A

Project Report

On

E - COMMERCE (AYUCARE)

MCA113-MINI PROJECT

By

Yash Chauhan

3123015010155

Mini Project work carried out at

Faculty of Commerce and Management

VISHWAKARMA UNIVERSITY

PUNE

December 2023

DECLARATION

I, the undersigned, hereby declare that the project report entitled "**E-commerce (Ayuacre)**" written and submitted by me to Vishwakarma University, Pune in partial fulfillment of the requirement for Mini Project (Semester I) as part of two-year full-time MCA programme. The project is for academic purposes only. This is the original work of the undersigned and has not been reproduced from any other sources.

Signature:

Name of the Student: Yash Chauhan

SRN: 31231418

Date: 12/12/2023

Place : Vishwakarma University, Pune

Mini Project Completion Certificate

MCA Programme

Batch: 2023-2025

This is to certify that Mr. / Ms. Yash Chauhan, student of Master of Computer Application (MCA) Programme under Faculty of Commerce and Management at Vishwakarma University, Pune, has completed the Mini Project titled "**E-commerce (Ayucare)**" as per university guidelines for the two-year full-time MCA programme.

Faculty Guide (Name):

Faculty Guide (Signature):

ACKNOWLEDGEMENTS

This project would not have been possible without the support of many people. I would like to acknowledge and extend my heartfelt gratitude to the following persons who have made the completion of this project possible.

My Supervisor, **Prof. Yogesh Desale** for his full support and guidance throughout this project. My additional examiner **Prof. Gauri Kulkarni** for reviewing project and helping me in design. My dear **Dr. Jayashree Vispute** for encouraging and supporting me to pursue the project.

I also thank to Vishwakarma University, Pune team for step by step updating regarding project with nice taxila support.

My team members who have helped me during various phases of the project. Last but not the least; I would like to express my love and gratitude to my beloved family, for their understanding & motivation, through the duration of this project.

ABSTRACT

This abstract outline the key features and benefits of an advanced E-commerce Web Application designed to elevate user experience and streamline online shopping processes. The application integrates cutting-edge technologies to provide a seamless and efficient platform for both consumers and merchants.

The E-commerce Web Application employs a user-centric design, prioritizing intuitive navigation and a visually appealing interface. Leveraging responsive design principles, the application ensures a consistent and optimized experience across various devices, including desktops, tablets, and smartphones. The user interface is thoughtfully crafted to enhance user engagement and simplify the shopping journey.

Incorporating secure and efficient payment gateways, the E-commerce Web Application ensures a smooth and trustworthy transaction process. The platform facilitates an easy and efficient product management system for merchants, allowing them to update inventory, track sales, and manage orders seamlessly. The application also integrates with third-party logistics and shipping services to provide real-time tracking and delivery updates, contributing to a transparent and reliable shopping experience.

Key words: Web Application, Responsive Design, Payment Gateways, Immersive Shopping

Signature of the Student	**Signature of the Supervisor**
Name: Yash Chauhan	**Name: Prof. Yogesh Desale**
Date: 11/12/2023	**Date:11/12/2023**
Place: Pune	**Place: Pune**

Tables of Contents

Table of Figures

Chapter 1 Introduction

1.1 What is an ecommerce web app?

An eCommerce web app, short for electronic commerce web application, is an online platform that facilitates the buying and selling of goods and services over the internet. It provides a virtual marketplace where businesses and consumers can engage in online transactions. Here are some

1.1.1 key features

Product Catalog:

Displays a list of products or services available for purchase, including detailed descriptions, images, and prices.

Shopping Cart:

Allows users to add products to a virtual cart for later purchase. Users can view and manage the items in their cart before proceeding to checkout.

User Accounts:

Enables users to create accounts, log in, and manage their personal information, order history, and preferences.

Search Functionality:

Helps users find products quickly by implementing search features with filters and sorting options.

Payment Integration:

Supports various payment methods, such as credit cards, digital wallets, and other online payment systems.

Checkout Process:

Guides users through the steps to complete a purchase, including entering shipping information and selecting a payment method.

Order Management:

Allows users to track the status of their orders, view order history, and manage returns or cancellations.

Security Features:

Implements secure protocols for transactions, encrypts sensitive data, and ensures the protection of user information.

Responsive Design:

Ensures the web app is accessible and user-friendly across different devices, including desktops, tablets, and smartphones.

Inventory Management:

Helps businesses keep track of their product stock, update product availability, and manage product listings.

1.2 Why do we need Ecommerce app

E-commerce apps fulfill several important roles and provide numerous benefits, contributing to their widespread adoption and continued relevance. Here are some key reasons why we need e-commerce apps:

Convenience:

E-commerce apps offer unparalleled convenience, allowing users to browse and shop for products or services anytime, anywhere.

Users can make purchases with just a few taps on their mobile devices, eliminating the need to visit physical stores.

24/7 Accessibility:

E-commerce apps provide round-the-clock access to products and services, enabling users to shop at any time of the day or night, regardless of business hours.

Global Reach:

E-commerce apps break down geographical barriers, enabling businesses to reach a global audience without the need for a physical presence in every location.

Variety and Selection:

Users have access to a vast array of products and services from various vendors, giving them a wide range of choices and the ability to find specific items easily.

Personalization:

E-commerce apps leverage user data and preferences to provide personalized recommendations, enhancing the overall shopping experience and increasing customer satisfaction.

Time Savings:

Users can save time by quickly locating products, comparing prices, and making purchases online, without the need for time-consuming trips to brick-and-mortar stores.

Cost Efficiency:

E-commerce platforms often offer competitive pricing, discounts, and promotions, providing users with cost-effective options for their purchases.

Secure Transactions:

E-commerce apps implement secure payment gateways and encryption protocols, ensuring that users' financial information is protected during transactions.

Order Tracking and Notifications:

E-commerce apps provide real-time order tracking, allowing users to monitor the status of their purchases and receive timely notifications on shipping and delivery updates.

Feedback and Reviews:

Users can access and contribute to product reviews, fostering transparency and helping others make informed decisions based on the experiences of previous buyers.

Mobile Optimization:

With the increasing use of smartphones, e-commerce apps are optimized for mobile devices, providing a user-friendly experience on smaller screens.

Chapter 2: System Analysis

2.1. Objectives and Scope:

Objectives: Increase sales, improve user experience, and streamline the shopping process.

Scope: Web and mobile platforms, featuring product browsing, shopping cart, secure checkout, and order management.

2.2 Requirements Gathering:

Functional Requirements:

- ✧ User registration and authentication.
- ✧ Product catalog with search and filter options.
- ✧ Shopping cart functionality.
- ✧ Secure payment processing.
- ✧ Order confirmation and tracking.
- ✧ Non-Functional Requirements:
- ✧ Response time: <2 seconds.
- ✧ Security: SSL encryption, secure user authentication.

Scalability:

- ✧ Handle up to 10,000 simultaneous users.
- ✧ Cross-browser and cross-device compatibility.

3. Current System Analysis:

- ✧ Competitor analysis indicates the need for a user-friendly interface, quick checkout process, and personalized recommendations.

4. Use Case Modeling:

- ✧ Use Cases:
- ✧ User registration.
- ✧ Browsing products.
- ✧ Adding/removing items from the shopping cart.
- ✧ Secure checkout process.
- ✧ Order tracking.

5. Data Modeling:

Entities:

- ✧ User
- ✧ Product
- ✧ Order
- ✧ Payment

Relationships:

- ✧ Users place orders.
- ✧ Products are part of orders.
- ✧ Orders include payment details.

6. System Architecture:

- ✧ High-Level Architecture
- ✧ Frontend: React for web, React Native for mobile.
- ✧ Backend: Node.js with Express.
- ✧ Database: MongoDB for product and user data

7. Security Analysis:

- ✧ SSL/TLS for data encryption.
- ✧ Secure user authentication using JWT.
- ✧ Regular security audits to identify and address vulnerabilities.

8. User Interface (UI) Design:

- ✧ User-friendly design with intuitive navigation.
- ✧ Responsive layout for a seamless experience on various devices.

9. Prototyping:

- ✧ Developed interactive prototypes for user testing.
- ✧ Incorporated feedback to refine the user interface and flow.

Chapter 3. Software and Hardware Requirements:

3.1 Software Specifications:

Software's that are used for implementation is as follows:

- Language: css, HTML,Javascript
- Database: MongoDB
- Operating System: Windows 7, Windows 8, Windows 10 and above

3.2 Hardware Specifications:

The minimum hardware requirements for running the project are like below

- Processor: Multi-core processors (e.g., quad-core or higher) for handling concurrent user requests.
- RAM: 4 GB
- Hard Disk: 8gb

Chapter 4. Selected Software

4.1 React js:

React.js, commonly referred to as React, is a popular open-source JavaScript library for building user interfaces (UIs) or user interface components. Developed and maintained by Facebook, react has gained widespread adoption in the web development community due to its efficiency and flexibility. Here are some key features and concepts associated with React:

Component-Based Architecture:

React follows a component-based architecture where UIs are broken down into reusable and modular components. This approach promotes code reusability and maintainability.

Virtual DOM (Document Object Model):

React uses a virtual DOM to optimize the rendering process. Instead of directly manipulating the actual DOM, React creates a virtual representation of it in memory and updates only the necessary parts. This minimizes the number of actual DOM manipulations, improving performance.

JSX (JavaScript XML):

React uses JSX, a syntax extension for JavaScript that allows developers to write HTML-like code within JavaScript files. JSX makes the code more readable and concise, and it is later transpiled to standard JavaScript by tools like Babel.

Unidirectional Data Flow:

React enforces a unidirectional data flow, meaning that data in an application flows in a single direction. This makes it easier to understand how data changes over time and helps prevent common bugs related to data synchronization.

React Components:

React components are the building blocks of a React application. They can be either functional components (stateless) or class components (stateful). Components define how the UI should appear based on their current state and properties.

State and Props:

State represents the data that a component can maintain and modify. Props (short for properties) are inputs that a parent component passes to its child components. By managing state and props, React components can dynamically update and render UI elements.

Lifecycle Methods:

Class components in React have lifecycle methods that allow developers to execute code at specific points in the component's life, such as when it is mounted, updated, or unmounted. This provides control over the rendering process.

React Hooks:

Introduced in React 16.8, hooks are functions that allow functional components to use state and lifecycle features previously available only in class components. Hooks, such as useState and useEffect, simplify the management of component state and side effects.

Routing:

While React itself focuses on the UI, developers often use additional libraries for routing in single-page applications (SPAs). React Router is a popular choice for handling navigation and URL management.

React Ecosystem:

React has a rich ecosystem with numerous libraries and tools that complement its functionality. For example, Redux is commonly used for state management in larger applications, and Axios is a popular library for making HTTP requests.

4.1.1 Advantages Of React js

React.js offers several advantages for web development, making it a popular choice among developers

Virtual DOM for Improved Performance:

React uses a virtual DOM, which is a lightweight copy of the actual DOM. By updating the virtual DOM and then efficiently applying the changes to the real DOM, React minimizes the number of manipulations, resulting in improved performance and faster rendering.

Component-Based Architecture:

React follows a component-based architecture, allowing developers to create reusable and modular UI components. This promotes code reusability, maintainability, and a more organized code structure.

Declarative Syntax with JSX:

React uses JSX (JavaScript XML), a syntax extension that allows developers to write HTML-like code within JavaScript files. This declarative syntax makes the code more readable and helps visualize the structure of the UI.

One-Way Data Binding:

React enforces a unidirectional data flow, making it easier to manage state and understand how data changes over time. This helps prevent common bugs related to data synchronization in complex applications.

Reusability of Components:

React components can be easily reused throughout an application, promoting the development of modular and scalable UIs. This reusability is especially beneficial for large projects with complex user interfaces.

React Native for Cross-Platform Development:

React Native, a framework based on React, allows developers to build mobile applications for iOS and Android using the same React principles. This enables cross-platform development and code reuse between web and mobile applications.

Ecosystem and Community Support:

React has a thriving ecosystem with a vast array of libraries, tools, and extensions. The active community contributes to continuous improvements, provides support, and shares best practices, making it easier for developers to find solutions to common challenges.

Strong Developer Tools:

React comes with a set of powerful developer tools that enable efficient debugging, profiling, and inspecting the component hierarchy. These tools contribute to a smoother development and debugging experience.

4.2 HTML

HTML, which stands for HyperText Markup Language, is the standard markup language for creating web pages. It provides the structure and semantics of content on the web. HTML is essential for web development and is used in conjunction with Cascading Style Sheets (CSS) and JavaScript to build interactive and visually appealing websites. Here are some key aspects of HTML:

HTML Document Structure:

An HTML document is structured with various elements, each serving a specific purpose. The basic structure includes the <!DOCTYPE html> declaration, the <html> element, the <head> section (containing metadata), and the <body> section (containing the content).

HTML Elements:

HTML elements are the building blocks of a web page. They are defined by tags, such as <p> for paragraphs, <h1> for headers, <a> for links, <img> for images, and many others. Elements can have attributes that provide additional information.

Attributes:

HTML elements can have attributes that provide additional information or configuration. Attributes are included within the opening tag of an element and are written as name-value pairs. For example, the <a> element may have an href attribute specifying the link destination.

Headings and Paragraphs:

HTML provides heading elements <h1> to <h6> for defining headings of different levels. Paragraphs are created using the <p> element.

Links:

Links are created using the <a> (anchor) element. The href attribute is used to specify the URL to which the link points. Links can navigate to other pages, resources, or sections within the same page.

Images:

Images are inserted using the <img> element. The src attribute specifies the path to the image file. Additional attributes, such as alt for alternative text, can provide more information about the image.

Lists:

HTML supports ordered lists <ol> (numbered) and unordered lists <ul> (bulleted). List items are defined using the <li> element.

Forms:

Forms are created with the <form> element and can include input elements like text fields (<input type="text">), checkboxes (<input type="checkbox">), radio buttons (<input type="radio">), and more. The form's data is typically submitted to a server for processing.

Tables:

Tables are constructed using the <table> element, with rows defined by <tr> and cells by <td> (data cell) or <th> (header cell). Tables are commonly used to organize and display tabular data.

4.3 Mongodb Atlas

MongoDB Atlas is a fully managed cloud database service provided by MongoDB, Inc. It is designed to make it easy to deploy, operate, and scale MongoDB databases in the cloud. MongoDB Atlas offers a range of features and benefits for developers and organizations looking to leverage MongoDB in a cloud environment. Here are some key aspects of MongoDB Atlas:

Cloud-Hosted MongoDB:

MongoDB Atlas allows users to deploy MongoDB databases on major cloud platforms such as AWS (Amazon Web Services), Azure (Microsoft Azure), and GCP (Google Cloud Platform). This eliminates the need for users to manage the underlying infrastructure.

Fully Managed Service:

MongoDB Atlas is a fully managed database service, which means that tasks such as hardware provisioning, setup, configuration, maintenance, and backups are handled by MongoDB Atlas. This allows developers to focus more on building applications and less on database management.

Automated Backups and Point-in-Time Recovery:

MongoDB Atlas provides automated backups with configurable snapshot schedules. This enables point-in-time recovery, allowing users to restore their database to a specific moment in time.

Security Features:

MongoDB Atlas includes various security features to protect data, including encryption at rest, encryption in transit, network isolation, and access controls. It also integrates with identity providers such as AWS IAM and Azure Active Directory for authentication.

Scalability:

MongoDB Atlas offers seamless horizontal scaling, allowing users to easily scale their databases by adding or removing shards. It also provides features like automated scaling and data tiering to optimize performance and cost.

Global Clusters:

MongoDB Atlas supports the creation of global clusters that allow users to deploy databases in multiple regions around the world. This helps in achieving low-latency access for users in different geographic locations.

Monitoring and Analytics:

MongoDB Atlas provides a monitoring and analytics dashboard that gives users insights into the performance and health of their databases. It includes real-time metrics, query analysis, and alerting capabilities.

Integration with Additional Services:

MongoDB Atlas integrates with various cloud services and tools, including AWS CloudWatch, Azure Monitor, and GCP Stackdriver. This allows users to leverage additional services for monitoring, logging, and analytics.

MongoDB Atlas Data Lake:

MongoDB Atlas Data Lake allows users to query and analyze data across AWS S3 data lakes and MongoDB Atlas clusters. This enables the exploration and analysis of data stored in diverse environments.

4.3.1 Advantages of Mongodb Atlas:

Ease of Use:

MongoDB Atlas simplifies the deployment and management of MongoDB databases. Users don't need to worry about infrastructure provisioning, configuration, or maintenance tasks, allowing them to focus more on application development.

Fully Managed Service:

As a fully managed database service, MongoDB Atlas takes care of routine administrative tasks, including backups, updates, and scaling. This reduces the operational burden on development teams.

Automated Backups and Point-in-Time Recovery:

MongoDB Atlas provides automated and regular backups with point-in-time recovery options. This ensures data durability and offers the ability to restore databases to a specific historical state.

Scalability:

MongoDB Atlas supports seamless horizontal scaling. Users can easily scale their databases by adding or removing shards, allowing applications to handle increased workloads without significant operational overhead.

Global Distribution:

With support for global clusters, MongoDB Atlas enables the deployment of databases in multiple regions. This helps reduce latency for users accessing applications from different geographic locations.

Security Features:

MongoDB Atlas prioritizes security, offering features such as encryption at rest and in transit, network isolation, and role-based access controls. It integrates with identity providers for authentication and supports audit logging for compliance.

Chapter 5: System Design

5.1 UML Diagrams:

Unified Modeling Language or UML Diagrams are used to represent the system diagrammatically.

5.1.1 Use case diagram:

Use case diagrams are drawn to represent the functionality of the system.

- **Admin**

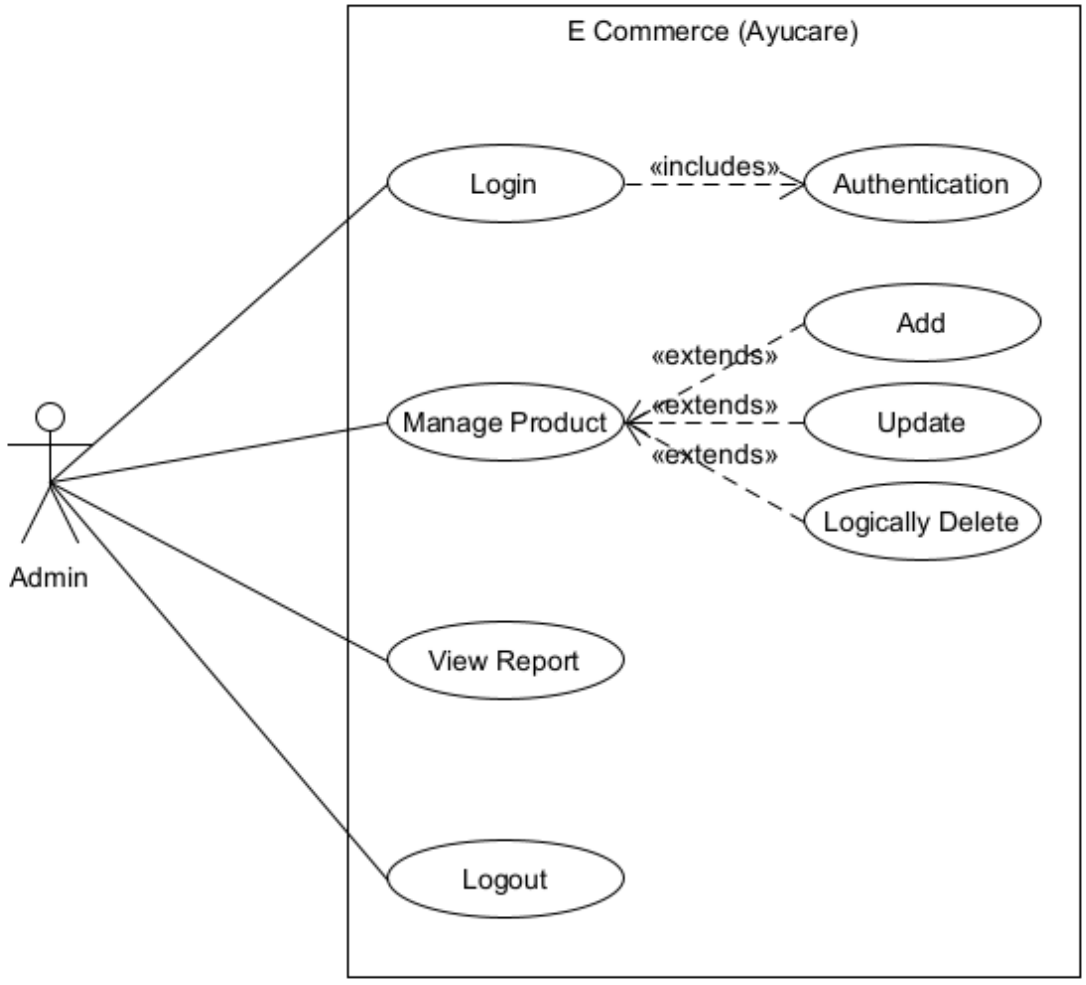
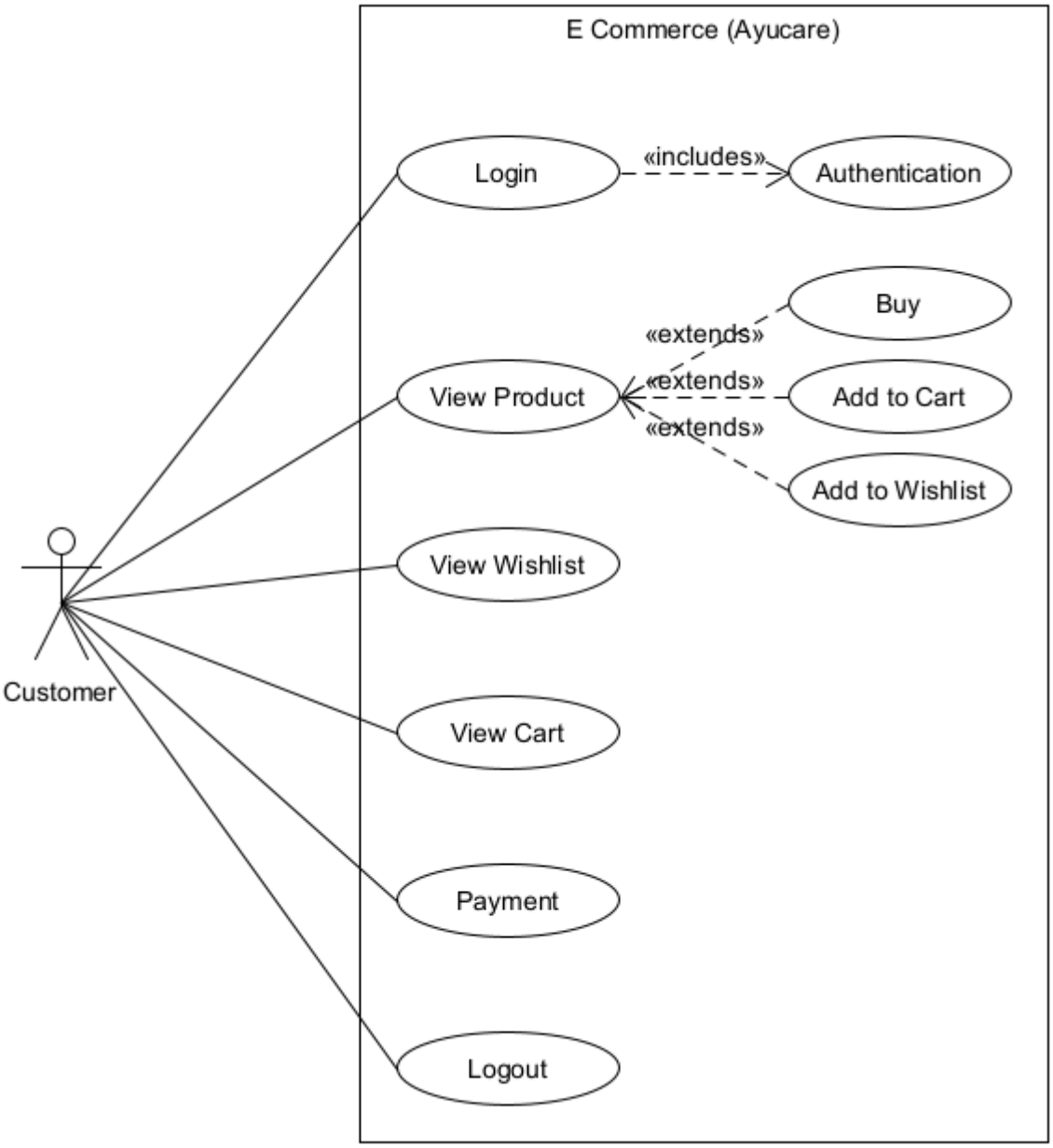


Fig1: Admin Use case

- **Customer**


E Commerce (Ayucare)
Login
«includes»
Authentication
Buy
«extends»
View Product
«extends»
Add to Cart
«extends»
Add to Wishlist
View Wishlist
Customer
View Cart
Payment
Logout

5.2 Class Diagram:

Class diagrams (Fig 4) are used to represent the classes used in the system and their relationships.

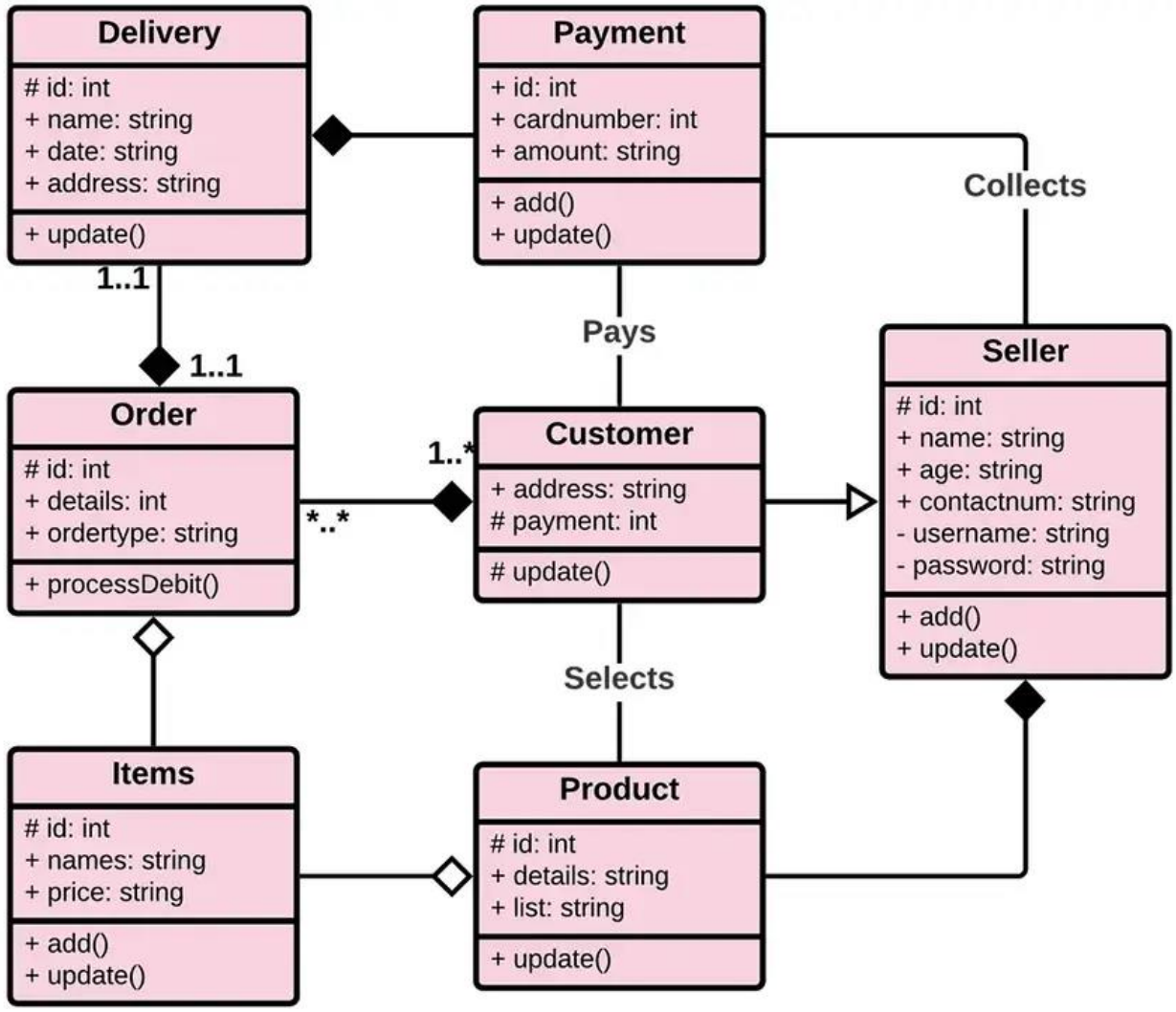


Fig3: Class Diagram

5.1.3 ERD DIAGRAM:

- **Entities:**
  - **Customer**:
  - **Product:**
  - **Order:**

- **Relationships:**

  Customer places Order:

  One-to-Many relationship between Customer and Order (One customer can place multiple orders)

  Order contains Product:

  Many-to-Many relationship between Order and Product (One order can contain multiple products, and one product can be in multiple orders)

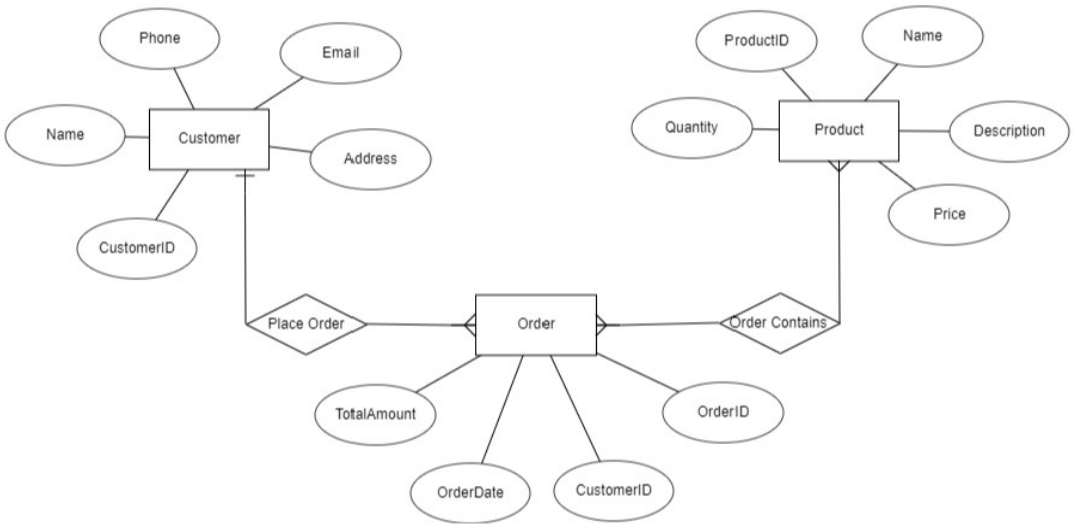
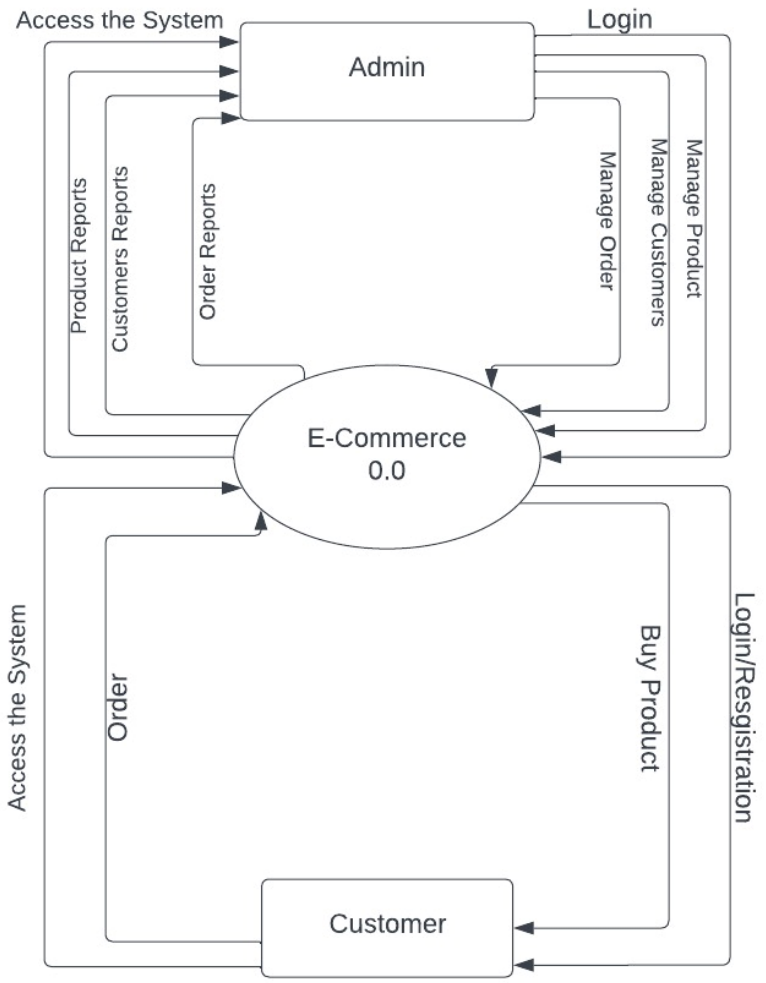
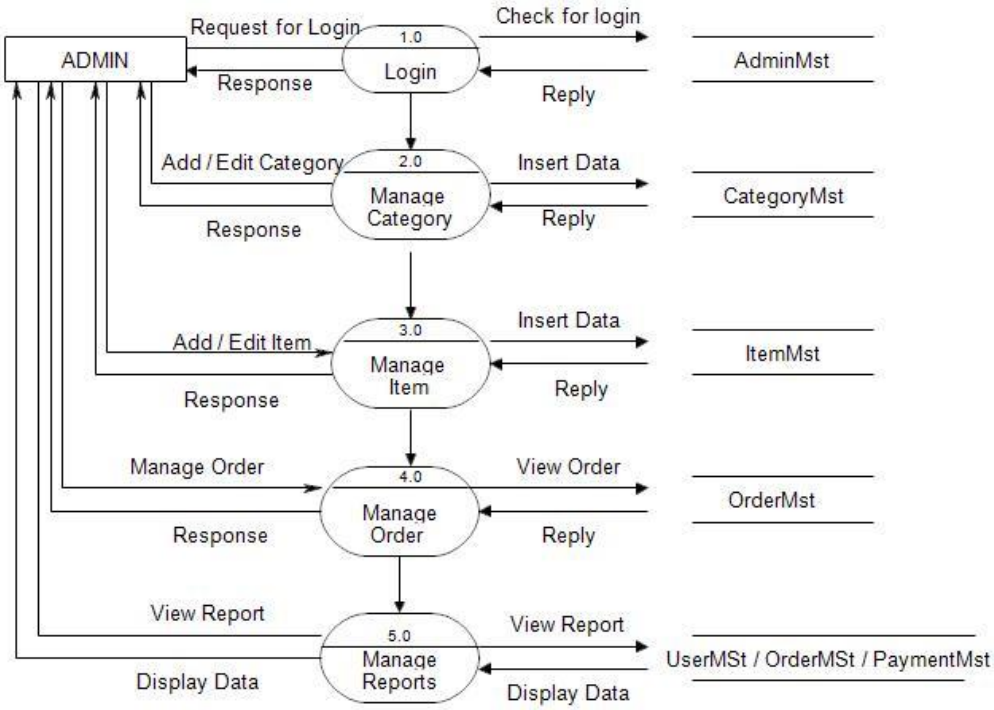


Fig4: Er diagram

5.1.4 DFD(Data Flow Diagram):



Admin Side DFD - 1st Level



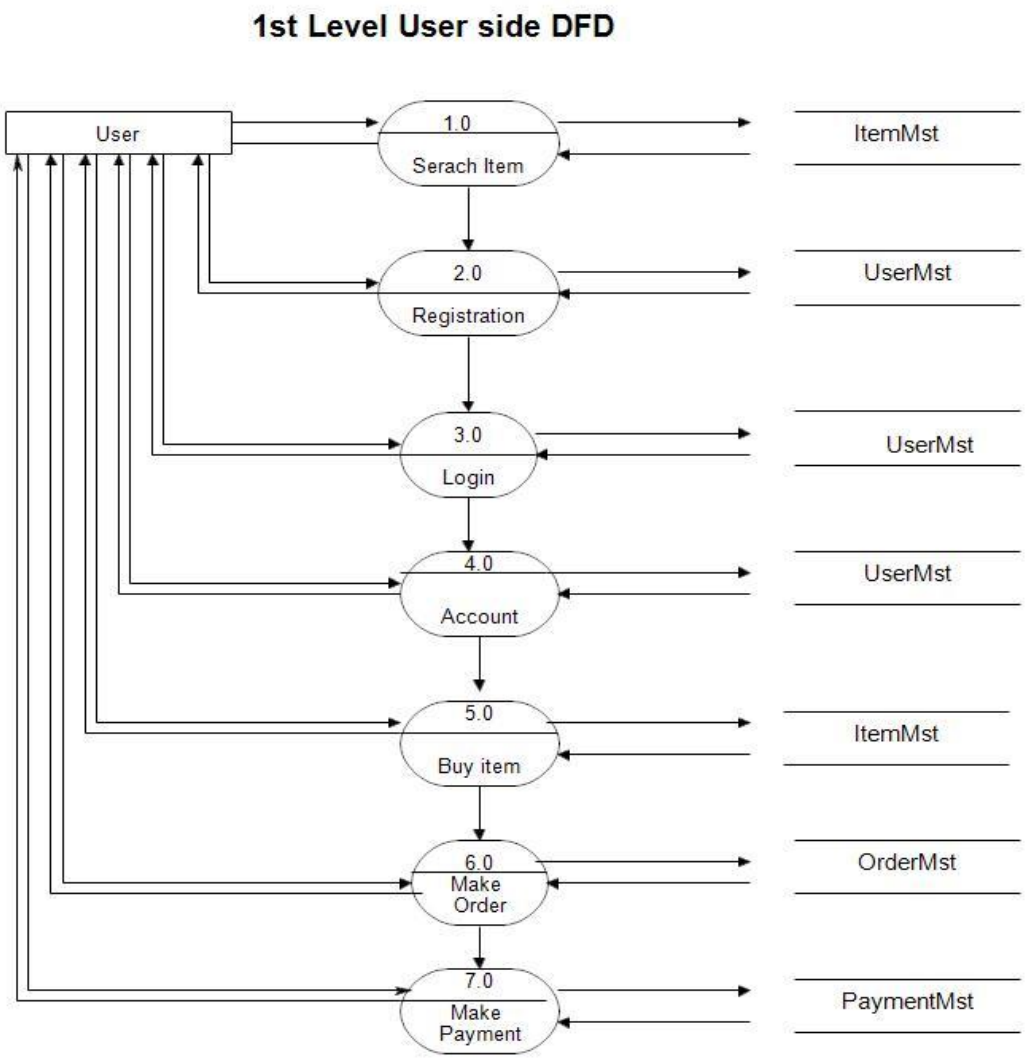

1st Level User side DFD
User
1.0
Serach Item
2.0
Registration
3.0
Login
4.0
Account
5.0
Buy item
6.0
Make Order
7.0
Make Payment
ItemMst
UserMst
UserMst
UserMst
ItemMst
OrderMst
PaymentMst

Fig4:DFD(2)

Chapter 6 System Implementation

6.1 Plan of implementation:

System Development Life Cycle (SDLC) Model

- ✓ **Waterfall model:**

The Waterfall Model follows a linear, step-by-step approach, where each phase must be completed before moving on to the next. This methodology is well-suited for projects with well-defined and stable requirements, where changes are expected to be minimal once the project has started.

Requirements Analysis:

Objective:

Gather and document detailed requirements for the eCommerce web app.

Activities:

Engage with stakeholders to define the scope, functionalities, and constraints. Document all requirements thoroughly.

System Design:

Objective: Create a detailed system design based on the gathered requirements.

Activities: Design the system architecture, modules, and components. Develop specifications for hardware, software, and network requirements.

Implementation:

Objective: Code the eCommerce web app based on the detailed system design.

Activities: Start coding individual modules and components. Conduct unit testing for each module to ensure they meet the specifications.

Integration:

Objective: Combine all developed modules to create the complete eCommerce system.

Activities: Integrate individual modules, ensuring they work together seamlessly. Conduct integration testing to identify and resolve any issues.

Testing:

Objective: Conduct thorough testing to verify the system meets specified requirements.

Activities: Perform system testing to validate the entire eCommerce web app. Identify and fix defects or issues. Conduct user acceptance testing (UAT) with stakeholders.

Deployment:

Objective: Deploy the eCommerce web app to the production environment.

Activities: Transfer the finalized system to the production servers. Conduct final checks and tests before making the app available to users. Train end-users and provide documentation.

Maintenance:

Objective: Address any issues or bugs that arise post-deployment and provide ongoing support.

Activities: Monitor the system for performance and issues. Address any bugs or problems promptly. Provide ongoing support and maintenance as needed.

6.2 Sample code

HomePage:

```
import React from 'react'
import Carousel from '../Components/Carousel'
import Features from "../Components/Features"
import Newsletter from '../Components/Newsletter'
import Herosection from '../Components/Herosection'
import jew2 from "../assets/jew2.jpg"
import fashion from "../assets/fashion.jpg"
const Homepage = () => {

  return (
    <div>

<div id="default-carousel" class="relative w-full" data-carousel="slide">

    <div class="relative h-56 overflow-hidden rounded-lg md:h-96">

        <div class="hidden duration-700 ease-in-out" data-carousel-item>
                                <img      src="https://images.unsplash.com/photo-1587749091716-
f7b291a87f87?q=80&w=1471&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"
class="absolute block w-full  object-fit -translate-x-1/2 -translate-y-1/2 top-1/2 left-1/2"
alt="..."/>
```

```
      </div>

      <div class="hidden duration-700 ease-in-out" data-carousel-item>
         <imgsrc={fashion} class="absolute block w-full -translate-x-1/2 -translate-y-1/2 top-1/2 left-1/2" alt="..."/>


      </div>
      <div class="hidden duration-700 ease-in-out" data-carousel-item>
                              <img src="https://images.unsplash.com/photo-1505022610485-0249ba5b3675?q=80&w=1470&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D" class="absolute block w-full -translate-x-1/2 -translate-y-1/2 top-1/2 left-1/2" alt="..."/>
      </div>


   </div>

   <div class="absolute z-30 flex space-x-3 -translate-x-1/2 bottom-5 left-1/2">
         <button type="button" class="w-3 h-3 rounded-full" aria-current="true" aria-label="Slide 1" data-carousel-slide-to="0"></button>
         <button type="button" class="w-3 h-3 rounded-full" aria-current="false" aria-label="Slide 2" data-carousel-slide-to="1"></button>
         <button type="button" class="w-3 h-3 rounded-full" aria-current="false" aria-label="Slide 3" data-carousel-slide-to="2"></button>
        {/* <button type="button" class="w-3 h-3 rounded-full" aria-current="false" aria-label="Slide 4" data-carousel-slide-to="3"></button>
         <button type="button" class="w-3 h-3 rounded-full" aria-current="false" aria-label="Slide 5" data-carousel-slide-to="4"></button> */}
   </div>

   <button type="button" class="absolute top-0 left-0 z-30 flex items-center justify-center h-full px-4 cursor-pointer group focus:outline-none" data-carousel-prev>
      <span class="inline-flex items-center justify-center w-10 h-10 rounded-full bg-white/30 dark:bg-gray-800/30 group-hover:bg-white/50 dark:group-hover:bg-gray-800/60 group-focus:ring-4 group-focus:ring-white dark:group-focus:ring-gray-800/70 group-focus:outline-none">
                  <svg class="w-4 h-4 text-white dark:text-gray-800" aria-hidden="true" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 6 10">
export default Contact
```

Chapter 7: System Testing

7.1 Unit testing:

Unit testing is a crucial aspect of the software development process, including when developing an eCommerce web app using the Waterfall Model or any other development methodology.

Unit testing involves testing individual units or components of a software application in isolation to ensure they perform as intended.

✓ **Testing Methodology.**

- Our testing approach followed a comprehensive strategy, encompassing unit testing, integration testing, and system testing. Test-driven development (TDD) practices were implemented to ensure that each component met its requirements before integration.

✓ **Type of Testing.**

- Unit Testing: Ensured individual components functioned as intended.
- Integration Testing: Verified the proper collaboration of integrated components.
- System Testing: Validated the end-to-end functionality of the entire E-commerce site.
- User Acceptance Testing (UAT): Involved stakeholders to confirm the system's compliance with business requirements.

✓ **Testing Tools.**

- **Jest and Enzyme**: For front-end unit testing.
- **ThunderClient**: For API testing and validation.
- **Selenium:** For automated browser testing. (**Used in System**)
- **Jenkins:** For continuous integration and automated testing.

Chapter 8. Sample Screen

8.1 Home Screen:

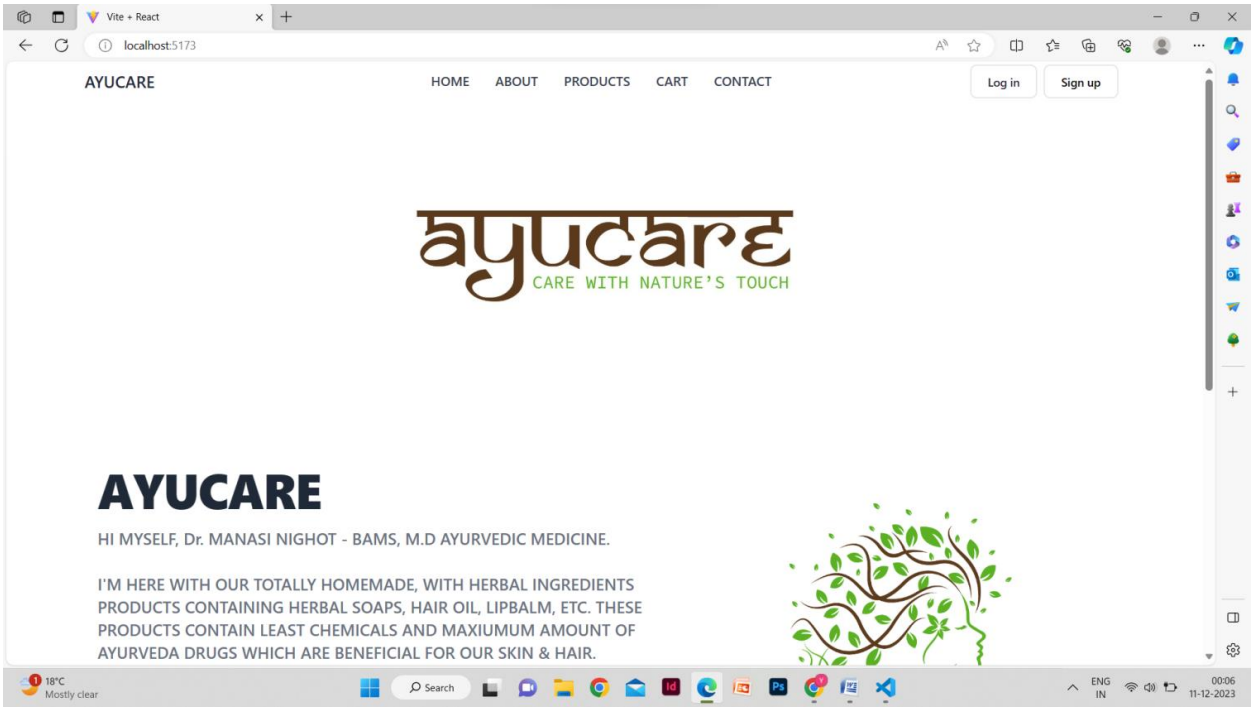


Fig5: Homescreen

8.2 Products screen:

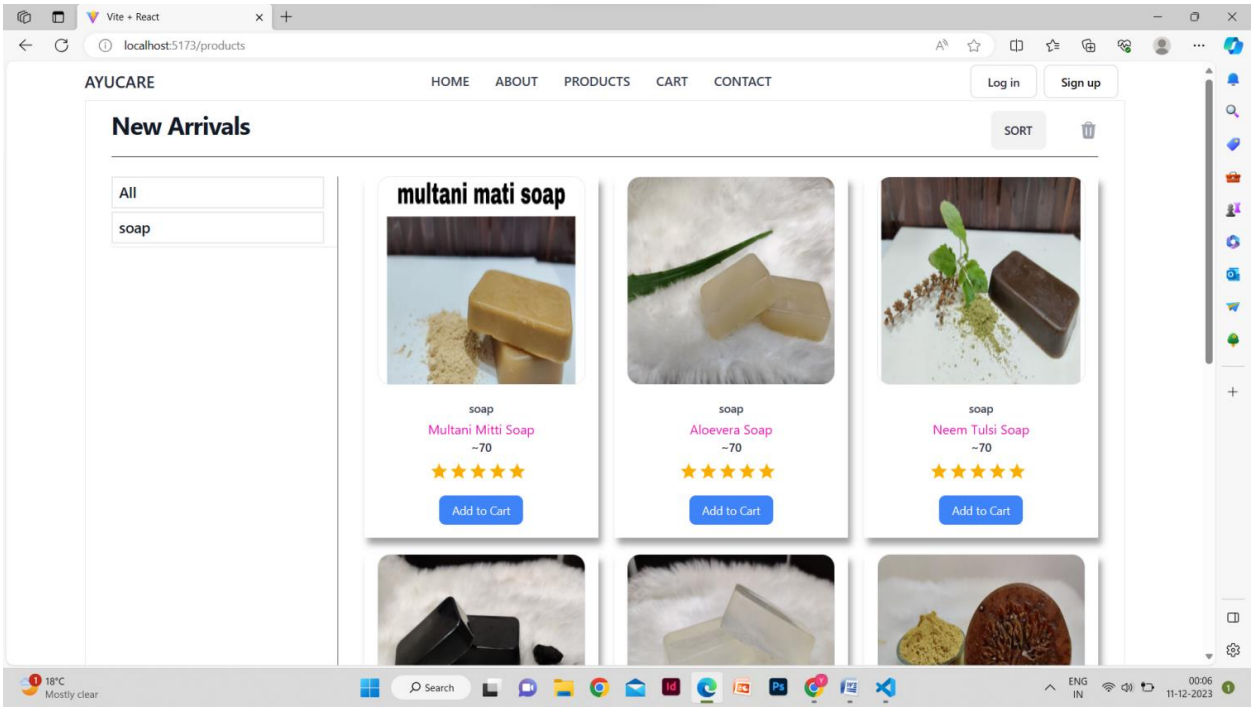


Fig6: Product page

8.3 Admin screen:

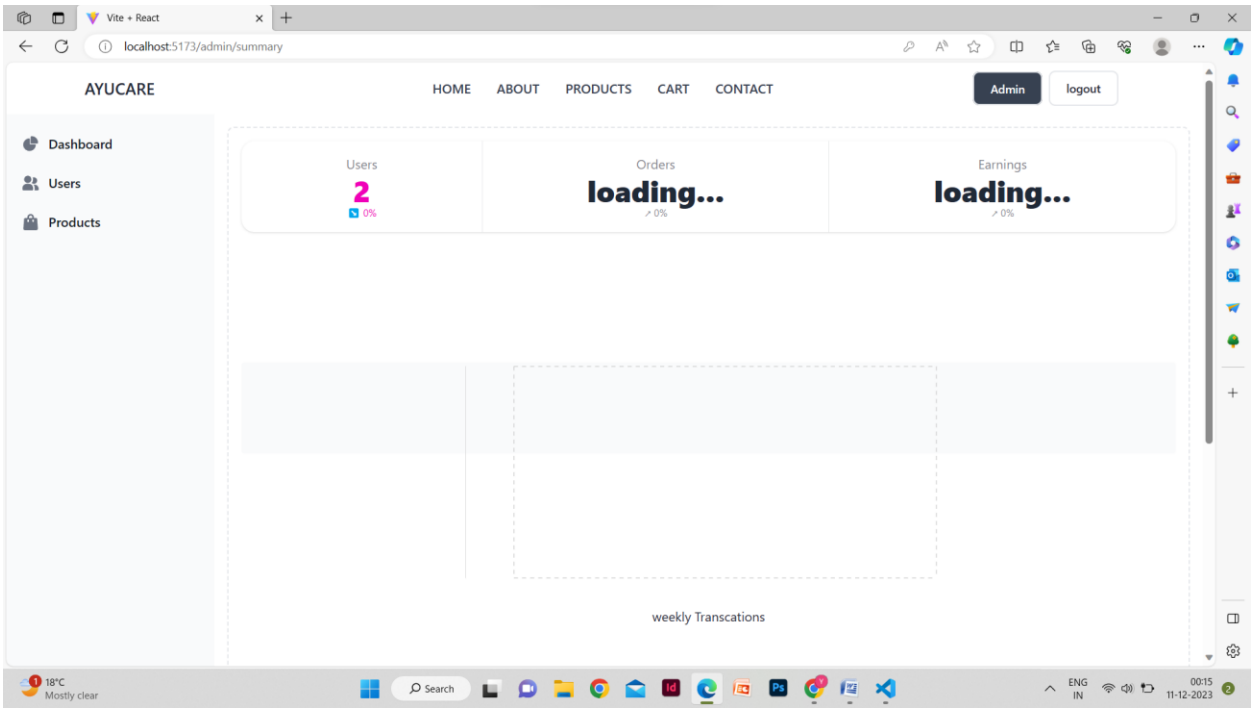


Fig7: Admin page

Conclusion

In conclusion, the development of an eCommerce website is a complex and multifaceted process that requires careful planning, strategic decision-making, and meticulous execution. Whether following a traditional Waterfall Model or a more iterative approach, certain key considerations and activities are essential for success.

The market research phase sets the foundation by understanding the target audience, analyzing competitors, and identifying market trends. This information informs the conceptualization and planning stage, where the purpose, features, and goals of the eCommerce website are defined. Careful selection of the technology stack ensures the chosen tools align with scalability, security, and integration requirements.

The design phase is critical for creating an attractive and user-friendly interface, involving wireframing, prototyping, and finalizing the visual design and branding elements. The subsequent development phase brings the design to life, implementing features such as product listings, shopping carts, user authentication, and payment processing.

Testing plays a vital role in ensuring the eCommerce website functions correctly and meets quality standards. From unit testing individual components to system testing and user acceptance testing, identifying and resolving issues is integral before deployment. The deployment phase involves making the website accessible to users, including server configurations, training, and documentation.

Ultimately, a successful eCommerce website requires a holistic approach that considers not only the technical aspects of development but also the user experience, market dynamics, and ongoing maintenance. Flexibility and responsiveness to changing market conditions and user expectations are key to ensuring the long-term success and sustainability of an eCommerce website.

References

- ✓ WWW.ReactJs.com
- ✓ www.MongodbAtlas.com
- ✓ WWW.github.com
- ✓ www.Tailwindcss.com
- ✓ Www.stripedocs.com
- ✓ www.Materialui.com