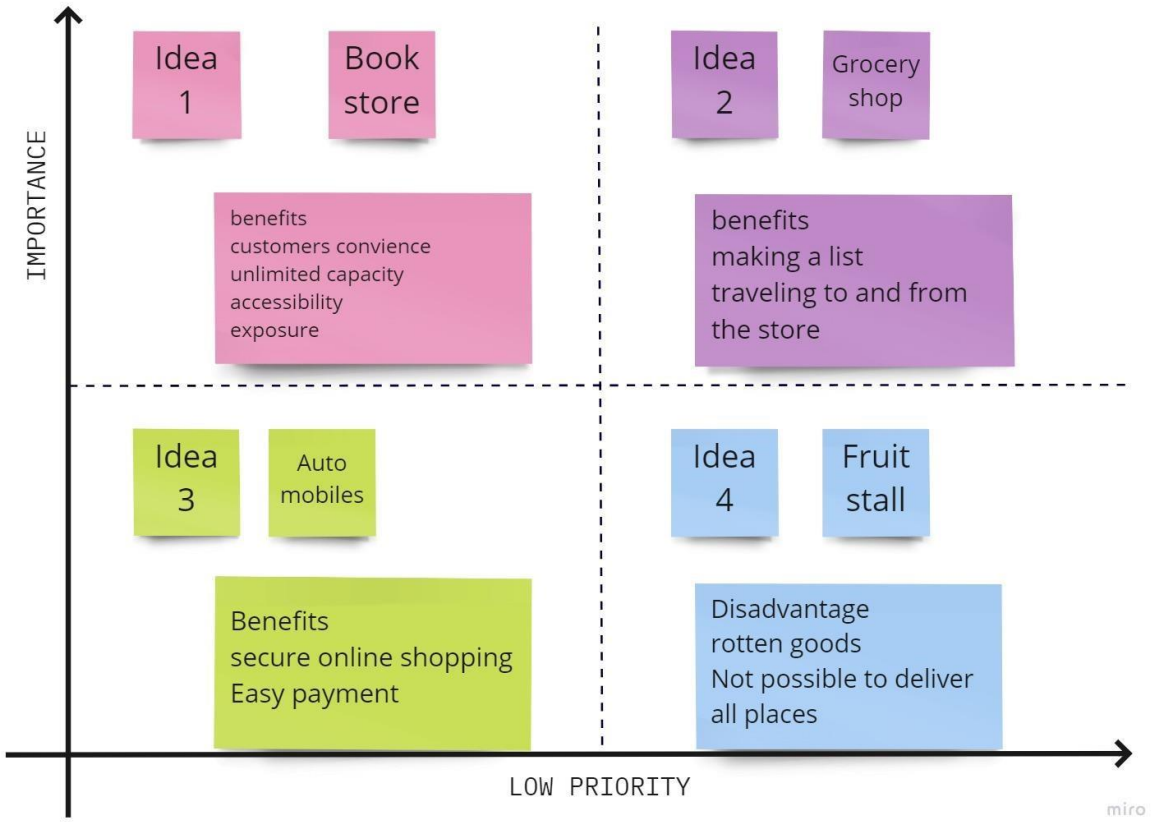
Statement:

The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.

Step-2: Brainstorm, Idea Listing and Grouping


Induja J
Idea 1
Book store
user login
online payment
customer contact
adding stocks
Devisri S
Idea 2
Grocery shop
customer details
offers
online payment
on-time delivery
Gayathri S
Idea 3
Auto mobiles
quality product
on-time delivery
customer reliability
different models
Divya S
Idea 4
Fruits stall
fresh goods
online payment
contact details
miro

Step-3: Idea Prioritization



IDEA PRIORITIZATION:

- customers convenience
- unlimited capacity
- Enhanced customer experience

- Accessibility
- Exposure

3.3 PROPOSED SOLUTION:

I am (Customer)	I'm trying to	But	Because	Which makes me feel

Consumer	Buy a thing	It takes a long time	The website is not responsive and it takes longer time to load	Frustrated
Consumer	Buy a Product	I cannot click the buy button	If I click the button it shows	Frustrated

3.4PROBLEM SOLUTION FIT:

Define CS, fit into CC

1· CUSTOMER SEGMENT(S)

CS

Who is your customer?

Both parents and adults· It mostly used for teenagers· Stiching a trending clothes through a customer Requirements

6· CUSTOMER CONSTRAINTS

What constraints prevent your customers from taking action or limit their choices of solutions?

1· Discounts of clothes
2· Colorful Accessories
3· Quality of Cothes

5· AVAILABLE SOLUTIONS

Which solutions are available to the customers when they face the problem or need to get the job done?

Pros:

1· Easy payment
2· Quality good
3· Fast shipping

Cons:

Explore AS, differenti

Focus on J&P, tap into BE, understand RC

2· JOBS-TO-BE-DONE / PROBLEMS

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides·?

On our end, we mistakenly desire to increase our size· For this, we wish to offer options for body part measurements· The errors can then be fixed·

9· PROBLEM ROOT CAUSE

RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?

The issue with existing apps is that some clothing dimensions are identical and hence susceptible to

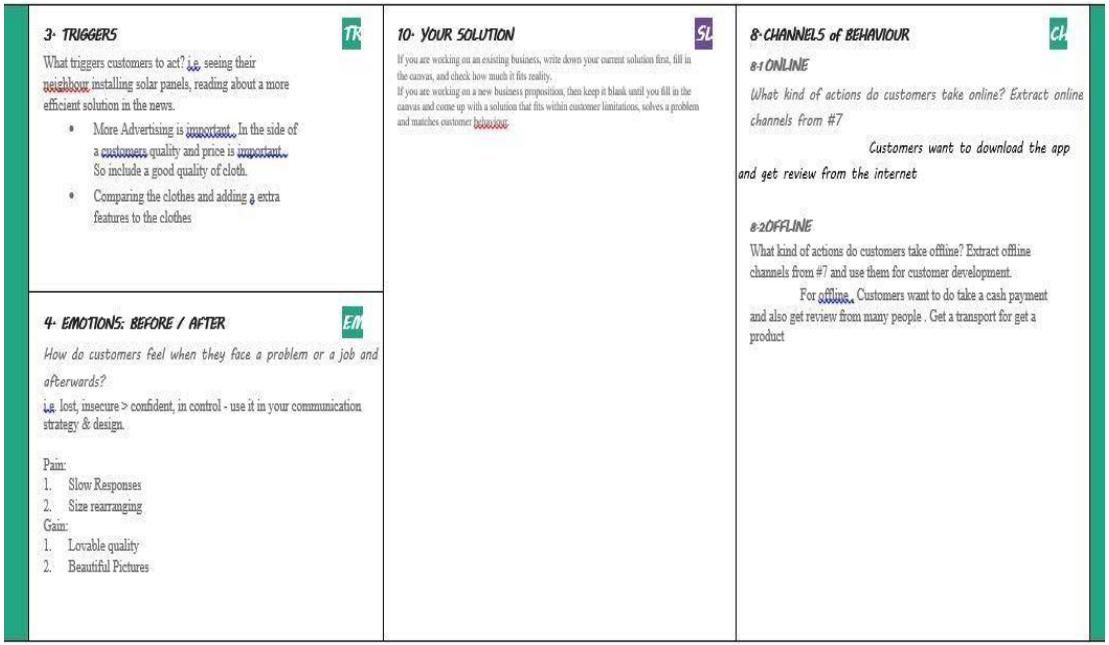
7· BEHAVIOUR

BE

What does your customer do to address the problem and get the job done?
Directly related: □nd the right solar panel installer, calculate usage and bene□ts; indirectly associated: customers spend free time on volunteering work (i·e· Greenpeace)

Customers can compare the pricing to malls or other dress shops, which is directly related· Customers volunteer during their free time, which is indirectly related·

Focus on J&P, tap into BE, understand RC



4.REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENTS:

All the requirements are specified using OCL a software specification language in the second phase of my presentation.

- Capture customer information, store it and make it open at the crucial point in time.
- Present the customers with a constant introduction on the bits of dress status.

- Generate reports exactly and fortunate.
-

4.2 NON-FUNCTIONAL REQUIREMENTS:

- The system has straightforward interfaces.
- This ensures the straight forwardness with which the structure can be learned or used.
- The structure can empower customers to present and work it with for all intents and purposes no arrangement.

.

6. PROJECT PLANNING AND SCHEDULING

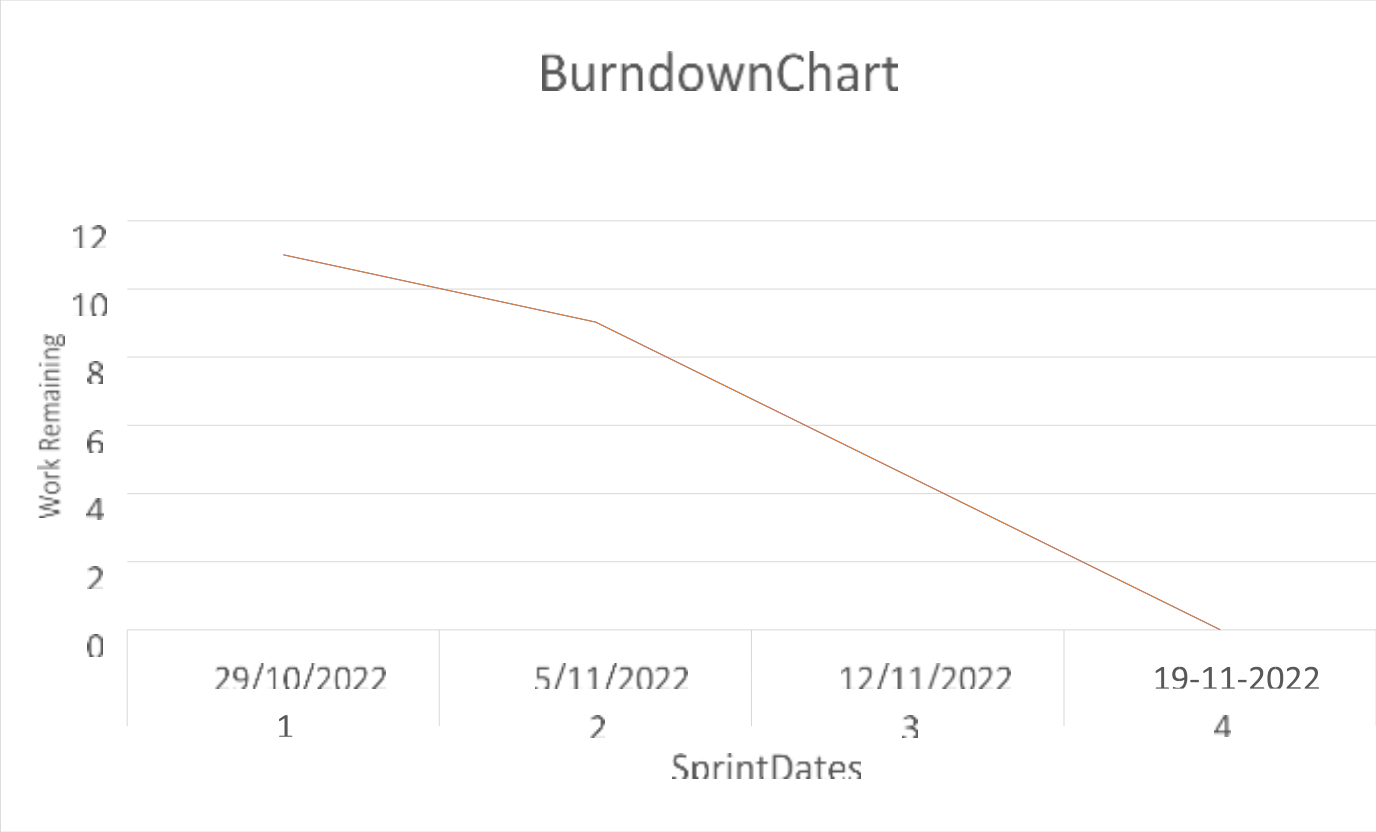
6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	1
Sprint-1	Confirmation mail sending	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	1
Sprint-2	Creating Account	USN-3	As a user, I can register for the application through Facebook	2	Low	2
Sprint-1	Homepage	USN-4	As a user, I can register for the application through Gmail	2	Medium	3

Sprint-1	Login and Dashboard	USN-5	As a user, I can log into the application by entering email & password	1	High	4

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$



6.2 SPRINT DELIVERY AND SCHEDULE

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the most relevant 5 papers have been made and the information gathered have been submitted.	17 SEPTEMBER 2022
Prepare Empathy Map	Prepare an empathy map canvas to capture the user's pains & gains and almost 4 WH questions have been answered and prepare the list of problem statements.	17 SEPTEMBER 2022
Ideation	To list by organizing brainstorming sessions And prioritize the top three ideas based on feasibility and importance.	17 SEPTEMBER 2022

Proposed Solution	To prepare the proposed solution documents, which includes the novelty, feasibility of ideas, business model, social impact, scalability of the solution, etc.	24 SEPTEMBER 2022
Problem Solution Fit	Preparing the problems Solution fit document.	25 SEPTEMBER 2022
Development Phase	Preparing the documentation and source code	13 NOVEMBER 2022

Coding solution:

```
('includes/header.inc.php');

   if ($_SERVER["REQUEST_METHOD"] == "GET"){
      if (isset($_GET["logout"])){
         $_SESSION['User'] = null;
         $_SESSION['UserType'] = null;
         $_SESSION['ID'] = null;
```

```
        header("Location: home.php");
      }
      if (isset($_GET["id"]) && isset($_GET["email"]) && isset($_GET["type"])){
        $id = $_GET["id"];
        $email = $_GET["email"];
        $type = $_GET["type"];


        if ($type == "client" && $client = verifyClient($id, $email)){
          $_SESSION['User'] = $client;
          $_SESSION['UserType'] = "client";
          $_SESSION['ID'] = $client->id;
          header("Location: home.php");
        } else if ($type == "staff" && $staff = verifyStaff($id, $email)){
          $_SESSION['User'] = $staff;
          $_SESSION['UserType'] = "staff";
          $_SESSION['ID'] = $staff->id;
          header("Location: home.php");
        } else {
          echo '<p> Could not verify you sorry! </p>';
        }
      }
    }
?>

<main class="ui segment doubling stackable grid container">

<section class="five wide column">
```

```
<form class="ui form" method="GET">
    <h4 class="ui dividing header">Login</h4>

    <div class="field">
        <label>ID</label>
        <div class="ui mini icon input">
            <input name="id" type="text" placeholder="id...">
        </div>
    </div>

    <div class="field">
        <label>Email</label>
            <div class="ui mini icon input">
                <input name="email" type="text" placeholder="email...">
            </div>
        </div>

    <div class="field">
        <label>Type</label>
        <div class="ui medium icon input">
            <label for="staff_type">staff</label>
            <input id="staff_type" type="radio" name="type" value="staff">

            <label>client</label>
            <input type="radio" name="type" value = "client">
        </div>
    </div>
```

```
        <button class="small ui orange button" type="submit">
            Sign in
        </button>

        <a class="ui icon button" href="register.php">
            sign up
        </a>
    </form>
</section>
</main>

  <footer class="ui black inverted segment">
      <div class="ui container">LIBER Bookstore - /div>
  </footer>
</body>
</html>
[11/16, 8:16 PM] Induja: Home page
[11/16, 8:16 PM] Induja: <?php
    require_once('includes/header.inc.php');

    if ($_SERVER["REQUEST_METHOD"] == "GET"){
        if (isset($_GET["logout"])){
            $_SESSION['User'] = null;
            $_SESSION['UserType'] = null;
            $_SESSION['ID'] = null;
            header("Location: home.php");
        }
        if (isset($_GET["id"]) && isset($_GET["email"]) && isset($_GET["type"])){
```

```
        $id = $_GET["id"];
        $email = $_GET["email"];
        $type = $_GET["type"];


        if ($type == "client" && $client = verifyClient($id, $email)){
          $_SESSION['User'] = $client;
          $_SESSION['UserType'] = "client";
          $_SESSION['ID'] = $client->id;
          header("Location: home.php");
        } else if ($type == "staff" && $staff = verifyStaff($id, $email)){
          $_SESSION['User'] = $staff;
          $_SESSION['UserType'] = "staff";
          $_SESSION['ID'] = $staff->id;
          header("Location: home.php");
        } else {
          echo '<p> Could not verify you sorry! </p>';
        }
      }
    }
?>

<main class="ui segment doubling stackable grid container">

<section class="five wide column">
   <form class="ui form" method="GET">
      <h4 class="ui dividing header">Login</h4>
```

```
<div class="field">
   <label>ID</label>
   <div class="ui mini icon input">
      <input name="id" type="text" placeholder="id...">
   </div>
</div>

<div class="field">
   <label>Email</label>
      <div class="ui mini icon input">
         <input name="email" type="text" placeholder="email...">
      </div>
   </div>

<div class="field">
   <label>Type</label>
   <div class="ui medium icon input">
      <label for="staff_type">staff</label>
      <input id="staff_type" type="radio" name="type" value="staff">

      <label>client</label>
      <input type="radio" name="type" value = "client">
   </div>
</div>

<button class="small ui orange button" type="submit">
   Sign in
</button>
```

```
        <a class="ui icon button" href="register.php">
           sign up
        </a>
    </form>
</section>
</main>

  <footer class="ui black inverted segment">
      <div class="ui container">LIBER Bookstore - </div>
  </footer>
</body>
</html>
[11/16, 8:16 PM] Induja: Manage.php
[11/16, 8:16 PM] Induja: <?php
  require_once('includes/header.inc.php');

  if (!isset($_SESSION['User']) && !($_SESSION['UserType'] == "staff")){
header("Location: login.php");
  }
?>

<main class="ui segment doubling stackable grid container">
<section class="five wide column">
        <form class="ui form" method="GET">
          <h4 class="ui dividing header">Book To Manage</h4>

          <div class="field">
```

```
          <label>ISBN Of Book To Manage/Add</label>
          <div class="ui mini icon input">
             <input name="isbn" type="text" placeholder="isbn...">
             <i class="search icon"></i>
          </div>
        </div>

        <button class="small ui orange button" type="submit">
          <i class="filter icon"></i> Continue
        </button>

      </form>
    </section>
<section class="eleven wide column">
      <h1 class="ui header">Book Information Panel</h1>
      <ul class="ui divided items" id="booksList">


      <?php
  $isbn = "";

        if ($_SERVER["REQUEST_METHOD"] == "GET"){
          if (isset($_GET["isbn"]) && $_GET["isbn"] != ""){
            $isbn = $_GET["isbn"];

  // $books = getBookByISBN($isbn) or die("Failed to connect to the database");
  $books = getBookByISBN($isbn);
```

```
// value is "" for adding, is "disabled" for existing books
$addMode = "disabled";
$panelTitle = "Manage Book With ISBN " . $isbn;
$buttonName = "Save Changes";

// $isbn = "";
$title = "?";
$author_name = "?";
$genre = "?";
$publisher = 0;
$num_pages = 0;
$cost = 0.00;
$price = 0.00;
$publisher_percent = 0.00;
$stock = 0;
$threshold = 0;

if (is_null($books)) {
$addMode = "";
$panelTitle = "Add New Book To Store";
$buttonName = "Add Book";
} else {
$isbn = $books->isbn;
$title = $books->title;
$author_name = $books->author_name;
$genre = $books->genre;
$publisher = $books->publisher;
$num_pages = $books->num_pages;
```

```
  $cost = $books->cost;
  $price = $books->price;
  $publisher_percent = $books->publisher_percent;
  $stock = $books->stock;
  $threshold = $books->threshold;
  }

  echo '
<form class="ui form" method="GET">
<h4 class="ui dividing header">'.$panelTitle.'</h4>
<div class="field">
<label>ISBN</label>
<div class="ui mini icon input">
<input name="isbn" type="text" value='.$isbn.' readonly>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Title</label>
<div class="ui mini icon input">
<input name="title" type="text" value="'.$title.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Author</label>
<div class="ui mini icon input">
<input name="author_name" type="text" value="'.$author_name.'" '.$addMode.'>
```

```
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Genre</label>
<div class="ui mini icon input">
<input name="genre" type="text" value="'.$genre.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Publisher</label>
<div class="ui mini icon input">
<input name="publisher" type="text" value="'.$publisher.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Number of Pages</label>
<div class="ui mini icon input">
<input name="num_pages" type="text" value="'.$num_pages.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Cost</label>
<div class="ui mini icon input">
<input name="cost" type="text" value="'.$cost.'">
```

```
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Price</label>
<div class="ui mini icon input">
<input name="price" type="text" value="'.$price.'">
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Publisher % Cut</label>
<div class="ui mini icon input">
<input name="publisher_percent" type="text" value="'.$publisher_percent.'">
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Stock</label>
<div class="ui mini icon input">
<input name="stock" type="text" value="'.$stock.'">
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Threshold</label>
<div class="ui mini icon input">
<input name="threshold" type="text" value="'.$threshold.'">
```

```
<i class="search icon"></i>
</div>
</div>
<button class="small ui orange button" type="submit">
<i class="filter icon"></i> '.$buttonName.'
</button>
</form>';
if ($addMode == "disabled") {
echo '
<form class="ui form" method="POST">
<div class="field">
<input name="to_delete" type="hidden" value="'.$isbn.'">
</div>
<button class="small ui orange button" type="submit">
<i class="filter icon"></i> Remove Book
</button>
</form>';
}
          }
         }

       </form>
   </section>
<section class="eleven wide column">
       <h1 class="ui header">Book Information Panel</h1>
       <ul class="ui divided items" id="booksList">
```

```
    <?php
$isbn = "";

      if ($_SERVER["REQUEST_METHOD"] == "GET"){
        if (isset($_GET["isbn"]) && $_GET["isbn"] != ""){
          $isbn = $_GET["isbn"];

// $books = getBookByISBN($isbn) or die("Failed to connect to the database");
$books = getBookByISBN($isbn);

// value is "" for adding, is "disabled" for existing books
$addMode = "disabled";
$panelTitle = "Manage Book With ISBN " . $isbn;
$buttonName = "Save Changes";

// $isbn = "";
$title = "?";
$author_name = "?";
$genre = "?";
$publisher = 0;
$num_pages = 0;
$cost = 0.00;
$price = 0.00;
$publisher_percent = 0.00;
$stock = 0;
$threshold = 0;

if (is_null($books)) {
```

```
  $addMode = "";
  $panelTitle = "Add New Book To Store";
  $buttonName = "Add Book";
  } else {
  $isbn = $books->isbn;
  $title = $books->title;
  $author_name = $books->author_name;
  $genre = $books->genre;
  $publisher = $books->publisher;
  $num_pages = $books->num_pages;
  $cost = $books->cost;
  $price = $books->price;
  $publisher_percent = $books->publisher_percent;
  $stock = $books->stock;
  $threshold = $books->threshold;
  }

  echo '
<form class="ui form" method="GET">
<h4 class="ui dividing header">'.$panelTitle.'</h4>
<div class="field">
<label>ISBN</label>
<div class="ui mini icon input">
<input name="isbn" type="text" value='.$isbn.' readonly>
<i class="search icon"></i>
</div>
</div>
<div class="field">
```

```
<label>Title</label>
<div class="ui mini icon input">
<input name="title" type="text" value="'.$title.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Author</label>
<div class="ui mini icon input">
<input name="author_name" type="text" value="'.$author_name.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Genre</label>
<div class="ui mini icon input">
<input name="genre" type="text" value="'.$genre.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Publisher</label>
<div class="ui mini icon input">
<input name="publisher" type="text" value="'.$publisher.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
```

```
<label>Number of Pages</label>
<div class="ui mini icon input">
<input name="num_pages" type="text" value="'.$num_pages.'" '.$addMode.'>
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Cost</label>
<div class="ui mini icon input">
<input name="cost" type="text" value="'.$cost.'">
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Price</label>
<div class="ui mini icon input">
<input name="price" type="text" value="'.$price.'">
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Publisher % Cut</label>
<div class="ui mini icon input">
<input name="publisher_percent" type="text" value="'.$publisher_percent.'">
<i class="search icon"></i>
</div>
</div>
<div class="field">
```

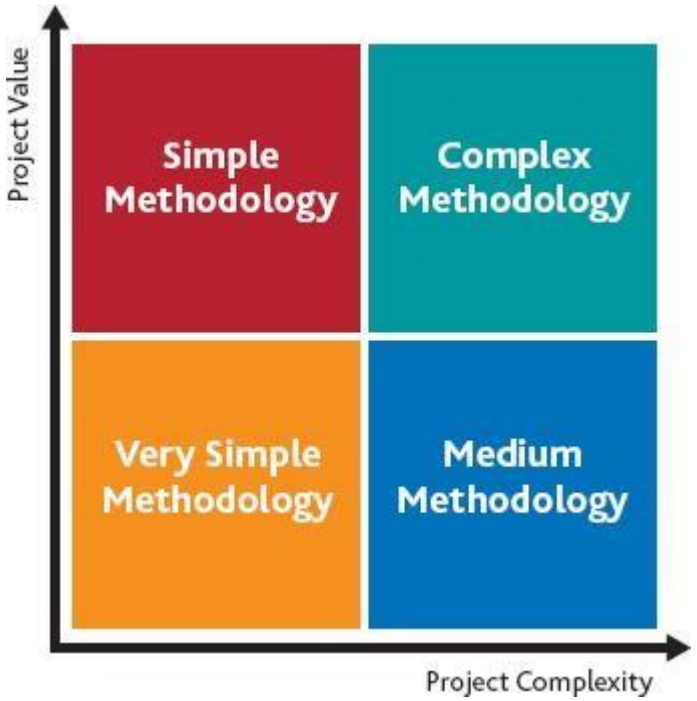
```
<label>Stock</label>
<div class="ui mini icon input">
<input name="stock" type="text" value="'.$stock.'">
<i class="search icon"></i>
</div>
</div>
<div class="field">
<label>Threshold</label>
<div class="ui mini icon input">
<input name="threshold" type="text" value="'.$threshold.'">
<i class="search icon"></i>
</div>
</div>
<button class="small ui orange button" type="submit">
<i class="filter icon"></i> '.$buttonName.'
</button>
</form>';
if ($addMode == "disabled") {
echo '
<form class="ui form" method="POST">
<div class="field">
<input name="to_delete" type="hidden" value="'.$isbn.'">
</div>
<button class="small ui orange button" type="submit">
<i class="filter icon"></i> Remove Book
</button>
</form>';
}
```

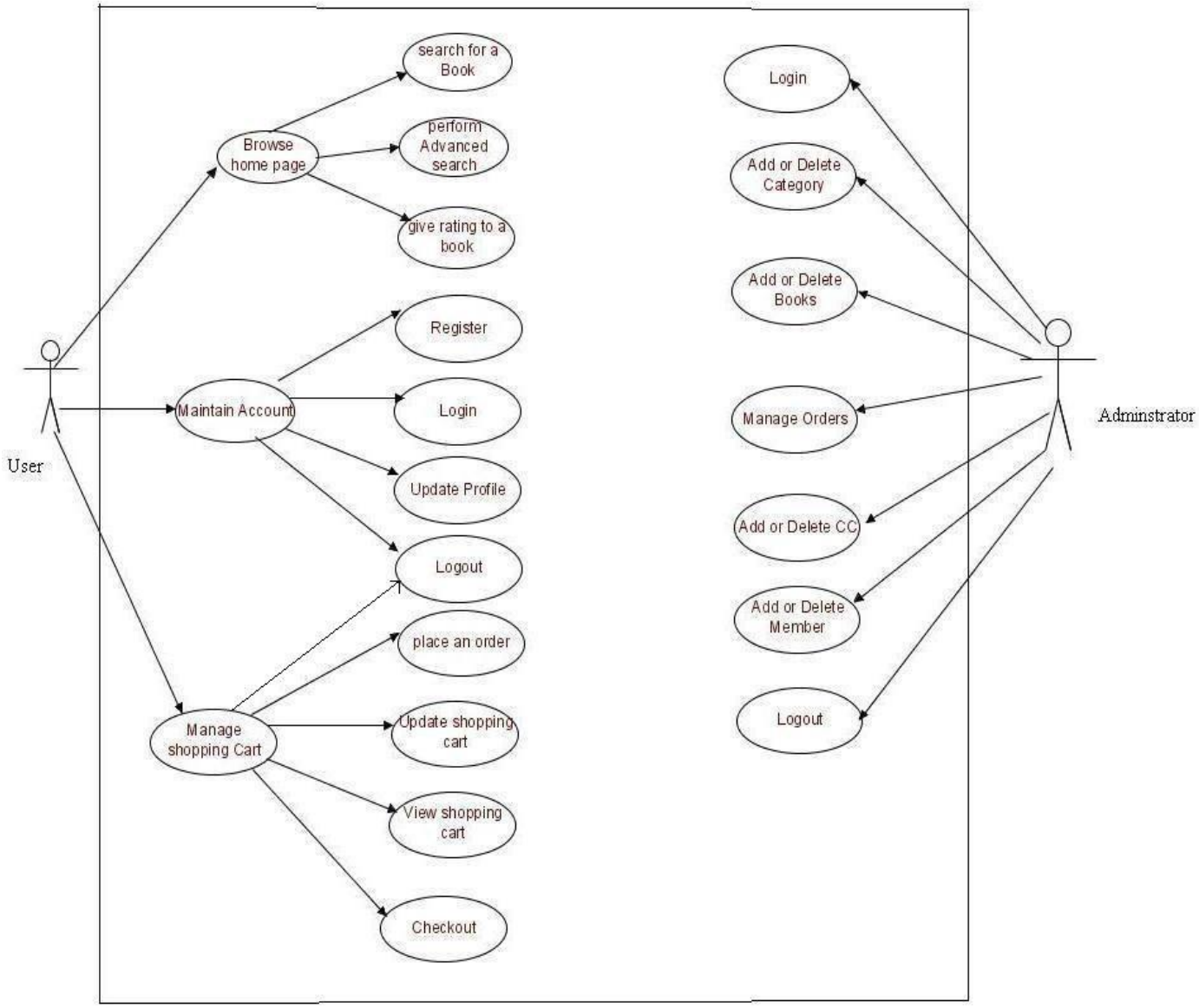
```
    }
  }
```

FEATURE 2

8. Results

Alterations make your garments fit you correctly, providing comfort, ease of movement and above all a flattering finish. Tailoring does not have to be a luxury reserved for only formal and business wear, and we believe there doesn't need to be a special occasion in order for you to want to look and feel special.


Project Value
Simple Methodology
Complex Methodology
Very Simple Methodology
Medium Methodology
Project Complexity


search for a Book
Browse home page
perform Advanced search
give rating to a book
Register
Maintain Account
Login
Update Profile
Logout
place an order
Manage shopping Cart
Update shopping cart
View shopping cart
Checkout
User
Login
Add or Delete Category
Add or Delete Books
Manage Orders
Add or Delete CC
Add or Delete Member
Logout
Adminstrator

10. Advantages and Disadvantages

Advantages:

Alterations make your garments fit you correctly, providing comfort, ease of movement and above all a flattering finish. Tailoring does not have to be a luxury reserved for only formal and business wear, and we believe there doesn't need to be a special occasion in order for you to want to look and feel special.

Disadvantages:

1) **Sizes could be an issue**: Yes, at times you may like a dress, but have to drop it because of the reason that you do not have a size that fits you.
2) The proper fit could be an issue: It often happens that you buy a ready-made garment that you have liked, but unfortunately it does not fit you properly.

11. Conclusion

The system is a general hobby for reading and books on the needs of the consumer customer base, with a certain degree of practical online bookstore system. It is mainly different from the traditional physical bookstores, to overcome a series of physical bookstores limited variety, fixed location, limited space and narrow sales channels and other issues, for people to purchase the book has brought convenience. Online bookstore system not only can easily find the information and purchase books, and the operating conditions are simple, user-friendly, to a large extent to solve real-life problems in the purchase of books, but the design of the system because of the completion time Of the restrictions, some of the functions of the system is not perfect, but the basic functions have been achieved.

Future Scope

The Objectives are

- To read and save the details of book and its author's name.
- To provide an essence of online book store via simple and yet powerful medium.
- View books of all categories.
- Economical, quick and time convenient.

The limitations are:

- Cannot be used without internet.
- Users cannot give feedbacks.
- Limited books are only available.
- Error in spelling by the user cannot provide the exact book they want.

The future scope for our project can be:

- Purchase books online

- Publishers’ insolvency

14. Appendix

model OnlineBookstore

-- Classes

class User attributes UserID: string password: string LoginStatus: string

operations Verifylogin (): Boolean
end

class Administrator < User
attributes AdminID: string password: string Name: string
email: string
phoneNo: integer

operations addCategory(): Boolean deleteCategory:Boolean addMember():Boolean deleteMember():Boolean addBook():Boolean deleteBook():Boolean addCCtype():Boolean

deleteCCtype():Boolean

end

class Customer < User attributes customerID:string password:string Name:string address:string email:string phoneno:integer
CCInfo: string

operations register(): Boolean login(): Boolean
--updateprofile is used to update user information
updateProfile(customerID: string, Name:string, address:string, email:string, phoneno: integer, CCInfo: string):Boolean
--Customer already has an customerID
pre: Customer.allInstances.customerID->includes (customerID)
post: Customer.allInstances.customerID = user.allInstances.customerID@pre
--Customer name has been created
post:
Customer.allInstances.
Name
=user.allInstances->select(C:Customer | C.customerID <>customerID).Name@pre->includes (Name) --Customer address has been created
post: Customer.allInstances.email= Customer.allInstances->select(C:Customer | C.customerID<>customerID).email@pre->includes(email)
post: Customer.allInstances.address = Customer.allInstances->select(C:Customer |

C.customerID<>customerID).address@pre->includes(address) --Customer Phoneno has been created post: Customer.allInstances.phoneno = Customer.allInstances->select(C:Customer |
C.customerID<>customerID).phoneno@pre->includes(phoneno)
post: Customer.allInstances.CCInfo = Customer.allInstances->select(C:Customer |
C.customerID<>customerID). CCInfo @pre->includes(CCInfo)

end

class Category attributes categoryID:integer
categoryName:string

operations
getCategoryBooks(bookID:int):Set(Book) =
Book.allInstances->select(b:Book| b. bookID = bookID) end

class Book attributes bookID:integer categoryID:integer bookName:string authorName:string notes:string price:float imageurl:string producturl:string
rating:int

operations getBook():Boolean end

class ShoppingCart attributes orderID:integer customerID:integer price:float operations addCart():Boolean deleteCart():Boolean updateCart():Boolean

end

class BookOrder attributes orderID:integer customerID:integer price:float quantity:integer operations placeOrder(BO:BookOrder):Boolean
--User authentication is verified
pre: BO.User. Verifylogin (BO.User. UserID, BO.User.password)=true pre: BookOrder.allInstances->excludes(BO)
--Check whether customerid and customerid in Order is same
post: Customer.allInstances -> forAll(C:Customer | C.customerID=BO.customerID implies BO.orderID = C.orderID) post: BookOrder.allInstances.orderID = BookOrder.allInstances.orderID@pre->includes(BO.orderID) end

class Search attributes bookTitle:String categoryID:integer operations getBookset():Boolean end

class BookOrder attributes orderID:integer customerID:integer price:float quantity:integer operations placeOrder(BO:BookOrder):Boolean
--User authentication is verified
pre: BO.User. Verifylogin (BO.User. UserID, BO.User.password)=true pre: BookOrder.allInstances->excludes(BO)

--Check whether customerid and customerid in Order is same
post: Customer.allInstances -> forAll(C:Customer | C.customerID=BO.customerID implies BO.orderID = C.orderID) post: BookOrder.allInstances.orderID = BookOrder.allInstances.orderID@pre->includes(BO.orderID) end

class Search attributes bookTitle:String categoryID:integer
operations getBookset():Boolean end

class AdvSearch attributes bookTitle:String categoryID:integer bookAuthor:String bookLowCost:float bookHighCost:float
operations getBooksetbyAdv():Boolean end

class BookSet attributes bookID:Int bookName:String end

-- Assosiations

--Each book should belong to only one Category
association bookCategory between
Category[1] role subCategory Book[1..*] role allBook
end

--Each BooksOrder should contain atleast one Book association BooksOrderHasBook between
BooksOrder[1] role theBookOrder Book[1..*] role theBook

end

--Each BooksOrder should belong to exactly one Customer
association CustomerHasOrder between Customer[1] role belongstocustomer
BooksOrder[0..*] role thecustomerbook end

--Each Shopping cart should belong to only one Customer association CustomerrelatedtoShoppingCart between
Customer[1] role thecustomer ShoppingCart[0..*] role thecart
end

--Each shoppingcart should have atleast one BooksOrder association ShoppingCartHasOrder between
ShoppingCart[1] role thebookCart BooksOrder [1..*] role theorder
end

--Each search should result some bookset association searchhassomebookset between
Search[1] role thesearch BookSet[0..*] role thesearchset
end

--Each Advsearch should result some bookset association Advsearchhassomebookset between
AdvSearch[1] role theAdvsearch BookSet[0..*] role theAdvsearchset
--Constraints

-- Each user should have different userID context User inv distinctuserID:
User.allInstances -> forAll(user1, user2 |user1 <> user2 implies user1.userID <> user2.userID)

-- Each Book should belong to exactly one Category context Category inv BookHasoneCategory:
Category.allInstances -> forAll(C1,C2 |C1<>C2 && C1. getCategoryBooks -> includes(book) implies C2. getCategoryBooks -> excludes(book))

--Each Book should have a different bookID context Book inv DistinctBookID:
Book.allInstances -> forAll(B1, B2 |B1 <> B2 implies B1.bookID<> B2.bookID)

--The OrderID for each Order must be different
context BookOrder inv DistinctOrderID:
BookOrder.allInstances -> forAll(BO1, BO2 |BO1 <> BO2 implies BO1.orderID<> BO2.orderID)

--Each BookOrder should have some books context BookOrder inv BookOrderHasbooks
self.contains -> notEmpty()

-- Each BookOrder belongs to exactly one customer context BookOrder inv OrdertoOneCustomer
Order.allInstances -> forAll (BO1, BO2 | BO1.orderID <> BO2.orderID

implies BO1.customerID <> BO2.customerID)

-- Quantity should always be a positive value
context BookOrder inv BookOrder Positive
self.quantity > 0
--Each Shopping cart belongs to only one customer context ShoppingCart inv CarthasOneCustomer
ShoppingCart.allInstances -> forAll (SC1, SC2 | SC1.orderID <> SC2.orderID implies SC1.customerID <> SC2.customerID)

--In search Low price should be less than High Price context AdvSearch inv PriceCompare
self. bookLowCost < self. bookHighCos

--