

1. OBJECTIVE:

The primary objective of **BookMart** is to offer a diverse range of books across various genres, including Philosophy, Novels, Self-development and more. This website aims to cater to the diverse reading preferences of customers and ensure a comprehensive collection of titles. Every user of this system uses only a web browser as a client to surf through the system the ensuing site design successfully offers four main services, namely

- ***Ensure a user-friendly browsing and shopping experience:*** BookMart is focused on delivering an intuitive and user-friendly interface, making it easy for customers to browse, search, and find the books they are interested in. Implementing features such as advanced search options, filters, and relevant recommendations enhances the overall shopping experience.
- ***Provide detailed book information:*** This website presents comprehensive details about each book, including author, publication date, synopsis, reviews, and ratings. This information helps customers make informed purchasing decisions and enhances their trust in the website's reliability.
- ***Offer competitive pricing and discounts:*** The objective is to provide competitive pricing for books, ensuring that customers receive good value for their purchases. Additionally, the website can offer periodic discounts, promotions, or bundled deals to attract customers and encourage repeat visits.

SCOPE OF THE PROJECT:

The scope of **BookMart** is selling books can be quite extensive, covering various aspects related to the business and customer experience. Here are some key areas that fall within the scope of an e-commerce website selling books:

- ***Book Catalog Management:*** This website have a robust system for managing and organizing a diverse catalog of books. This includes adding new book titles, updating book information, categorizing books by genre or subject, and ensuring accurate inventory management.
- ***User Registration and Authentication:*** To provide personalized services and enable secure transactions, the website allows users to create accounts, login securely, and manage their profiles. This feature facilitates a customized experience, order history tracking, and the ability to save preferences.
- ***Search and Navigation:*** This website has a user-friendly interface with intuitive search functionality, allowing customers to easily find books based on their preferences. Advanced search options, filters by genre, author, publication date, and other relevant criteria enhance the browsing experience.
- ***Book Details and Reviews:*** Each book listing should include detailed information, such as author, ISBN, publication date, synopsis, reviews, ratings, and other relevant details. This helps customers make informed decisions and increases their confidence in the website's offerings.
- ***Shopping Cart and Checkout:*** This website provides a seamless shopping cart experience, allowing customers to add books to their cart, review their selections, and proceed to a secure checkout process. This includes options for applying discount codes, selecting shipping preferences, and entering payment information.
- ***Order Management and Fulfillment:*** The website should have an efficient system for managing orders. This ensures that customers receive timely updates on their orders and can track their shipments.
- ***Customer Support:*** The website should offer customer support channels, such as email, live chat, or phone, to assist customers with inquiries, order-related issues, or general assistance. Timely and helpful customer support enhances the overall customer experience and satisfaction.

- ***Mobile-Friendly and Responsive Design:*** Given the increasing usage of mobile devices for online shopping, this website has a responsive design that provides a seamless browsing and shopping experience across various screen sizes and devices.

Overall, the scope of **BookMart** is selling books encompasses a wide range of functionalities, including book catalog management, user registration, search and navigation, shopping cart and checkout, order management, customer support, and mobile responsiveness.

2. **BENEFIT:**

- ***Convenience:*** Customers can browse and purchase books from the comfort of their own homes or any location with internet access. They can avoid the hassle of traveling to physical bookstores, saving time and effort.
- ***Wide Selection:*** BookMart have a vast selection of titles across various genres and categories. Customers have access to a much broader range of books compared to brick-and-mortar stores, allowing them to explore different authors, topics, and niche interests.
- ***24/7 Availability:*** The website is accessible round the clock, allowing customers to shop for books at any time that suits them. They are not restricted by the operating hours of physical stores and can make purchases at their convenience.
- ***Detailed Book Information:*** BookMart provide comprehensive details about books, including author information, summaries, reviews, ratings, and related recommendations. This information helps customers make informed decisions and discover new books of interest.
- ***Customer Reviews and Ratings:*** This website allow customers to provide feedback, reviews, and ratings for books they have purchased. This feature helps potential customers gauge the quality and relevance of a book based on the experiences of others.
- ***Expanded Reach:*** BookMart selling books can reach customers beyond the limitations of a physical store's geographical location. It can attract customers from different regions or even international markets, increasing the potential customer base.

3. **CATEGORY OF THE PROJECT:**

BookMart has various categories to organize and classify the books available for purchase. It is a web based project. The specific categories chosen may vary depending on the website's target audience, the range of books offered, and other factors. BookMart is to offer a diverse range of books across various genres, including Philosophy, Novels, Self-development and more.

4. THEORITICAL BACKGROUND:

E-commerce refers to the buying and selling of goods and services over the internet. It involves the use of electronic platforms and technologies to facilitate online transactions. E-commerce has revolutionized the retail industry, providing businesses with the opportunity to reach a global audience and customers with the convenience of shopping from anywhere at any time. UX design is a key consideration in the development of an e-commerce website selling books. It involves designing the website interface and features in a way that enhances user satisfaction, usability, and overall experience. Information architecture focuses on organizing and structuring information within a website to ensure easy navigation and findability. Online retailing, also known as e-tailing, specifically focuses on selling products through online platforms. E-commerce websites selling books fall under the umbrella of online retailing.

5. DEFINATION OF THE PROBLEM:

Traditional bookstores have geographical limitations, making it challenging for customers to access a diverse range of books. Customers may not have convenient access to specific book genres or niche titles, especially if they reside in remote areas or lack nearby bookstores. This limitation restricts customers' ability to explore and purchase a wide selection of books.

Physical bookstores operate within specific hours, requiring customers to visit during those times. This can be inconvenient for individuals with busy schedules, limiting their ability to shop for books at their preferred times. Additionally, customers may face time constraints when searching for specific books across multiple stores, leading to a time-consuming and inefficient shopping process.

The problem of an e-commerce website selling books addresses these limitations by providing a digital platform that offers a wide selection of books, convenient access, comprehensive book information, personalized recommendations, efficient search and browsing capabilities, secure online payment options, and convenient shipping or delivery methods. By addressing these problems, an e-commerce website selling books aims to enhance the overall book purchasing experience for customers and overcome the constraints of traditional bookstores.

6. System Requirement Specification:

Requirement engineering is the first and most important step of software development. It states the customer's needs that the system must satisfy. The outcomes of requirement engineering functional and non-functional requirements— are critical to the success of any software project.

a. Functional Requirements:

Functional requirements state what functions the system should perform. A sample list representing the functionality of the BookMart is given below:

- The website should allow users to create accounts, providing personal information such as name, email address, and password.
- Users should be able to authenticate their identities through login credentials, ensuring secure access to their accounts and personalized features.
- Users should have the ability to manage their accounts, including updating personal information, managing payment methods, tracking orders, and accessing order history.
- The website should have a comprehensive catalog management system to handle book listings. This includes the ability to add new books, update book details (title, author, description, cover image, etc).
- The website should provide intuitive search functionality that allows users to search for books based on various criteria, such as title, author, genre, keywords.
- Each book listing should provide detailed information about the book, including the synopsis, author bio, publication details, ratings, and customer reviews.
- The website should have a shopping cart feature that allows users to add books to their cart for later purchase. Users should be able to review and modify the contents of their cart before proceeding to the checkout process.

- The website should integrate secure online payment systems to enable users to make purchases electronically.
- The website should have an order management system that allows users to track the status of their orders.
- The website should allow users to provide reviews and ratings for books they have purchased. Users should be able to read and evaluate reviews to assist in their decision-making process.
- The website should be mobile-friendly, ensuring a seamless and responsive user experience across different devices, including smartphones and tablets.

b. Non-functional Requirements:

Non-functional requirements for an e-commerce website selling books encompass various aspects of the system's performance, usability, security, and reliability. Here are some key non-functional requirements for an e-commerce website selling books:

- The website should be designed to deliver fast response times and provide a seamless browsing and shopping experience for users.
- It should load pages quickly, display book listings promptly, and process search queries efficiently.
- The website should be scalable to handle increasing user demands and accommodate growth in the number of books, users, and transactions.
- It should be able to handle spikes in traffic during peak periods, such as holiday seasons or promotional events, without affecting performance or causing downtime.
- The website should have an intuitive and user-friendly interface that enables easy navigation, search, and browsing.
- It should be designed to provide a seamless and enjoyable user experience, ensuring that users can easily find books, access relevant information, add items to their cart, and complete the checkout process without confusion or frustration.
- The website should be responsive and compatible with different devices and screen sizes.
- It should adapt to mobile devices, tablets, and desktops, providing a consistent and optimized experience across platforms. The layout and content should adjust dynamically to fit the user's screen without sacrificing usability or readability.
- Security is crucial for an e-commerce website selling books. It should implement strong security measures to protect user data, financial information, and ensure secure online transactions.
- The website should be highly reliable and available to users at all times. It should have robust server infrastructure, data backup mechanisms, and disaster recovery plans in place to minimize downtime and ensure continuous availability of services.
- The website should be compatible with different web browsers (e.g., Chrome, Firefox, Safari, Edge) to ensure a consistent experience for users.
- The website should have regular data backups and a robust backup and recovery strategy to protect against data loss. This ensures that critical information, including

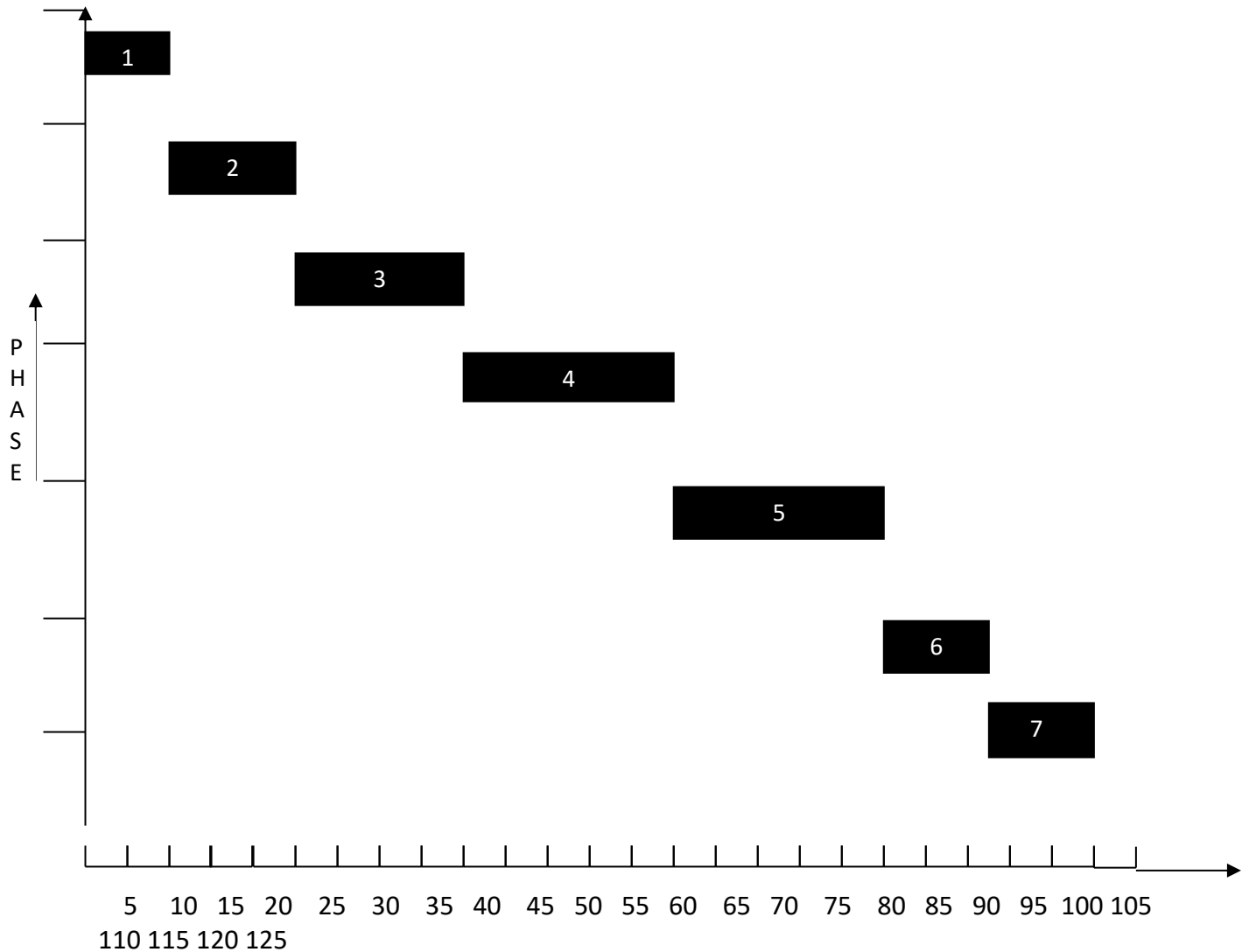
book listings, customer data, and order history, is securely backed up and can be restored in the event of a system failure or data corruption.

- The website should have a dedicated maintenance plan in place to address any issues, bugs, or technical glitches promptly. Regular updates, bug fixes, and security patches should be implemented to keep the website running smoothly and securely.

Addressing these non-functional requirements ensures that the e-commerce website provides a reliable, secure, user-friendly, and high-performance platform for customers to browse, search, and purchase books online.

7. SYSTEM PLANNING (PERT Chart):

-:PERT CHART:-



1. Identification of need
2. Feasibility study
3. Analysis
4. Design
5. Testing
6. Implementation
7. Evaluation

8. Details of Hardware & Software Used for development:

| | |
|------------------|---|
| Computer | Intel or compatible Pentium 4 1.60 MHz or Higher. |
| Memory (RAM) | 2 GB minimum, 4 GB recommended |
| Hard Disk Space | 60 GB and Higher |
| Monitor | VGA or higher resolution 1280x1024 or higher |
| Pointing Device | Microsoft mouse or compatible |
| CD-ROM | Yes |
| Operating System | Windows 8 or later |

VScode:

Visual Studio Code (VS Code) is a popular code editor widely used for web application development. It is a lightweight, cross-platform editor developed by Microsoft that offers a wide range of features and extensions to enhance the development workflow.

Requirements for Visual Studio Code (VS Code):

- Operating System: VS Code is compatible with Windows, macOS, and Linux distributions.
- Hardware: The hardware requirements for running VS Code are minimal, and it can run on machines with lower specifications.
- Dependencies: VS Code requires certain dependencies, such as Node.js and Electron, to be installed on the system.

Key Features of Visual Studio Code (VS Code):

- Lightweight and Fast: VS Code is designed to be lightweight and fast, providing a smooth and responsive coding experience.
- Cross-Platform Compatibility: VS Code is available for Windows, macOS, and Linux, allowing developers to work seamlessly across different operating systems.
- Intuitive User Interface: The user interface of VS Code is clean and intuitive, providing a clutter-free coding environment. The sidebar, status bar, and activity bar provide easy access to various functionalities.

- **IntelliSense and Code Completion:** VS Code offers intelligent code completion and suggestions (IntelliSense) for multiple programming languages, making coding faster and reducing errors.
- **Built-in Git Integration:** VS Code has built-in Git integration, enabling developers to manage version control directly within the editor. It supports common Git operations, such as commit, push, pull, and merge.
- **Integrated Terminal:** VS Code provides an integrated terminal within the editor, allowing developers to run commands, execute scripts, and perform various tasks without leaving the editor.
- **Extensions Marketplace:** VS Code has a rich ecosystem of extensions available in the marketplace. These extensions provide additional functionality and customization options, ranging from language support to debugging tools and productivity enhancements.
- **Debugger:** VS Code supports debugging for multiple programming languages, providing breakpoints, variable inspection, and step-by-step execution to help developers identify and fix issues.
- **Task Runner:** VS Code includes a task runner that allows developers to define and execute custom tasks, automating build processes, running tests, and performing other routine tasks.
- **Integrated Terminal:** VS Code provides an integrated terminal within the editor, allowing developers to execute commands, run scripts, and perform various tasks without leaving the editor interface.
- **Snippets:** VS Code supports code snippets, allowing developers to quickly insert commonly used code snippets or templates with predefined structures.
- **Theming and Customization:** VS Code allows users to customize the editor's appearance, including themes, fonts, icons, and layout, to suit their preferences and create a personalized coding environment.
- **Language Support and Extensions:** VS Code provides extensive language support out-of-the-box, and developers can further enhance it by installing language-specific extensions from the marketplace.
- **Live Share:** VS Code includes the Live Share extension, enabling developers to collaborate and code together in real-time, regardless of their physical locations.

- **Multiple File Format Support:** VS Code supports a wide range of file formats, including text files, source code files, configuration files, markdown files, and more.

These features make Visual Studio Code a versatile and powerful code editor, widely used by developers for various programming tasks, including web development, software development, scripting, and more.

How to Use:

- **Install VS Code:** Download and install the appropriate version of VS Code for your operating system from the official website (<https://code.visualstudio.com/>). Follow the installation instructions specific to your platform.
- **Open VS Code:** After installation, launch VS Code from your applications or Start menu.
- **Set up Preferences:** Upon opening VS Code, you can configure various preferences to customize your editor. You can access preferences by clicking on "File" in the top menu and selecting "Preferences" or by using the keyboard shortcut (Ctrl + , on Windows/Linux or Command + , on macOS).
- **Explore the User Interface:** Familiarize yourself with the VS Code user interface, which consists of the editor window, sidebar, status bar, and activity bar. The sidebar contains icons for file explorer, source control, extensions, and other functionalities. The status bar displays information about the active file, Git status, and more. The activity bar provides quick access to various features and extensions.
- **Open a Folder or File:** To work on a project, you can either open a folder or directly open a single file by clicking on "File" in the top menu and selecting "Open Folder" or "Open File". Alternatively, you can use the "Open Folder" or "Open File" options in the welcome screen.
- **Edit Code:** Start writing or editing your code in the editor window. VS Code supports syntax highlighting, code formatting, and other editing features to enhance your coding experience.
- **Use Extensions:** VS Code has a vast marketplace of extensions that can enhance the editor's functionality and support specific programming languages, frameworks, and tools. You can explore and install extensions by clicking on the square icon in the sidebar or by going to the Extensions view (Ctrl + Shift + X on Windows/Linux or Command + Shift + X on macOS).

- **Customize Themes and Settings:** VS Code allows you to customize the editor's appearance and behavior. You can change themes, adjust settings, and install custom extensions to personalize your coding environment. Access the settings through the "File" menu or by using the keyboard shortcut mentioned earlier.
- **Utilize Productivity Features:** VS Code offers various productivity features like IntelliSense, code snippets, integrated terminal, and debugging capabilities. Take advantage of these features to write code more efficiently and debug your applications.
- **Collaborate with Others:** VS Code includes the Live Share extension, which enables real-time collaboration with other developers. You can share your workspace, edit code together, and communicate via chat or voice.
- **Save and Run Code:** Save your code files using Ctrl + S (Windows/Linux) or Command + S (macOS). To run code, you can either use built-in terminal commands or configure custom tasks to automate build and execution processes.
- **Explore Documentation and Community:** VS Code has comprehensive documentation available on their website, providing detailed information about various features and functionalities. Additionally, the VS Code community is active, with forums, tutorials, and online resources available for support and learning.

9. Explanation of web technologies:

a. HTML:

HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is a **markup** language
- A markup language is a set of markup **tags**
- The tags **describe** document content
- HTML documents contain **HTML tags** and plain **text**
- HTML documents are also called **web pages**

HTML Tags:

HTML markup tags are usually called HTML tags

- HTML tags are keywords (tag names) surrounded by **angle brackets** like `<html>`
- HTML tags normally **come in pairs** like `` and ``
- The first tag in a pair is the **start tag**, the second tag is the **end tag**
- The end tag is written like the start tag, with a **forward slash** before the tag name
- Start and end tags are also called **opening tags** and **closing tags**

`<tagname>content</tagname>`

HTML Elements:

"HTML tags" and "HTML elements" are often used to describe the same thing. But strictly speaking, an HTML element is everything between the start tag and the end tag, including the tags.

HTML Element: `<p>This is a paragraph. </p>`

HTML Versions:

Since the early days of the web, there have been many versions of HTML:

| Version | Year |
|-----------|------|
| HTML | 1991 |
| HTML+ | 1993 |
| HTML 2.0 | 1995 |
| HTML 3.2 | 1997 |
| HTML 4.01 | 1999 |
| XHTML 1.0 | 2000 |
| HTML5 | 2012 |
| XHTML5 | 2013 |

HTML5:

HTML5 will be the new standard for HTML. The previous version of HTML, HTML 4.01, came in 1999. The web has changed a lot since then.

HTML5 is still a work in progress. However, the major browsers support many of the new HTML5 elements and APIs.

- New Elements
- New Attributes
- Full CSS3 Support
- Video and Audio
- 2D/3D Graphics
- Local Storage
- Local SQL Database
- Web Applications

HTML5 - New Features:

Some of the most interesting new features in HTML5:

- The <canvas> element for 2D drawing
- The <video> and <audio> elements for media playback
- Support for local storage
- New content-specific elements, like <article>, <footer>, <header>, <nav>, <section>
- New form controls, like calendar, date, time, email, url, search

Browser Support for HTML5:

HTML5 is not yet an official standard, and no browsers have full HTML5 support. But all major browsers (Safari, Chrome, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions.

How HTML5 Get Started:

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

Some rules for HTML5 were established:

- New features should be based on HTML, CSS, DOM, and JavaScript
- Reduce the need for external plugins (like Flash)
- Better error handling
- More markup to replace scripting
- HTML5 should be device independent
- The development process should be visible to the public

b. CSS:

- CSS stands for Cascading Style Sheets
- Styles define **how to display** HTML elements
- Styles were added to HTML 5.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

Styles Solved a Big Problem

HTML was never intended to contain tags for formatting a document. HTML was intended to define the content of a document, like:

`<h1>This is a heading</h1>`

`<p>This is a paragraph. </p>`

When tags like ``, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process. To solve this problem, the World Wide Web Consortium (W3C) created CSS. In HTML 5.0, all formatting could be removed from the HTML document, and stored in a separate CSS file. All browsers support CSS today. CSS Saves a Lot of Work! CSS defines HOW HTML elements are to be displayed. Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file!

CSS 3:

CSS is used to control the style and layout of Web pages. CSS3 is the latest standard for CSS. CSS3 is completely backwards compatible, so you will not have to change existing designs. CSS stands for Cascading Style Sheets. Style Sheets are used to create the formatting and style, such as background color and layout, for markup languages like HTML and XHTML. CSS3 is the latest upcoming standard for Cascading Style Sheets. It builds on CSS2, the current standard, and adds several new features. Many of these new features allow designers to apply advanced styles and designs to elements in a webpage using just style sheets. In the current standard, for example, an element's borders have square edges. Designers who want to give their Webpages a softer feel with rounded borders have to resort to a tedious process of splicing and inserting images of rounded borders. Now with CSS3, the designer can create a rounded border by simply specifying it with the border property.

As of this writing CSS3 is still in development. To make the process easier, the entire standard has been divided into modules, which pertain to specific design and style features. For example, there are modules for color, selectors, fonts, template layout, animations etc.

CSS3 Modules:

CSS3 is split up into "modules". The old specification has been split into smaller pieces, and new ones are also added. Some of the most important CSS3 modules are:

- Selectors
- Box Model
- Backgrounds and Borders
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

c. *JavaScript:*

JavaScript is a high-level, interpreted programming language primarily used for front-end web development. It allows developers to add interactivity, dynamic behavior, and functionality to web pages. JavaScript is also increasingly used for server-side development (Node.js) and mobile app development (React Native).

Vanilla JavaScript refers to the use of pure, standard JavaScript without any additional frameworks or libraries. It implies writing code using the native capabilities and features provided by the JavaScript language itself, without relying on third-party tools or dependencies.

- **Core JavaScript:** Vanilla JavaScript encompasses the fundamental building blocks of the JavaScript language, including variables, data types, operators, control structures (such as if-else statements and loops), functions, objects, and arrays. It allows developers to leverage the full power and flexibility of JavaScript without any abstractions or modifications.
- **Browser APIs:** Vanilla JavaScript interacts directly with the browser's Document Object Model (DOM) and other browser-specific APIs to manipulate and control web page elements, handle events, make HTTP requests, perform client-side validation, and perform other web-related tasks. This includes methods and properties provided by the window object, document object, XMLHttpRequest object, and more.
- **Cross-Browser Compatibility:** Vanilla JavaScript is designed to work consistently across different web browsers, ensuring compatibility and consistent behavior without relying on browser-specific or framework-specific features.
- **Lightweight and Performance:** Since Vanilla JavaScript does not depend on external frameworks or libraries, it typically has a smaller code footprint, leading to faster load times and improved performance. It allows developers to optimize code and fine-tune performance based on specific project requirements.
- **Learning the Core Language:** Using Vanilla JavaScript helps developers gain a deeper understanding of the language and its underlying concepts. It encourages developers to master the core language features, best practices, and design patterns, making it easier to transition to other frameworks or libraries if needed.
- **Flexibility and Customization:** Vanilla JavaScript provides developers with the flexibility to tailor their code and design patterns to meet the specific needs of their

projects. Without the constraints imposed by frameworks, developers have full control over their codebase and can implement custom solutions and optimizations.

- **Community Support:** Vanilla JavaScript has a vast and active community of developers who share knowledge, provide tutorials, write libraries, and offer support. This community helps developers learn, solve problems, and stay updated with the latest advancements in the JavaScript ecosystem.

Version Information: JavaScript has evolved over the years, with various versions released. Here are some notable versions:

- **ECMAScript 5 (ES5):** This version, released in 2009, introduced significant improvements to the language, including strict mode, JSON support, and additional array methods like `forEach`, `map`, and `filter`.
- **ECMAScript 6 (ES6) / ECMAScript 2015:** Released in 2015, ES6 introduced many new features and syntax enhancements to JavaScript. Some notable features include arrow functions, classes, template literals, destructuring assignments, modules, and promises.
- **ECMAScript 2016 and Beyond:** Subsequent versions of ECMAScript have introduced additional features and improvements. Some notable features include the spread operator, `async/await` for handling asynchronous operations, destructuring assignments, default function parameters, and more.

d. Table Designed:

Description of LocalStorage in JavaScript: LocalStorage is a web storage feature provided by modern web browsers, allowing web applications to store data locally on the user's device. It provides a simple key-value storage mechanism that persists even when the browser is closed and reopened. The data stored in LocalStorage remains available to the web application within the same domain.

LocalStorage Methods:

| Method | Description |
|---------------------|---|
| setItem(key, value) | Stores a key-value pair in LocalStorage. The key and value can be any string. |
| getItem(key) | Retrieves the value associated with the specified key from LocalStorage. Returns null if the key doesn't exist. |
| removeItem(key) | Removes the key-value pair associated with the specified key from LocalStorage. |
| clear() | Removes all key-value pairs from LocalStorage. |
| length | Returns the total number of key-value pairs stored in LocalStorage. |
| key(index) | Returns the key at the specified index in LocalStorage. |
| localStorage | Represents the LocalStorage object that provides the methods and properties for interacting with the local storage. |

Example:

// Storing data in LocalStorage

```
localStorage.setItem("name", "John");
```

```
localStorage.setItem("age", "25");
```

```
localStorage.setItem("city", "New York");
```

// Retrieving data from LocalStorage

```
let name = localStorage.getItem("name");
```

```
let age = localStorage.getItem("age");  
  
let city = localStorage.getItem("city");  
  
// Removing a key-value pair from LocalStorage  
  
localStorage.removeItem("age");  
  
// Clearing all data from LocalStorage  
  
localStorage.clear();
```

In this example, we use the `localStorage` object to store and retrieve key-value pairs. The `setItem()` method is used to store data, while the `getItem()` method retrieves the value associated with a specific key. The `removeItem()` method removes a key-value pair, and the `clear()` method clears all data from `LocalStorage`.

10. Hardware and Software Requirement:

a. Hardware Requirements:

| | |
|-----------------|---|
| Computer | Intel or compatible Pentium 4 1.60 MHz or Higher. |
| Memory (RAM) | 2 GB minimum, 4 GB recommended |
| Hard Disk Space | 60 GB and Higher |
| Monitor | VGA or higher resolution 1280x1024 or higher |
| Pointing Device | Microsoft mouse or compatible |
| CD-ROM | Yes |

b. Software Requirements:

HTML, CSS, Vanilla JS.

O.S – Windows, Linux, Mac.

11. Implementation of security mechanisms at various levels:

Implementing security mechanisms at various levels of the website is crucial to protect user data, secure transactions, and prevent unauthorized access. Here are some key areas where security measures should be implemented:

- **User Authentication and Authorization:** **Strong Password Policies:** Enforce password complexity requirements and encourage users to choose strong passwords. **Account Lockouts:** Implement mechanisms to lock user accounts after a certain number of unsuccessful login attempts to prevent brute-force attacks.
- **Secure Payment Processing:** **Payment Gateway Integration:** Use reputable payment gateways that adhere to industry security standards for secure payment processing. **Address Verification System (AVS):** Implement AVS to verify the address provided during payment with the billing address on file for the cardholder. **Card Verification Value (CVV):** Collect and validate the CVV code on the payment card to add an extra layer of authentication during transactions.
- **Data Protection:** Only collect and store the necessary user data. Avoid storing sensitive information that is not required for the functioning of the website. **Regular Data Backups:** Implement regular backup procedures to ensure data availability and recovery in case of system failures or data breaches.
- **Secure Coding Practices:** **Input Validation:** Implement input validation to prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS) attacks. **Secure Session Management:** Implement secure session handling techniques to protect against session hijacking and fixation attacks.
- **Security Monitoring and Logging:** **Log Analysis:** Implement logging mechanisms to capture and analyze system logs for detecting suspicious activities or security incidents.

It is important to note that security is an ongoing process, and regular audits, risk assessments, and updates to security measures should be conducted to ensure the continued protection of the website and its users' data.

12. System Design:

Software design is a creative task. During the design phase, a software developer concludes how to implement a software system to meet customers' needs, and a good design should exhibit high cohesion and low coupling. Many design methods have been put forward to achieve this goal. Several case tools have matured to the point that they can help developers efficiently walk through the design process.

a. DFD (Data Flow Diagram):

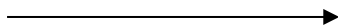
Data models have been used in the project to realize the problem domain and solution procedures visually. Data flow diagram techniques have been depicted to identify the solution process and the way to materialize it.

Data flow diagram that demonstrates the flow of various data and information through the system. It also enables the identification of the processes that converts one set of data to another set of information. DFD is helpful to identify the entities external to the system and the data stores that should be included in the project.

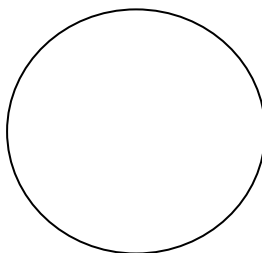
DFD Symbols:



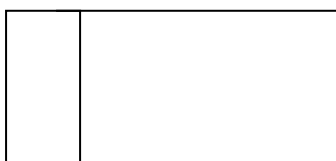
Rectangle represents the source or destination of system data also called an external entity.



Arrow represents Data flow.

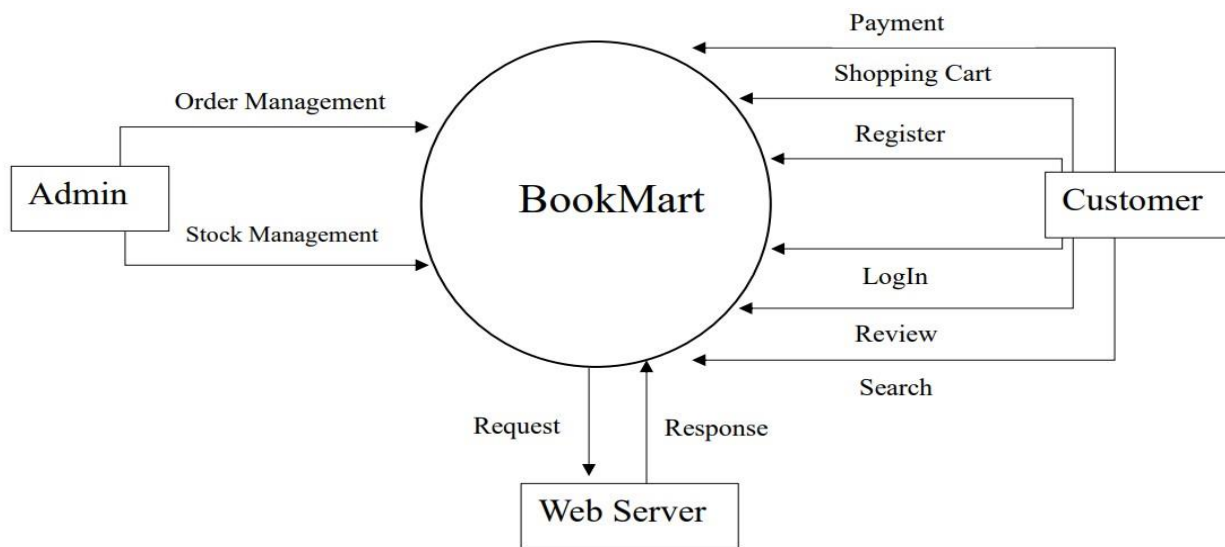


Circle or bobble represents a process that transform data from one from to another by performing some tasks with the data call process.

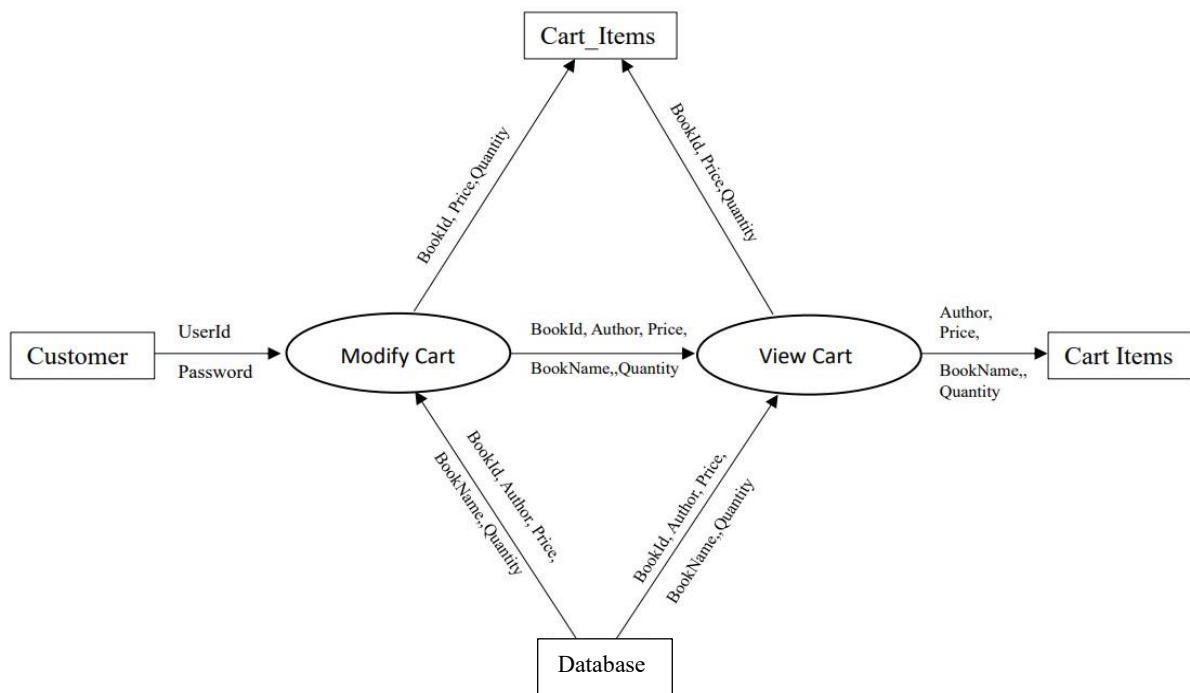


An open rectangle is data store.

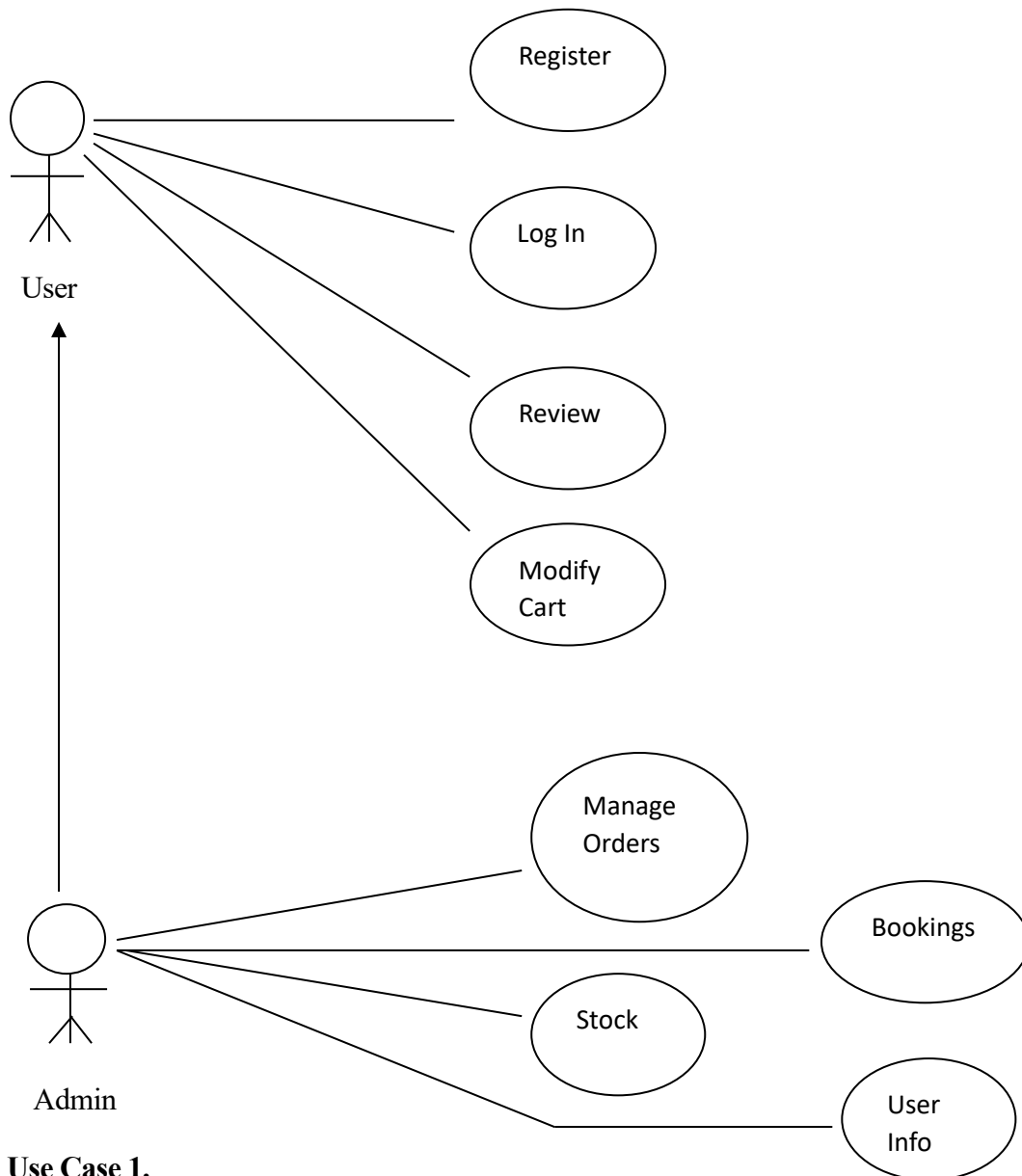
0 Level



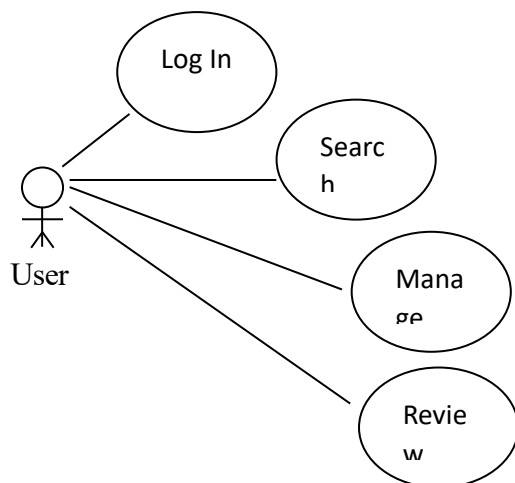
1Level



System design Architecture



Use Case 1.



b. System Characteristics:

System Characteristics of BookMart include:

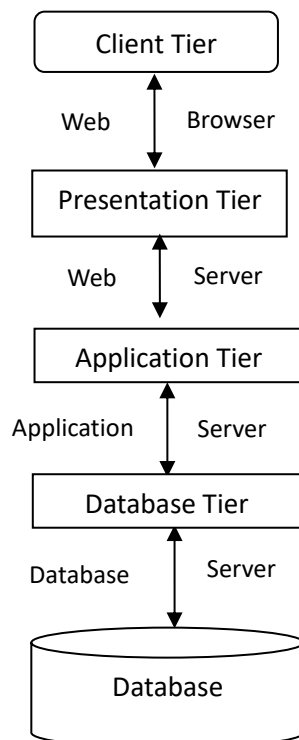
- **Scalability:** The system should be able to handle a growing number of users, products, and transactions without a significant decrease in performance. It should be scalable to accommodate increased traffic and data volume.
- **Reliability:** The website should be reliable, ensuring that it is available and accessible to users at all times. It should have mechanisms in place to handle potential failures, minimize downtime, and recover quickly from any disruptions.
- **Security:** The system should implement robust security measures to protect user data, prevent unauthorized access, and ensure secure payment transactions. This includes encryption of sensitive information, user authentication mechanisms, and adherence to security best practices.
- **Performance:** The website should deliver fast response times and smooth user experiences. It should be optimized to minimize page load times, handle concurrent user requests efficiently, and provide a seamless browsing and purchasing experience.
- **Usability:** The system should be user-friendly and intuitive, allowing users to easily navigate the website, search for books, view product details, and complete purchases. Clear and well-designed user interfaces should be implemented to enhance usability.
- **Mobile Responsiveness:** As mobile devices are widely used for online shopping, the website should be responsive and optimized for different screen sizes and devices. It should provide a consistent and user-friendly experience across desktops, tablets, and mobile phones.
- **Integration:** The system should be capable of integrating with external services and APIs. This includes payment gateways, shipping providers, inventory management systems, and other third-party services to streamline operations and provide a seamless end-to-end experience.
- **Analytics and Reporting:** The system should have built-in analytics and reporting capabilities to track website performance, user behavior, sales metrics, and other relevant data. This helps in making data-driven decisions and improving the overall effectiveness of the e-commerce website.

- **Support and Maintenance:** The system should have provisions for ongoing support and maintenance, including bug fixes, software updates, and enhancements. Timely support should be available to address any issues and ensure smooth operations.
- **Compliance:** The e-commerce website should comply with relevant legal and industry regulations, such as data protection and privacy laws, payment card industry (PCI) standards, and consumer protection guidelines.

These system characteristics are crucial for a book selling e-commerce website to provide a secure, reliable, and user-friendly platform that meets customer expectations and drives successful business operations.

C. System Architecture:

The system architecture of BookMart is typically follows a multi-tiered architecture, separating different components and responsibilities.



Explanation of the System Architecture:

- **Client Tier:** This tier represents the client-side components, which include the web browser or mobile application used by the website users to interact with the system.

- **Presentation Tier:** The presentation tier consists of the web server that handles incoming HTTP requests from clients and serves the appropriate responses. It can also include a load balancer for distributing requests across multiple web servers.
- **Application Tier:** The application tier contains the application server, which processes business logic, handles user authentication, manages sessions, and interacts with the database. It performs tasks such as managing the product catalog, processing orders, and handling user-related operations.
- **Database Tier:** The database tier consists of the database server that stores and manages the website's data. It can be a relational database management system (RDBMS) or a NoSQL database. It stores information related to products, orders, customer details, and other relevant data.

The system architecture diagram outlines the high-level components and their interactions in a book selling e-commerce website. It illustrates the flow of data and requests between the client, presentation, application, and database tiers, as well as the integration with external services.

d. Entity Relationship Diagram (ERD):

An Entity Relationship Diagram (ERD) is a network model that describes the stored data layout of the system. ERD are used because the data structures and relationships in a system may be so complex that one may want to highlight them and examine them independently of the processing that takes places in the system. The ERD was originally proposed by Peter Chen for the design of the relational database systems and has been extended by others. The primary purpose of the ERD is to represent the data objects and their relationships.

Important terms in E-R Modeling:-

Mapping Cardinalities:

Mapping Cardinalities Express the number of entities to which other entity can associate with each other via a relationship set. It can take following forms.

One to One (1:1):-

An in a associated with at most one entity in B and an entity in B is associated with at most one entity in A, where A and B are two different entity sets.

One to Many (1: N):-

An entity in A is associated with many number of entities in B, An entity in B however, can be associated with almost one entity in A.

Many to One (N: 1):-

An entity in A is associated with almost one entity in B. An entity in B, however can be associated with any number of entities in A.

Many to Many (M: N):-

An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

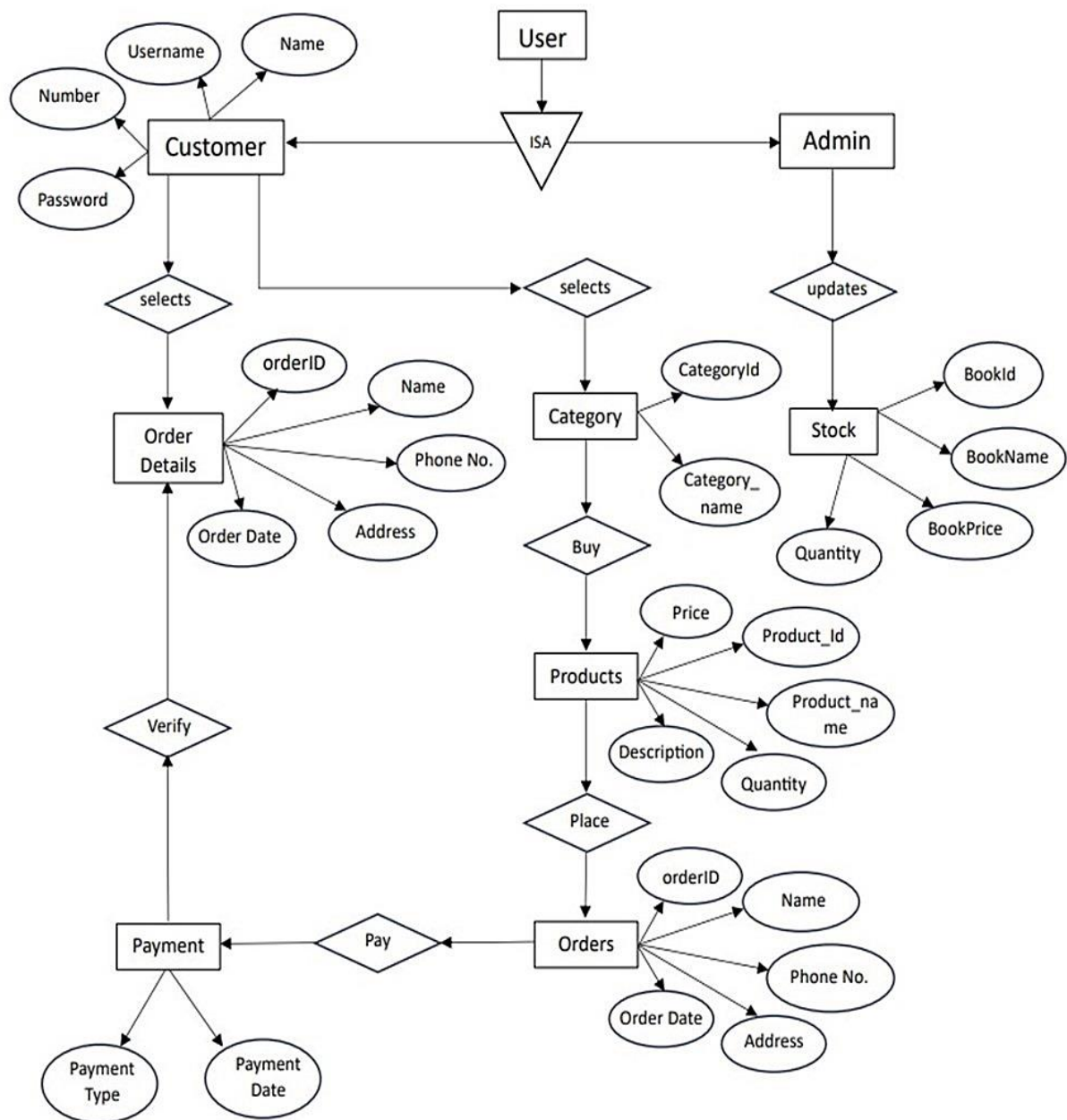


Figure: *Structure Schema with Details about the Entity Attributes*

e. Tables Design:

As a core of information exchange and processing, database plays an important role in information systems. It forms the foundation of web services and web-based systems. A very large number of computer applications are database related, and almost every web-based application uses databases to store information. A good database facilitates almost every aspect of an information management system.

Our BookMart's database is constructed in local Storage of web browser which store various data. Furthermore, a number of stored procedures, which encapsulate the operations on database tables, are used. Table 1 and Table 2 shows the main database tables used in BookMart.

Table 1

User Table:

| Field Name | Description |
|-------------------|---|
| regUsername | Stores the user name of user at the time of registration. |
| Name | Stores the full name of user. |
| num | Stores the mobile Number of user. |
| password | Stores the password of the user. |

Table 2

Order Table:

| Field Name | Description |
|------------|-----------------------------------|
| fname | Stores the name of user. |
| doo | Stores the date of order. |
| num | Stores the mobile number of user. |
| email | Stores the email of the user. |
| address | Stores the address of the user |
| orderid | Stores the order ID. |
| price | Stores the total order price. |

f. Identifying The Basic Relationships:

The entity User is central to the schema, being connected through several relationships to almost all the other entities. In particular:

1. User-Product Relationship:

- Users can browse and view products on the website.
- Users can add products to their shopping cart.
- Users can make purchases and place orders for products.
- Users can leave reviews and ratings for products.

2. User-Order Relationship:

- Users can create orders to purchase products.

- Users can view the status and details of their orders.

3. User-Review Relationship:

- Users can leave reviews and ratings for products.
- Users can view and edit their own reviews.

4. User-Address Relationship:

- Users can store and manage their shipping addresses.
- Users can add, edit, and delete their address information.

g. Software Engineering Process Involved:

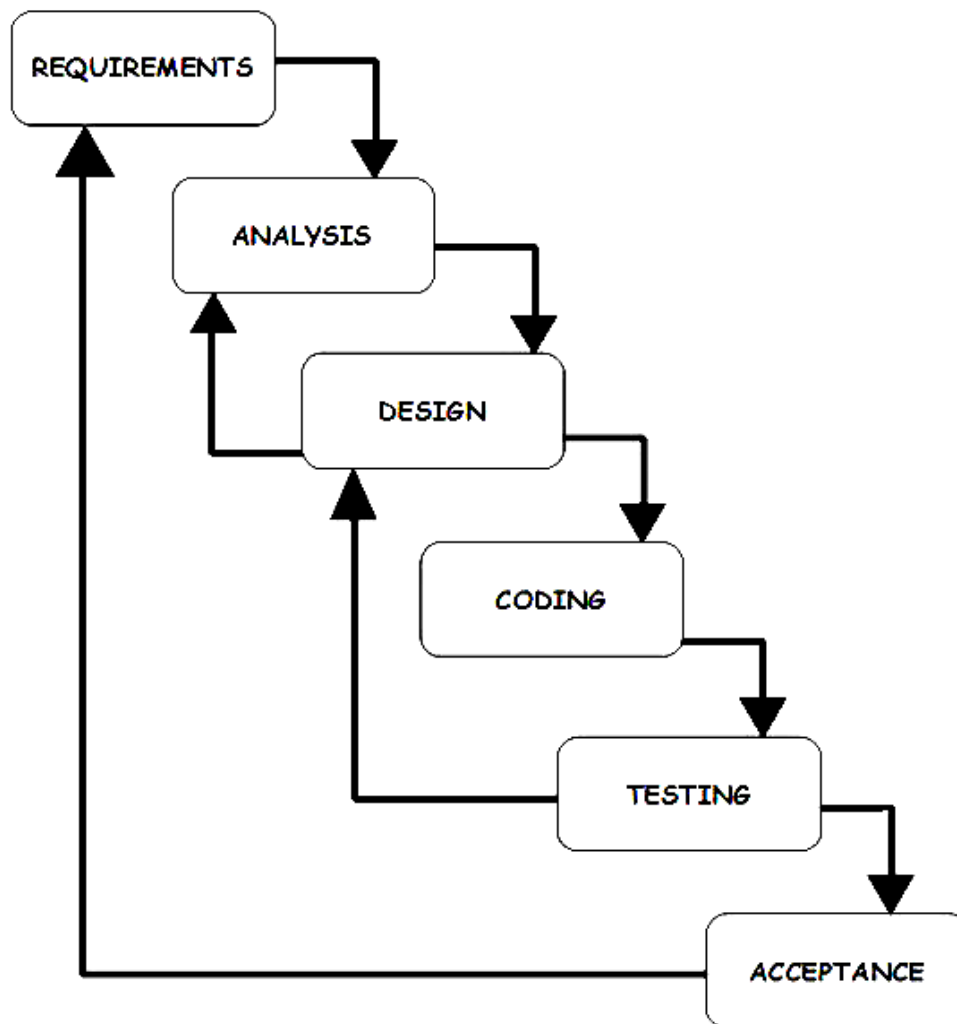


Fig: Process Model

- ***Requirements Specification & Analysis:***

A **Software requirements specification** (SRS), a requirements specification for a software system, is a complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

The software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.

- **System design:**

System design is a process of problem solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

- **Implementation:**

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy.

- **Testing:**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- meets the requirements that guided its design and development,
- works as expected,
- can be implemented with the same characteristics,
- and satisfies the needs of stakeholders.

- **Acceptance:**

The software full fill the SRS the it is accepted by customer that process called Acceptance

h. CODE OF PROJECT:

HomePage:

```
import books from "../DataBase/Data.js";

// new books section
const newBooksCarouselItem1 = document.getElementById("newBooksCarouselItem1");
const newBooksCarouselItem2 = document.getElementById("newBooksCarouselItem2");
const newBooksCarouselItem3 = document.getElementById("newBooksCarouselItem3");
const newestBookFiltred1 = books().filter(book => book.date <= 4);
const newestBookFiltred2 = books().filter(book => book.date <= 8 && book.date > 4);
const newestBookFiltred3 = books().filter(book => book.date <= 12 && book.date > 8);
function cartCreator(book) {
  return `<div class="col-lg-3 col-sm-6">
    <div class="thumb-wrapper">
      <div class="img-box">
        
      </div>
      <div class="thumb-content">
        <h4 class="book-name">${book.name}</h4>
        ${book.stars}
        <p class="item-price"><strike>${book.lastPrice} ||
      ""</strike><b>₹${book.price}</b></p>
        <button class="btn btn-outline-primary"> Add to Cart <i class="bi bi-bag"></i></button>
      </div>
    </div>`
}
newestBookFiltred1.forEach(book => {
  newBooksCarouselItem1.innerHTML += (cartCreator(book));
})
newestBookFiltred2.forEach(book => {
  newBooksCarouselItem2.innerHTML += (cartCreator(book));
})
```

```

newestBookFiltred3.forEach(book => {
  newBooksCarouselItem3.innerHTML += (cartCreator(book));
})
// best books section
const bestBooksCarouselItem1 = document.getElementById("bestBooksCarouselItem1");
const bestBooksCarouselItem2 = document.getElementById("bestBooksCarouselItem2");
const bestBooksCarouselItem3 = document.getElementById("bestBooksCarouselItem3");
const sortedBooksPerRate = books().sort(function (a, b) {
  let rateA = a.rating,
  rateB = b.rating;
  return rateB - rateA;
});
const slicedSortedBooksPerRate1 = sortedBooksPerRate.slice(0, 4);
const slicedSortedBooksPerRate2 = sortedBooksPerRate.slice(4, 8);
const slicedSortedBooksPerRate3 = sortedBooksPerRate.slice(8, 12);
slicedSortedBooksPerRate1.forEach(book => {
  bestBooksCarouselItem1.innerHTML += (cartCreator(book));
})
slicedSortedBooksPerRate2.forEach(book => {
  bestBooksCarouselItem2.innerHTML += (cartCreator(book));
})
slicedSortedBooksPerRate3.forEach(book => {
  bestBooksCarouselItem3.innerHTML += (cartCreator(book));
})
// best deals section
const bestDealsCarouselItem1 = document.getElementById("bestDealsCarouselItem1");
const bestDealsCarouselItem2 = document.getElementById("bestDealsCarouselItem2");
const bestDealsCarouselItem3 = document.getElementById("bestDealsCarouselItem3");
const dealedBooks = books().filter(book => book.deal);
dealedBooks.slice(0,4).forEach(book => {
  bestDealsCarouselItem1.innerHTML += (cartCreator(book));
})
dealedBooks.slice(4,8).forEach(book => {
  bestDealsCarouselItem2.innerHTML += (cartCreator(book));
})

```



```

    })
    dealedBooks.slice(8,12).forEach(book => {
        bestDealsCarouselItem3.innerHTML += (cartCreator(book));
    })
    // best of philosophy section
    const bestOfPhilosophyCarouselItem1 =
    document.getElementById("bestOfPhilosophyCarouselItem1");
    const bestOfPhilosophyCarouselItem2 =
    document.getElementById("bestOfPhilosophyCarouselItem2");
    const philosophyBooks = books().filter(book => book.categorie.includes("philosophy"));
    philosophyBooks.slice(0,4).forEach(book => {
        bestOfPhilosophyCarouselItem1.innerHTML += (cartCreator(book));
    })
    philosophyBooks.slice(4,8).forEach(book => {
        bestOfPhilosophyCarouselItem2.innerHTML += (cartCreator(book));
    })
    // best of self development section
    const bestSelfDevCarouselItem1 = document.getElementById("bestSelfDevCarouselItem1");
    const bestSelfDevCarouselItem2 = document.getElementById("bestSelfDevCarouselItem2");
    const bestSelfDevCarouselItem3 = document.getElementById("bestSelfDevCarouselItem3");
    const selfDevBooks = books().filter(book => book.categorie.includes("self-dev"));
    selfDevBooks.slice(0,3).forEach(book => {
        bestSelfDevCarouselItem1.innerHTML += (cartCreator(book));
    })
    selfDevBooks.slice(3,6).forEach(book => {
        bestSelfDevCarouselItem2.innerHTML += (cartCreator(book));
    })
    selfDevBooks.slice(6,9).forEach(book => {
        bestSelfDevCarouselItem3.innerHTML += (cartCreator(book));
    })
    // best of snovels section
    const bestNovelsCarouselItem1 = document.getElementById("bestNovelsCarouselItem1");
    const bestNovelsCarouselItem2 = document.getElementById("bestNovelsCarouselItem2");
    const bestNovelsCarouselItem3 = document.getElementById("bestNovelsCarouselItem3");

```

```

const novels = books().filter(book => book.categorie.includes("novel"));
const sortednovelsPerRate = novels.sort(function (a, b) {
    let rateA = a.rating,
        rateB = b.rating;
    return rateB - rateA;
});
sortednovelsPerRate.slice(0,4).forEach(book => {
    bestNovelsCarouselItem1.innerHTML += (cartCreator(book));
})
sortednovelsPerRate.slice(4,8).forEach(book => {
    bestNovelsCarouselItem2.innerHTML += (cartCreator(book));
})
sortednovelsPerRate.slice(8,12).forEach(book => {
    bestNovelsCarouselItem3.innerHTML += (cartCreator(book));
})
// username session
var user =sessionStorage.getItem("username");
// alert(user);
if ( user !== null)
{
    showSession(user);
    function showSession(user) {
        // alert("in func");
        sessionUsername.textContent = "Welcome"+" " +user+"!!";
    }
}

```

Registration Form:

```

//Registration event
document.addEventListener("DOMContentLoaded", function() {
    var registrationForm = document.getElementById("registrationForm");
    registrationForm.addEventListener("submit", function(event) {
        event.preventDefault();
        //alert("in click");
    }

```

```

var regUsername = document.getElementById("regUsername").value;
//alert("before userdata");
//alert("out userdata");
if (localStorage.getItem(regUsername)) {
    //alert("in storage");
    alert("Username already taken!");
} else {
    //alert("storing");
    // Store username and password in localStorage
    var userData= {
        regUsername:
        regUsername,
        Name:
        document.getElementById("name").value,
        num:
        document.getElementById("num").value,
        password:
        document.getElementById("password").value
    }
    var store=JSON.stringify(userData);
    localStorage.setItem(regUsername, store );
    alert("Registration successful! Please log in.");
    window.location = "login.html";
}
});
});

```

Log In Form:

```

function verify() {
    var user = document.getElementById("loginUsername").value;
    var pass = document.getElementById("loginPassword").value;
    var userRef = JSON.parse(localStorage.getItem(user)).regUsername;
    var passRef = JSON.parse(localStorage.getItem(user)).password;
    if (user == userRef && pass == passRef) {

```

```

    sessionStorage.setItem("username", user);
    window.open("../Home-Page/index1.html");
  } else {
    alert("It's seem you make a mistake...");
  }
}

```

Cart Page:

```

const productItemsList = document.getElementById("productItemsList");
let parsedData = JSON.parse(localStorage.getItem("data")) || [];
if (parsedData) {
  parsedData.forEach((product, number) => {
    const cartCard = `<tr>
      <td scope="row">
        <div class="d-flex align-items-center">
          
          <div class="flex-column ms-4">
            <p class="mb-2 d-none d-sm-inline-block">${product.name}</p>
          </div>
        </div>
      </td>
      </td>
      <td id="myAdderTd" class="align-middle">
        <button id="adderBtn" type="button" class="btn border-0 ms-4 mt-1 btn-outline-primary
btn-sm" data-mdb-toggle="tooltip"
        title="Duplicate item">
          <i class="bi bi-plus-lg"></i>
        </button>
      </td>
      <td class="align-middle">

```

```

<strike class="text-secondary">${product.lastprice || ""}</strike>
<p class="mb-0" style="font-weight: 500">₹${product.price}</p>
</td>
<td id="myTd" class="align-middle">
  <button id="deleterBtn" type="button" class="btn ms-4 mt-1 btn-outline-primary btn-
sm" data-mdb-toggle="tooltip"
  title="Remove item">
    <i class="bi bi-trash"></i>
<div class="d-none">${product.value}</div>
  </button>
</td>
</tr>
`;
productItemsList.innerHTML += cartCard;
const deleterBtn = document.querySelectorAll("#deleterBtn");
deleterBtn.forEach((el) => {
  el.addEventListener("click", () => {
    const elementValue =
      el.parentElement.parentElement.children[3].children[0].children[1]
        .innerHTML;
    const filteredElementByValue = parsedData.filter(
      (item) => item.value !== elementValue
    );
    localStorage.setItem("data", JSON.stringify(filteredElementByValue));
    location.reload();
  });
});
const adderBtn = document.querySelectorAll("#adderBtn");
adderBtn.forEach((el) => {
  el.addEventListener("click", () => {
    let data = {
      cover:
        el.parentElement.parentElement.children[0].children[0].children[0]
          .src,

```

```

name: el.parentElement.parentElement.children[0].children[0]
  .children[1].innerText,
price:
  el.parentElement.parentElement.children[2].children[1].innerText.slice(
    1
  ),
value: Math.random() * 10e60,
lastprice: (() => {
  if (el.parentElement.parentElement.innerText) {
    return el.parentElement.parentElement.children[2].children[0]
      .innerText;
  }
  })(),
};
console.log(data.value);
parsedData.push(data);
localStorage.setItem("data", JSON.stringify(parsedData));
location.reload();
});
});
});
}
// proceed to pay modal section
const modalBody = document.querySelector(".modal-body");
let totalPrice = 0;
parsedData.forEach((product) => {
  const productBody1 = `

---



46


```

```

        alt="Book"
    />
    <div class="flex-column ms-4">
        <p class="mb-2 d-none d-sm-inline fs-10">${product.name}</p>
    </div>
</div>
</div>
<div class="align-middle">
</div>
<div class="align-middle">
<strike class="text-secondary">${product.lastprice || ""}</strike>
<p class="mb-0" style="font-weight: 500">₹${product.price}</p>
</div>
</div>`;
modalBody.innerHTML += productBody1;
totalPrice += Math.round(Number(product.price) * 100) / 100;
});
const modalBody2 = document.querySelector(".modal-body-2");
function modalBody2Creator() {
    totalPrice = Math.round(totalPrice * 100) / 100;
    var user =sessionStorage.getItem("username");
    // console.log(user);
    var b =JSON.parse(localStorage.getItem(user));
    var temp={price:totalPrice};
    let temp1=Object.assign(b,temp);
    localStorage.setItem(user,JSON.stringify(temp1));
    const productBody2 = ` <div class="mb-5">
<div class="form-floating">
    <input type="number" class="form-control border border-primaryborder-opacity-25"
id="form3Examplea2" placeholder="v">
    <label class="form-label" for="form3Examplea2">Enter your discount code (for e.g. Enter a
number(1,70))</label>
</div>
</div>

```

```

<div class="d-flex justify-content-between">
<h5 class="text-uppercase">Total price : </h5>
<h5>₹${totalPrice}</h5>
</div>`;
    modalBody2.innerHTML = productBody2;
  }
  modalBody2Creator();
  // discount code process
  const formControlDiscount = document.querySelector(".form-control");
  formControlDiscount.addEventListener("keydown", function (event) {
    if (event.key === "Enter") {
      const discountNum = formControlDiscount.value;
      if (discountNum <= 70 && discountNum >= 1) {
        setTimeout(() => {
          totalPrice -= (totalPrice * discountNum) / 100;
          modalBody2Creator();
        }, 500);
      }
    }
  });

```

Book Data:

```

export default function books() {
  return [
    {
      cover:
        "https://images-na.ssl-images-
amazon.com/images/I/61tqfa+xbWL._AC_UL226_SR226,226_.jpg",
      name: "Verity",
      price: " 799",
      date: 1,
      deal: false,
      categorie: ["novel", "philosophy"],
      gone:false,

```



```

rating: 3.5,
stars: `

49


```

```

    price: "899",
    date: 3,
    deal: false,
    categorie: ["novel"],
    gone:false,
    rating: 4.5,
    stars: `

50


```

```

{
  cover:
    "https://images-na.ssl-images-
amazon.com/images/I/617uZq23IPL._AC_UL226_SR226,226_.jpg",
    name: "Reminders of Him: A Novel",
    price: "449",
    date: 6,
    deal: false,
    categorie: ["novel", "philosophy"],
    gone:false,

    rating: 4.5,
    stars: `

51


```

```

        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        </div>`,
    },
    {
        cover:
"https://images-na.ssl-images-
amazon.com/images/I/81O1oy0y9eL._AC_UL226_SR226,226_.jpg",
        name: "Where the Crawdads Sing",
        price: "499",
        date: 8,
        deal: false,
        categorie: ["novel"],
        gone:false,
        rating: 5,
        stars: `<div class="ratings">
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            </div>`,
    },
    {
        cover:
"https://images-na.ssl-images-
amazon.com/images/I/81wgcl4wxL._AC_UL226_SR226,226_.jpg",
        name: "Atomic Habits",
        price: "799",
        date: 9,
        deal: false,
        categorie: ["self-dev"],
        gone:false,

```

```

rating: 5,
stars: `<div class="ratings">
  <i class="bi bi-star-fill text-warning"></i>
  <i class="bi bi-star-fill text-warning"></i>
  <i class="bi bi-star-fill text-warning"></i>
  <i class="bi bi-star-fill text-warning"></i>
  <i class="bi bi-star-fill text-warning"></i>
</div>`,
},
{
  cover:
"https://images-na.ssl-images-
amazon.com/images/I/71cTtxm0p0L._AC_UL226_SR226,226_.jpg",
  name: "Solito: A Memoir",
  price: "1889",
  date: 10,
  deal: false,
  categorie: ["novel"],
  gone:false,
  rating: 4.5,
  stars: `<div class="ratings">
    <i class="bi bi-star-fill text-warning"></i>
    <i class="bi bi-star-fill text-warning"></i>
    <i class="bi bi-star-fill text-warning"></i>
    <i class="bi bi-star-fill text-warning"></i>
    <i class="bi bi-star-half text-warning"></i>
  </div>`,
},
{
  cover:
"https://images-na.ssl-images-
amazon.com/images/I/61xkvfPVupL._AC_UL226_SR226,226_.jpg",
  name: "November 9: A Novel",
  price: "499",

```

```

date: 11,
deal: false,
categorie: ["novel"],
gone:false,
rating: 4.5,
stars: `

54


```

```

"https://images-na.ssl-images-amazon.com/images/I/91cqEsyjd-
L._AC_UL226_SR226,226_.jpg",
  name: "The Very Hungry Caterpillar",
  price: "399",
  date: 13,
  deal: false,
  categorie: ["novel"],
  gone:false,
  rating: 5,
  stars: `

55


```

```

        </div>`,
    },
    {
        cover:
"https://images-na.ssl-images-
amazon.com/images/I/61uiYWcEQGL._AC_UL226_SR226,226_.jpg",
        name: "The Return of the Gods",
        price: "4014",
        date: 15,
        deal: false,
        categorie: ["novel"],
        gone:false,
        rating: 4.5,
        stars: `<div class="ratings">
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-half text-warning"></i>
        </div>`,
    },
    {
        cover:
"https://images-na.ssl-images-
amazon.com/images/I/71aG+xDKSYL._AC_UL226_SR226,226_.jpg",
        name: "The 48 Laws of Power",
        price: "799",
        date: 16,
        deal: false,
        categorie: ["self-dev", "philosophy"],
        gone:false,
        rating: 4.5,
        stars: `<div class="ratings">
            <i class="bi bi-star-fill text-warning"></i>

```



```

        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-half text-warning"></i>
    </div>`,
},

{
    cover:
    "https://images-na.ssl-images-
amazon.com/images/I/81823bTjKHL._AC_UL226_SR226,226_.jpg",
    name: "The Last Thing He Told Me: A Novel",
    price: "599",
    date: 30,
    deal: false,
    categorie: ["novel"],
    gone:false,
    rating: 4.5,
    stars: `<div class="ratings">
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-half text-warning"></i>
    </div>`,
},

{
    cover:
    "https://m.media-amazon.com/images/I/51U0RLLU7vS._AC_UF226,226_FMjpg_.jpg",
    name: "The Art of War",
    price: "220",
    date: 31,
    deal: true,
    lastPrice: "₹250",

```

```

categorie: ["novel", "philosophy"],
gone:false,
rating: 4.5,
stars: `

58


```

```

price: "358",
date: 33,
deal: true,
lastPrice: "₹550",
categorie: ["self-dev"],
gone:false,
rating: 4.5,
stars: `

59


```

```

{
  cover:
    "https://m.media-amazon.com/images/I/41S21miKn1S._AC_UF226,226_FMjpg_.jpg",
  name: "Close Your Eyes, Sleep",
  price: "599",
  date: 35,
  deal: true,
  lastPrice: "₹1099",
  categorie: ["novel"],
  gone:false,
  rating: 3.5,
  stars: `

60


```

```

        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        <i class="bi bi-star-fill text-warning"></i>
        </div>`,
    },
    {
        cover:
        "https://images-na.ssl-images-amazon.com/images/I/41ovZVriMRL._AC_SR226,226_.jpg",
        name: "Marx for Beginners",
        price: "1223",
        date: 37,
        deal: true,
        lastPrice: "₹1407",
        categorie: ["self-dev", "philosophy"],
        gone: false,
        rating: 4.5,
        stars: `<div class="ratings">
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-fill text-warning"></i>
            <i class="bi bi-star-half text-warning"></i>
        </div>`,
    },

```

Order Page:

```

var user =sessionStorage.getItem("username");
// console.log(user);
var key=JSON.parse(localStorage.getItem(user)).num;
var userData=JSON.parse(localStorage.getItem(key));
var tableData=document.querySelector("#table-data")
tableData.innerHTML="";
userData.forEach((data,index) => {
    console.log(index);

```

```

tableData.innerHTML+=`
<tr>
  <td>${data.orderid}</td>
  <td>${data.Fname}</td>
  <td>${data.address}</td>
  <td>${data.num}</td>
  <td>${data.email}</td>
  <td>${data.doo}</td>
  <td>${data.price}</td>
</tr>
<!-- and so on... -->
`;
});

```

Payment Page:

```

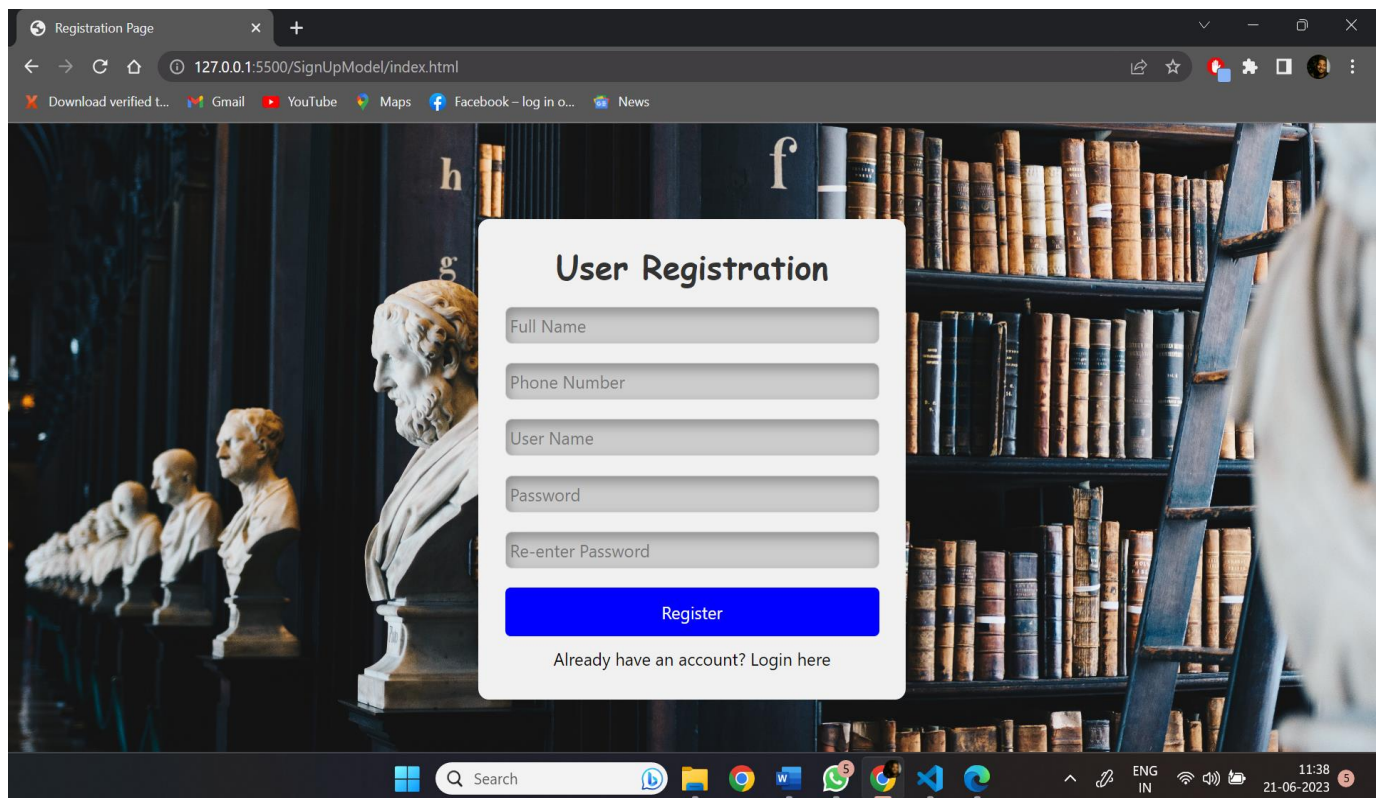
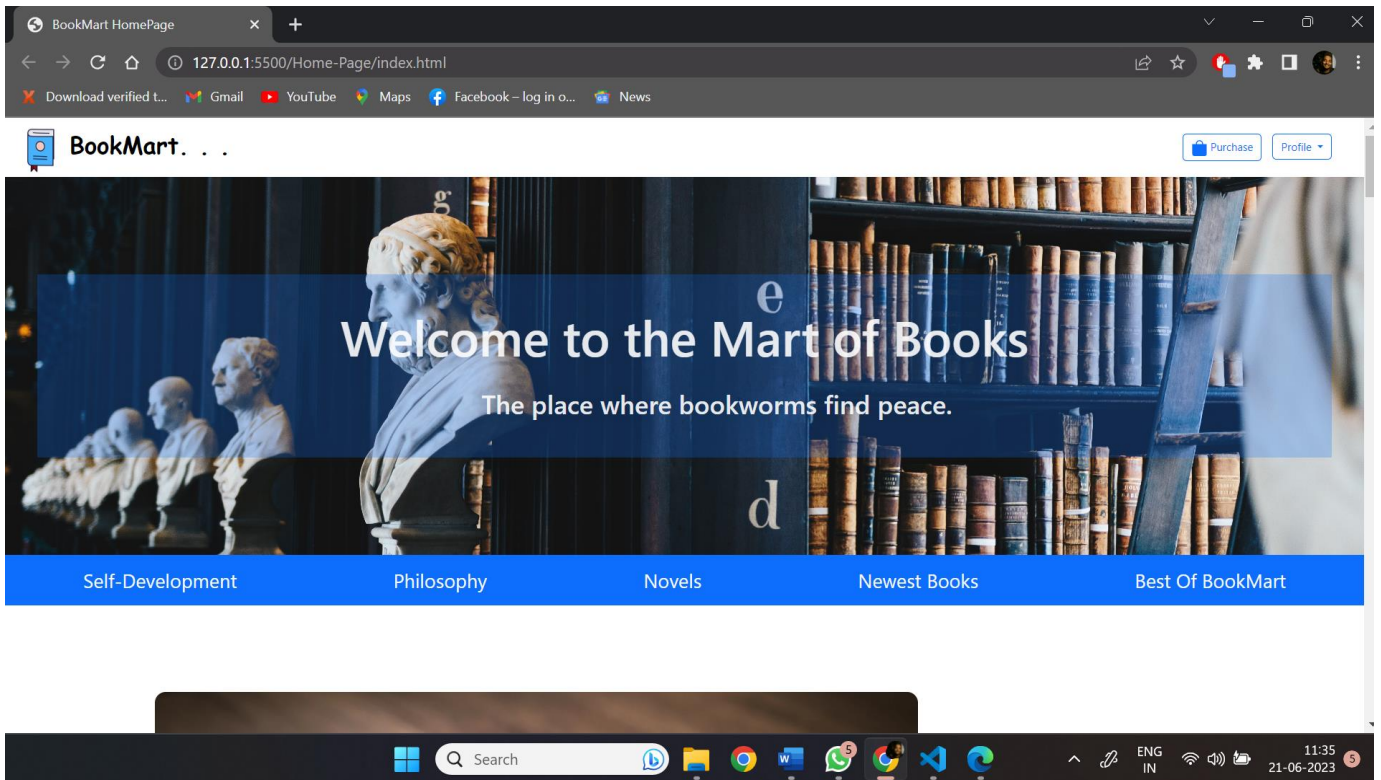
var data=[];
document.addEventListener("DOMContentLoaded", function() {
  var user =sessionStorage.getItem("username");
  var payment = document.getElementById("payment");
  payment.addEventListener("submit", function(event) {
    event.preventDefault();
    let currentDate = new Date().toJSON().slice(0, 10);
    // console.log(currentDate);
    data.push( {
      Fname:
        document.getElementById("fname").value,
      Nname:
        document.getElementById("nname").value,
      doo:
        currentDate,
      num:
        document.getElementById("num").value,
      email:
        document.getElementById("email").value,

```

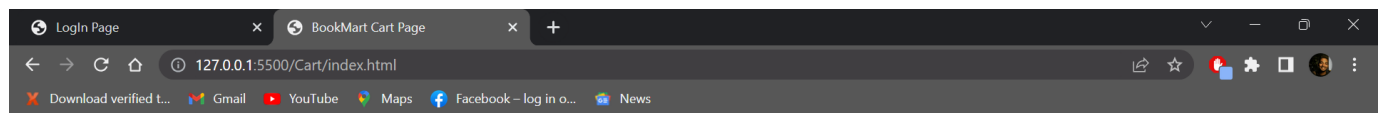
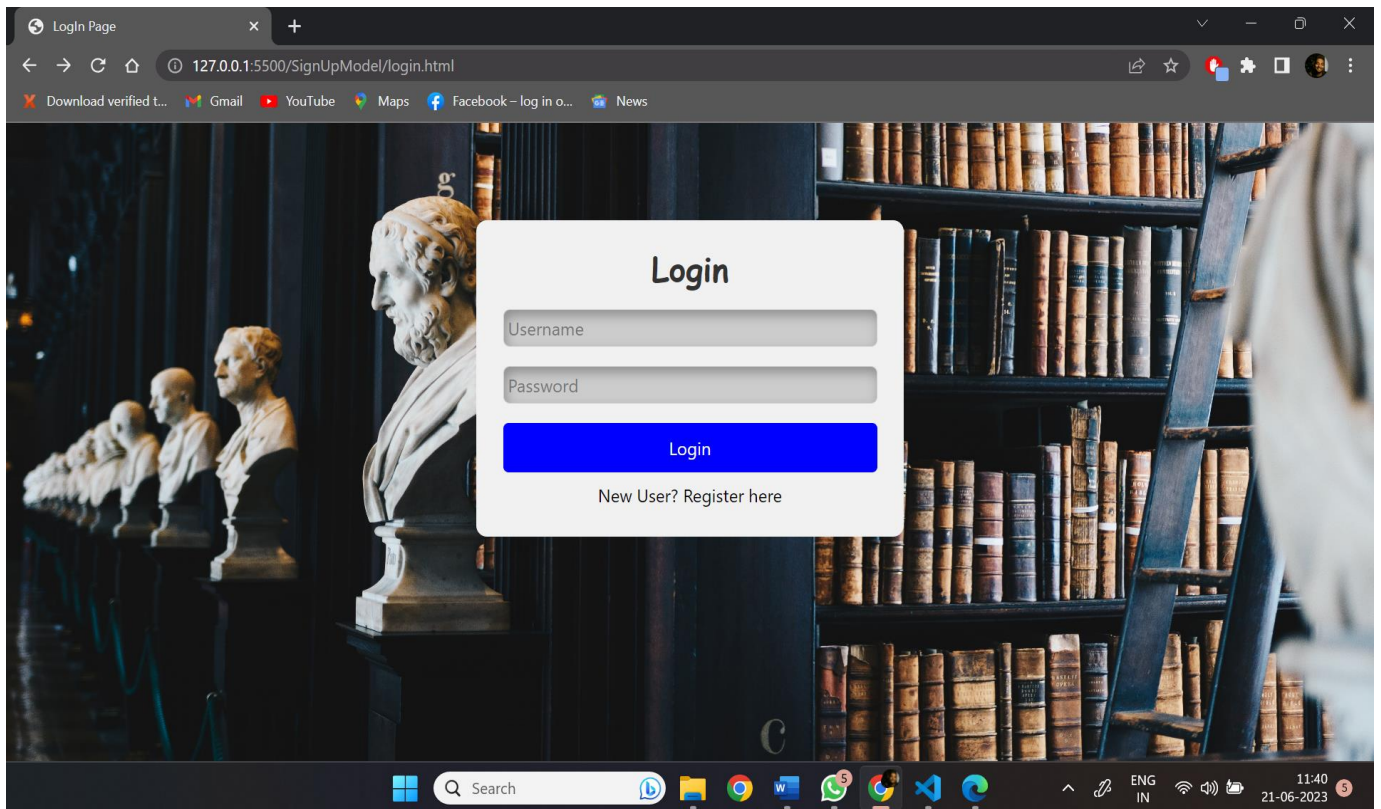
```
    address:
    document.getElementById("add").value,
    orderid:
    Math.floor(Math.random() * 10000),
    price:
    JSON.parse(localStorage.getItem(user)).price
});
key=JSON.parse(localStorage.getItem(user)).num;
localStorage.setItem(key,JSON.stringify(data));
window.open("load.html","_self");
});
});
var user =sessionStorage.getItem("username");
```

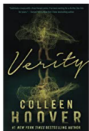

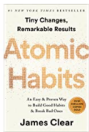

i. **SCREEN SHOT:**

HomePage & Registration:



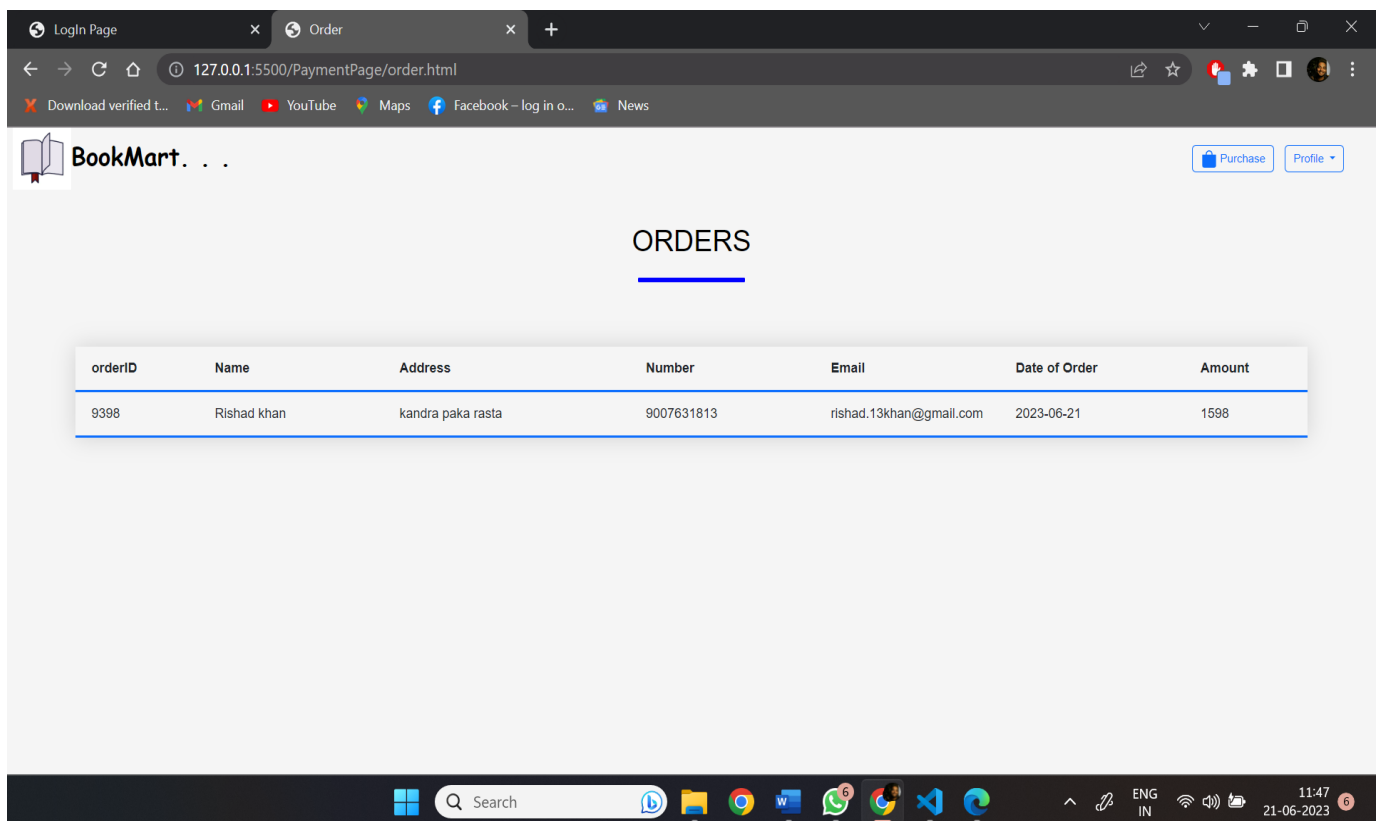
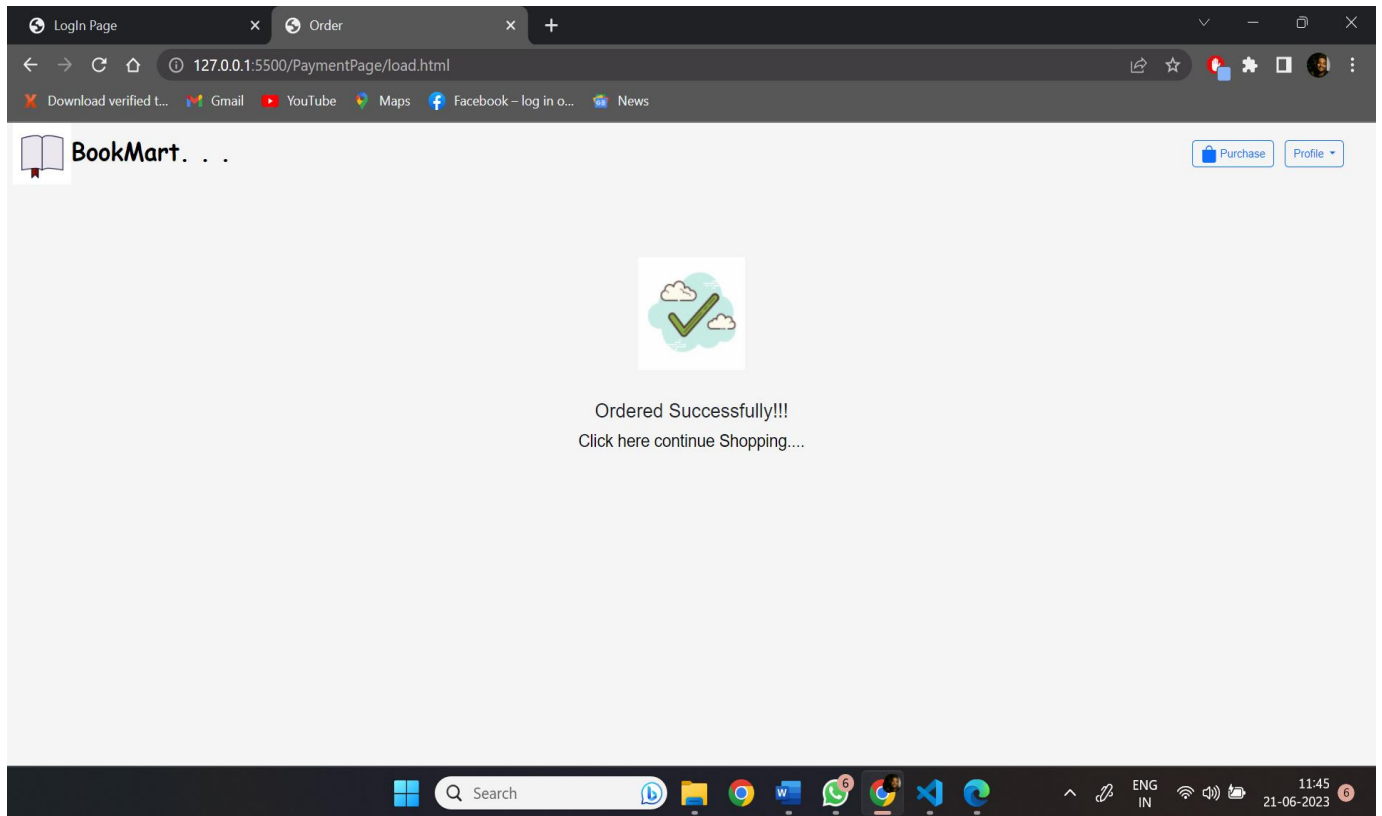
Log In & Cart :



| Shopping Bag | | Price | |
|---|---------------|-------|--|
|  | Verity | + | ₹799  |
|  | Atomic Habits | + | ₹799  |
| <input type="button" value="Proceed to Pay"/> | | | |



Order Page:



j. Methodology using System Testing:

After the completion of the Development phase of Software, the testing phase starts. In it, we test out of the software by the end users or self. There are various software testing modes. During this phase the system is used experimentally to ensure that the software does not fail, i.e. it will run according to its specifications and in a way users expect it to. Special data is input for processing and the results are examined to locate unexpected results. Persons other than the one who write original programs perform testing, i.e., using persons who do not know how certain parts were designed or programmed. It ensures more complete and unbiased testing and more reliable software.

This is testing the portions of the software and then before the system is implemented, all the components or processes are tested together to ensure that the entire system operates as required and should run error free.

Some of the major testing types are as follows:

- **White Box testing:** This type of testing goes inside the program and tests the paths, loops and branches in the code at least once to verify the programmer's intention.
- **Black Box testing:** This testing is done only by checking the outputs to see whether they are the expected ones. This type of testing verifies that the software generates the expected outputs with a given set of inputs.
- **Static analysis:** In this type of testing, the code is examined rather than exercised to verify its conformance to a set of criteria. This type of testing is most effective when it can be used to validate the traceability of software to a formal, mathematically rigorous specification. Such kinds of peer reviews are very effective in finding many kinds of errors.

- **Maintenance Strategy:**

This stage is optional in Software Life Cycle Model as it may exist or not. This phase is required to keep the software current. It has four types:

- **Adaptive Maintenance**
- **Preventive Maintenance**
- **Corrective maintenance**
- **Perfective Maintenance**

Now. In case of our products we need only adaptive and perfective maintenance. Adaptive maintenance makes software exist in the current hardware and software area and perfective

maintenance enables fine tuning and optimizing the system for faster and quicker response in a more efficient and effective way.

It does not need to install latest hardware and Software for keeping the initial database but as the database size increases gradually then frequent backups of the data should be taken.

There should be a way of logging the errors in a file. We would try to sort out these errors. After, removing the errors we would create the error free software. So, maintenance strategy helps in keeping the software at top-notch working more effectively and efficiently.

k. Test report:

Before we define the test cases, let us specifically state that this is not the canonical way to show test cases. A test case should give the exact input or action performed and the expected output. The tester then tests the system by using the test cases. The actual output is obtained and recorded in the test log. If the expected and actual outputs agree the test case is said to have “passed” else failed. Failed test cases are sent back to the development team for fixing and are then tested in the next round of testing. However, for the present we are giving some basic testing scenarios.

Login Form:

| S No. | Initial | Expected | Output |
|-------|---|-----------------------------|-----------|
| 1. | Login failure when Password is correct | Login when Password Matches | Wrong |
| 2. | Unable to access data from setting Table | Setting Table data accessed | Wrong |
| 3. | Password data accessed from Setting Table | Password Retrieved | Corrected |
| 4. | Login rechecked for Admin | Worked out | Corrected |

| S. No. | Initial | Expected | Output |
|--------|---|---|-----------|
| 1. | Menu bar doesn't deactivate according to login form | Menu bar should have visibility according to login type | Wrong |
| 2. | Image File mishandling if image file doesn't exist | Error! if no Image File exists | Wrong |
| 3. | Menu bar function according to login type | Worked out | Corrected |
| 4. | Error! If image not found | Worked out | Corrected |

Main Form:

Data Entries Form:

| S. No. | Initial | Expected | Output |
|--------|--|---|-----------|
| 1. | Form color settings not applied | Form color settings should be applied | Wrong |
| 2. | Primary key validation not working | Primary key validation available | Wrong |
| 3. | Foreign key validation not working | Foreign key validation available | Wrong |
| 4. | Numeric validation for numerical fields not functioning | Numeric validation for numerical fields should be working | Wrong |
| 5. | No E-mail validation facility | Checking of valid e- mail id | Wrong |
| 6. | Grid view of records not working | Grid view of record should be displayed | Wrong |
| 7. | Form color settings should be applied | Worked out | Corrected |
| 8. | Primary key validation available with error if duplicate | Worked out | Corrected |
| 9. | Foreign key validation available with proper error! | Worked out | Corrected |
| 10. | Numeric validation for numerical fields | Worked out | Corrected |
| 11. | Checking of valid entered e-mail id | Worked out | Corrected |
| 12. | Grid view of record should be displayed | Worked out | Corrected |

l. CONCLUSION:

We accomplished the project with all the features which are documented in project synopsis are successfully implemented. We in-depth explored the latest web technologies such as HTML5, CSS3, BOOTSTRAP and JAVASCRIPT for building this project.

We have learned a lot while developing the project as this was the first hands on project in our education career which can easily be plugged in any existing web-based application or can run as a stand-alone web application as well. Starting from the beginning I.e. feasibility study that of the implementation phase at the last, every personnel have sincerely supported and extended their cooperation with out which this project might not be built like that of the present status. The knowledge we gain while doing this project has become a limestone in our learning path which shall definitely become a catalyst for our future projects under any scenario.

M. Future Scope and Further Enhancement of The Project:

The future scope and potential enhancements for an e-commerce website are vast, as technology and consumer preferences continue to evolve. Mobile commerce (m-commerce) is rapidly growing, and it's essential to ensure your website is fully optimized for mobile devices. Enhance the user experience by implementing responsive design, intuitive navigation, and seamless mobile checkout processes. Utilize data analytics and customer behavior tracking to offer personalized recommendations and targeted promotions. Implement machine learning algorithms to understand user preferences, optimize search results, and provide customized shopping experiences. Leverage social media platforms to facilitate shopping experiences. Enable users to make purchases directly within social media apps, leverage user-generated content for product recommendations, and integrate social sharing functionalities to enhance brand exposure and customer engagement. Enhance customer support and engagement by implementing AI-powered chatbots. These chatbots can provide instant assistance, answer frequently asked questions, and guide users through the purchasing process, thereby improving customer satisfaction and reducing support costs.

There is a scope to improve the website's design to BookMart Functionality and better customer approach.

n. Reference:

➤ ***Bibliography:***

✓ ***Books:***

- i. Effective JavaScript: 68 specific ways to harness the power of JavaScript by David Herman
- ii. Object-Oriented Software Engineering: Using UML, Patterns and Java by Bernd Bruegge, Allen H. Dutoit.

✓ ***Website:***

- i) www.google.co.in
- ii) www.w3school.org
- iii) <https://www.javascript.com/>
- iv) www.wikipedia.org
- v) <https://www.youtube.com/>