

# **ONLINE SHOPPING SYSTEM**

## **A MINI PROJECT REPORT**

**Submitted by**

**HARINI A 220701084**

**HINDUSHA K 220701092**

**In partial fulfillment for the award of the degree of**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE**



**RAJALAKSHMI ENGINEERING COLLEGE(AUTONOMOUS)**

**THANDALAM**

**CHENNAI-602105**

**2023-24**

# **BONAFIDE CERTIFICATE**

Certified that this Project report “ **ONLINE SHOPPING SYSTEM** ” is the bonafide work of “ **HARINI A (220701084)** ” & **HINDUSHA K (220701092)** ” who carried out the project work under my supervision.

Submitted for the Practical Examination held on \_\_\_\_\_

## **SIGNATURE**

**Dr.R.SABITHA**

**Professor and Academic Head,  
Computer Science and Engineering,  
Rajalakshmi Engineering College  
(Autonomous),  
Thandalam, Chennai - 602 105.**

## **SIGNATURE**

**Dr.G.Dharani Devi**

**Associate Professor,  
Computer Science and Engineering,  
Rajalakshmi Engineering College  
(Autonomous),  
Thandalam, Chennai - 602 105.**

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# ABSTRACT

An online shopping system is a web-based platform that allows customers to purchase products or services from the comfort of their own homes. The system typically consists of a front-end user interface, a back-end database, and a server-side application that manages the interactions between the user interface and the database.

The front-end user interface provides a user-friendly interface for customers to view product details, and place orders. The user interface may also include features such as user accounts, order history.

The back-end database stores the data for the online shopping system, including product information, customer information, and order information. The database is typically designed to handle large volumes of data and support fast and efficient data retrieval.

The server-side application manages the interactions between the user interface and the database. This includes handling user requests, retrieving and updating data in the database, and processing orders. The server-side application may also include features such as payment processing and weekly reports on the orders.

Overall, an online shopping system provides a convenient and efficient way for customers to shop for products and services online. The system must be designed to handle large volumes of data, support fast and efficient data retrieval, and ensure the security and integrity of the data. By providing a user-friendly interface and a reliable and efficient shopping experience, an online shopping system can attract and retain a large customer base and drive sales and revenue for the business.

# **TABLE OF CONTENTS**

## **1. INTRODUCTION**

- 1.1. Introduction
- 1.2. Objectives
- 1.3. Modules

## **2. SURVEY OF TECHNOLOGIES**

- 2.1. Software Description
- 2.2. Languages

## **3. REQUIREMENTS AND ANALYSIS**

- 3.1. Requirements and Specification
- 3.2. Hardware and Software Requirements
- 3.3. Architecture Diagram
- 3.4. ER Daigram
- 3.5. Normalization

## **4. PROGRAM CODE**

## **5. RESULTS AND DISCUSSION**

- 5.1 Output
- 5.2 Testing

## **6. CONCLUSION**

## **7.REFERENCES**

# Ch 1. INTRODUCTION

## 1.1 INTRODUCTION

Our Online Shopping website name is 'Cartopia' which is designed to revolutionize the digital shopping experience by offering unparalleled convenience and personalization. Whether you're a tech-savvy shopper or someone looking for an effortless way to purchase everyday essentials, Cartopia has been crafted to meet your needs.

### **Seamless Registration and Login**

The platform begins with a straightforward registration process, allowing new users to quickly create an account. Once registered, users can log in to access personalized features. This customization ensures a more engaging and efficient shopping experience.

### **Extensive Product Range**

Cartopia is neatly categorized into various sections such as groceries and skincare. We chose these two categories as nowadays they are in great demand. This wide array of products ensures that customers can find everything they need in one place, from fresh produce to the latest skincare innovations. Each category is intuitively organized, allowing for easy navigation and quick access to desired items.

### **User-Friendly Cart and Checkout**

Adding items to the cart is a breeze with Cartopia's intuitive interface. Users can effortlessly select, and review products before finalizing their purchases. The checkout process is designed to list the added items with their respective quantity and the total price. It supports payment process also. This flexibility ensures that transactions are smooth catering to a variety of user preferences.

### **Reports and Account Management**

Post-purchase, Cartopia provides a comprehensive report page where customers can view their order history, and manage their account informations.

### **Secure and Private**

Cartopia prioritizes user security and privacy. After completing their shopping, users can easily log out.

## 1.2 OBJECTIVE

### Objectives of Cartopia Online Shopping System

#### 1. User Registration and Authentication

- Objective: Implement a robust registration and login system to authenticate users and personalize their shopping experience.

- Details: Ensure a smooth and secure process for new users to create accounts and for existing users to log in. Incorporate features like secure authentication protocols.

#### 2. Comprehensive Product Catalog

- Objective: Provide an extensive and well-organized catalog of products, specifically focusing on grocery and skincare items.

- Details: Categorize products efficiently to allow easy navigation and quick access to desired items. Maintain up-to-date product information, images, and pricing.

#### 3. Seamless Shopping Cart Functionality

- Objective: Enable users to add, review, and modify products in their shopping cart effortlessly.

- Details: Implement features that allow users to adjust quantities, remove items, and view total costs before proceeding to checkout.

#### 4. User-Friendly Checkout Process

- Objective: Facilitate a secure, efficient, and hassle-free checkout experience with payment options.

- Details: Ensure the checkout process is straightforward, with clear steps and support for payment methods.

#### 5. Reporting

- Objective: Provide a comprehensive report page where users can view their past purchases.

- Details: Implement detailed purchase history to enhance user control and transparency.

## 6. Security and Privacy

- Objective: Protect user data and transaction information .
- Details: Ensure all personal and payment data is encrypted and securely stored. Facilitate easy and secure logout options to maintain user privacy.

## 7. Accessibility and Usability

- Objective: Design the platform to be accessible and user-friendly for all users.
- Details : Cartopia therefore aims to provide a reliable shopping.

By meeting these objectives, Cartopia aims to provide a seamless, efficient, and secure online shopping experience that rivals traditional brick-and-mortar stores while leveraging the advantages of digital convenience and personalization.

## 1.3 MODULE

### Modules of Cartopia Online Shopping System

#### 1. User Management Module

- Home Page Integration:
  - Login and Register Access: Provides easy access to the login and registration functionalities directly from the home page.
- Registration:
  - Input Fields: Username, password as relevant information.
  - Validation: Ensures all required fields are filled out correctly and validates input.
  - Account Creation: Stores user information securely in the database.
- Login:
  - Input Fields: Username and password.
  - Authentication: Verifies user credentials against stored data.
  - Session Management: Initiates a user session upon successful login.
- Logout:
  - Session Termination: Ends the user session securely.

## **2. Product Catalog Module**

- Groceries and Skincare Pages:
  - Explore Events Button: Allows users to explore product categories.
- Product Display:
  - Product List: Shows a list of products within the selected category.
  - Quantity Management: Enables users to add or subtract product quantities.
  - Add to Cart: Allows users to add selected products and quantities to their cart.

## **3. Shopping Cart Module**

- Cart Page:
  - Product Details: Displays products added to the cart with their respective quantities and prices.
  - Quantity Management: Allows users to update the quantity of each product in the cart.
  - Price Calculation: Automatically updates the total cost based on the quantities and prices of items in the cart.
- Payment Process:
  - Payment Options: Provides payment methods
  - Order Confirmation: Confirms the payment and displays a successful transaction message.

## **4. Checkout and Payment Module**

- Order Summary:
  - Review: Shows a summary of the items in the cart, including quantities and total price.
- Payment Gateway Integration:
  - Secure Transaction: Ensures secure processing of payments using integrated payment gateways.
- Order Confirmation:
  - Success Message: Displays a success message upon successful payment.



## 5. Order Management Module

- Order Report:
  - User Details: Includes username.
  - Product Details: Lists purchased products with their quantities and prices.
  - Transaction Date: Records the date as well as time of the transaction.
  - Report Generation: Automatically generates and stores the report for user access.

## 6. Security Module

- Data Encryption:
  - Secure Storage: Encrypts user data and sensitive information.
- Authentication and Authorization:
  - Access Control: Manages user permissions to ensure secure access.
- Session Security:
  - Ensures secured storage of passwords and the user's order history. Therefore, reliable.

By structuring the Cartopia system with these modules, it ensures an organized and efficient shopping experience for users, covering all essential operations from registration and browsing products to managing carts, processing payments, and generating comprehensive reports.

# CHAPTER 2 SURVEY OF TECHNOLOGIES

## 2.1 SOFTWARE DESCRIPTION

### ARCHITECTURE DESCRIPTION

#### **Presentation Layer**

Responsible for the user interface and interaction.

Implements web pages, forms, and interfaces for users.

Utilizes HTML for content structure, CSS for styling, and JavaScript for interactivity.

PHP used for server-side processing and dynamic content generation.

#### **Application Layer**

Manages the business logic and workflow of the system.

Processes user requests, coordinates data flow, and interacts with the database.

Implements routing, request handling, and data manipulation logic.

Primarily consists of PHP scripts running on the server.

#### **Data Access Layer**

Handles interactions with the database.

Executes SQL queries for data retrieval, storage, and manipulation.

Ensures data integrity and security.

Utilizes MySQL as the relational database management system.

#### **Database Layer**

Stores and organizes system data, including username , password, product details .

Manages data persistence and retrieval operations.

Provides a structured storage mechanism for efficient data handling.

Relies on MySQL databases for data storage and management.

## **MODEL-VIEW-CONTROLLER (MVC) DESCRIPTION**

The Model-View-Controller (MVC) architecture is a design pattern that separates an application into three main logical components: the Model, the View, and the Controller. This separation helps in managing complex applications by dividing the responsibilities among different components.

### **MVC Components for Cartopia Online Shopping System**

#### **1. Model:**

- **User Model:**
  - Manages user data, including registration, login credentials, and user profiles.
  - Interacts with the database to store and retrieve user information.
- **Product Model:**
  - Manages product data for groceries and skincare items, including details like name, price, category, and quantity.
  - Handles database operations related to product information.
- **Cart Model:**
  - Manages cart data, including items added to the cart, their quantities, and total price.
  - Performs operations to add, update, and remove items from the cart.
- **Order Model:**
  - Manages order data, including the order summary, payment status, and order history.
  - Generates and stores order reports with username, product details, quantities, prices, and transaction dates.

#### **2. View:**

- **Home Page View:**
  - Displays login and registration options.
  - Provides navigation to other parts of the application.
- **Login Page View:**
  - Contains a form for users to enter their username and password.
- **Register Page View:**
  - Contains a form for new users to create an account by providing necessary details.
- **Groceries and Skincare Pages View:**
  - Displays product categories and "Explore Events" button.
  - Shows a list of products with options to add or subtract quantities and add to cart.
- **Cart Page View:**
  - Displays items in the cart with their quantities and prices.
  - Provides payment options and displays the total cost.

- **Payment Confirmation View:**
    - Displays a success message upon successful payment.
    - Shows the order summary and confirmation details.
3. **Controller:**
- **UserController:**
    - Handles user registration and login requests.
    - Validates user inputs and interacts with the User Model to create or authenticate users.
  - **ProductController:**
    - Manages requests to view products in the groceries and skincare categories.
    - Interacts with the Product Model to fetch and display product information.
  - **CartController:**
    - Manages the shopping cart, including adding and removing products and updating quantities.
    - Interacts with the Cart Model to maintain the current state of the cart.
  - **OrderController:**
    - Handles the checkout process and payment transactions.
    - Interacts with the Order Model to create orders and generate reports.
    - Displays the payment confirmation and order summary.

## 2.2. LANGUAGES

### Languages Used in Cartopia Online Shopping System

#### Backend: PHP

PHP is used for handling server-side operations, providing dynamic content, and interacting with the database. The following pages are created using PHP:

##### 1. Login Page (login.php)

- Description: Allows users to log into their accounts by entering their username and password. Upon successful authentication, users are redirected to the home page or dashboard.

- Functionality:

- Validates user credentials.
- Establishes user sessions.
- Redirects users upon successful login.

##### 2. Register Page (register.php)

- Description: Enables new users to create an account by filling out a registration form with necessary details such as username, password, and email.

- Functionality:

- Captures user information.
- Validates input data.
- Stores new user data in the database.

- Redirects users to the login page upon successful registration.

### 3. Report Page (report.php)

- Description: Generates a report that provides details about user orders, including the username, product names, quantities, prices, and transaction dates.
- Functionality:
  - Retrieves order data from the database.
  - Displays the order report in a tabular format.
  - Ensures only authenticated users can access their respective reports.

### 4. Logout Page (logout.php)

- Description: Logs users out of their accounts, terminating their session and redirecting them to the login page.
- Functionality:
  - Destroys the user session.
  - Redirects to the login page to ensure a secure logout.

## **Frontend: HTML, CSS, and JavaScript**

HTML, CSS, and JavaScript are used to create the client-side interface, ensuring a responsive and interactive user experience. The following pages are created using these technologies:

### 1. Home Page (index.html)

- Description: Serves as the main landing page, providing navigation to various parts of the application such as login, register, and product categories.
- Functionality:
  - Contains links to login, register, and product pages.
  - Provides an overview of the Cartopia platform.

### 2. Cart Page (cart.html)

- Description: Displays the products added to the cart by the user, including their quantities and total price. Users can modify quantities and proceed to checkout.
- Functionality:
  - Shows items in the cart with options to increase or decrease quantity.
  - Updates total price dynamically.
  - Provides a button to proceed to payment.

### 3. Skincare and Grocery Categories Pages (skincare.html, grocery.html)

- Description: : Provides detailed information about a selected product, including its description, price, and reviews. Users can add the product to their cart from this page. Lists products under the respective categories (skincare and grocery). Users can explore products, adjust quantities, and add them to the cart.
- Functionality:
  - Displays products in the selected category.
  - Allows users to adjust product quantities and add items to the cart.
  - Uses JavaScript for interactive features such as adding/removing items.

Integration of Frontend and Backend:

- Form Handling: Forms on the frontend (login, register) are submitted to PHP scripts for processing user data and interacting with the database.
- Session Management: PHP manages user sessions to maintain login states and secure user data.

By leveraging PHP for backend operations and HTML, CSS, and JavaScript for the frontend, Cartopia ensures a robust, interactive, and user-friendly online shopping experience.

# **CHAPTER 3**

## **REQUIREMENTS AND ANALYSIS**

### **3.1 REQUIREMENT SPECIFICATION**

#### **User Requirements for Cartopia**

- 1.As a new user, I want to register an account by providing my username, password, so that I can access personalized features and save my preferences.
- 2.As a registered user, I want to log into my account using my username and password, so that I can access my personalized shopping experience and saved preferences.
3. As a user, I want to browse groceries and skincare products categorized neatly, so that I can easily find the items I am interested in.
4. As a user, I want to view detailed information about a product, including images, description, price so that I can make informed purchasing decisions.
- 5.As a user, I want to add, remove the quantities of products in my cart, so that I can manage the items I plan to purchase.
6. As a user, I want to proceed to checkout with the items in my cart, review my order, and choose a payment method, so that I can complete my purchase securely.
7. As a user, I want to receive a confirmation message after successful payment, so that I know my purchase has been completed.
8. As a user, I want to view my order history, including past purchases with details like product names, quantities, prices, and dates, so that I can keep track of what I have bought.
- 9.As a user, I want to securely log out of my account, so that my personal information and shopping activities are protected.
10. As a user, I want to generate and view a detailed report of my orders, including my username, product details, quantities, prices, and transaction dates, so that I can review my purchases.

## 3.2 HARDWARE AND SOFTWARE REQUIREMENTS

### HARDWARE REQUIREMENTS :

#### 1. Server

- A dedicated server or cloud-based hosting capable of handling the expected traffic and data processing requirements of the system.
- Suggested specifications:
- CPU: Multi-core processor (e.g., Intel Xeon, AMD Ryzen)
- RAM: At least 8GB (16GB recommended)
- Storage: SSD storage for faster data access
- Network: Stable internet connection with sufficient bandwidth

#### 3. Database Server

☐ A separate database server to store user credentials, candidate information, and voting data securely.

#### 4. Client Devices

- ☐ Devices for users to access the system including desktop computers, laptops.
- ☐ Compatible web browsers for accessing the system's web interface.



# **SOFTWARE REQUIREMENTS :**

## **Software Requirements for Cartopia:**

### Development Environment

#### 1. Code Editor:

- HTML, CSS, JavaScript, and PHP files can be created and edited using text editors like Visual Studio Code.

### Backend Development

#### 1. Server Environment:

- XAMPP can provide a local development environment with MySQL, and PHP pre-configured.

#### 2. PHP:

- PHP programming language is used for server-side scripting and backend logic.
- PHP version compatible with the chosen web server .

#### 3. Database Management System:

- MySQL is used as the relational database management system (RDBMS) for storing user data, product information, and order details.
- PHP can be used as a graphical user interface for managing MySQL databases.

### Frontend Development

#### 1. HTML:

- Hypertext Markup Language (HTML) is used for creating the structure and content of web pages.

#### 2. CSS:

- Cascading Style Sheets (CSS) are used for styling the HTML elements and defining the layout and appearance of the web pages.

### 3. JavaScript:

- JavaScript is used for client-side scripting to enhance interactivity and functionality of the web pages.
- jQuery or other JavaScript libraries/frameworks can be utilized for DOM manipulation and AJAX requests.

### Deployment

#### I. Domain Name:

- a. A domain name is needed to provide users with a unique and memorable web address to access the Cartopia application.

#### II. SSL Certificate:

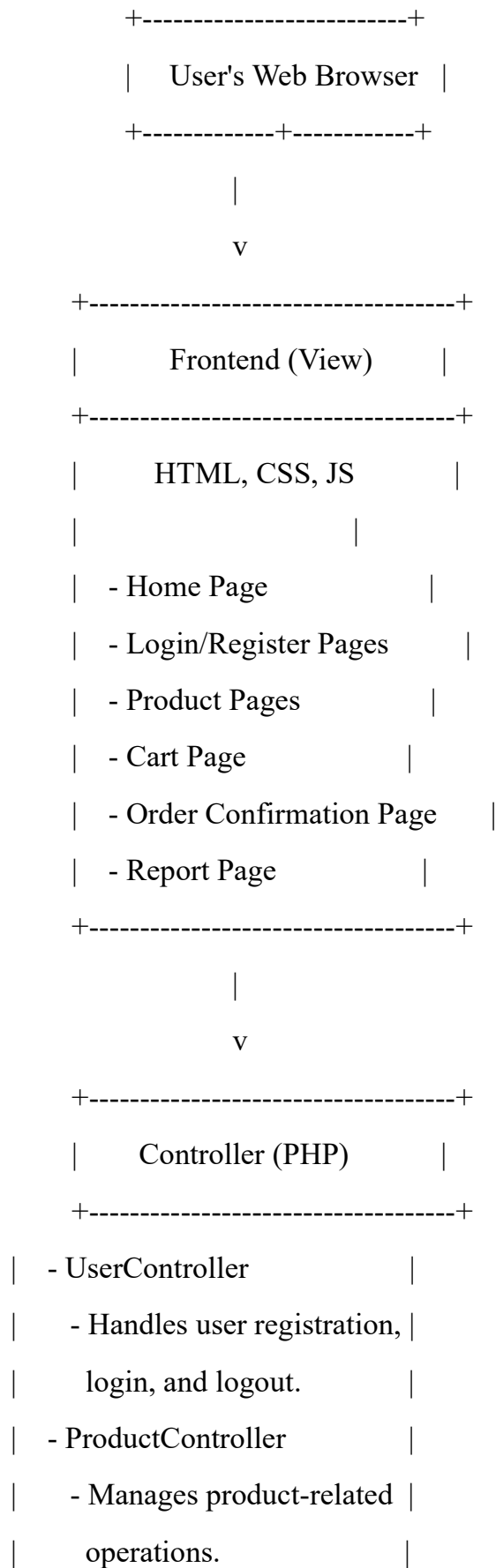
- a. Secure Socket Layer (SSL) certificate is required to enable HTTPS protocol and ensure secure communication between the web server and clients.

#### III. Server Configuration:

- a. Configuration of web server settings (e.g., PHP settings, file permissions) for optimal performance and security.

By fulfilling these software requirements, developers can effectively set up a development environment, build the backend and frontend components of Cartopia, and deploy the application for users to access and enjoy the online shopping experience.

### 3.3 ARCHITECTURE DIAGRAM



- CartController
- Manages cart operations.
- OrderController
- Handles checkout process
and order management.

+-----+

|

v

+-----+

Backend (Model)
-----------------

+-----+

PHP Scripts
-------------

- User Model
--------------

- Product Model
-----------------

- Cart Model
--------------

- Order Model
---------------

- Database Interaction
------------------------

+-----+

|

v

+-----+

Database (MySQL)
------------------

+-----+

User Table
------------

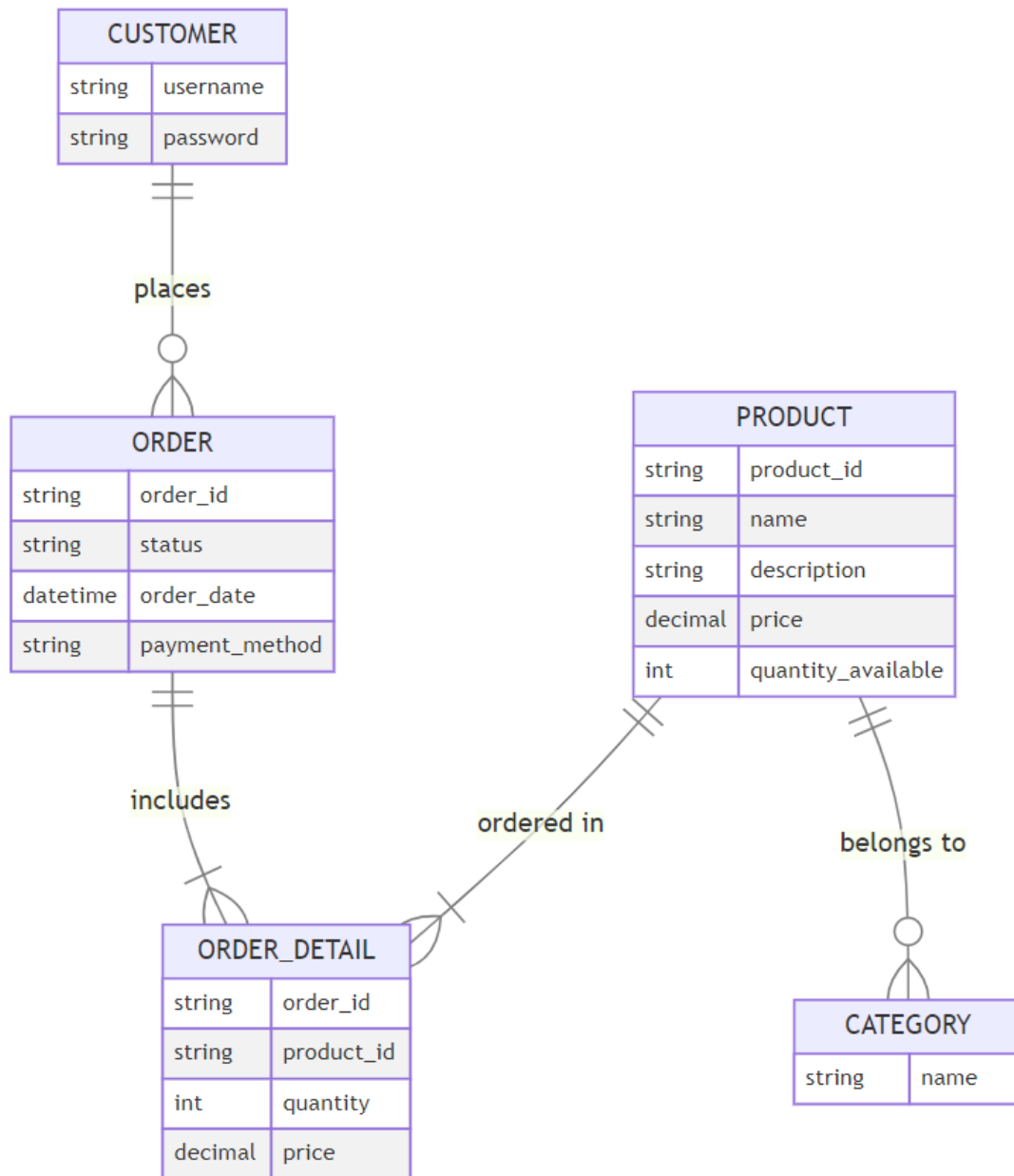
Product Table
---------------

Cart Table
------------

Order Table
-------------

+-----+

### 3.4 ER DIAGRAM



In Cartopia, a fictional online shopping system, several entities interact to facilitate the shopping experience. Let's describe the relationships, entities, and their cardinality and participation:

## Entities:

### 1. User:

- Represents individuals who interact with the system by registering, logging in, and making purchases.
- Attributes: username (primary key), password.

### 2. Product:

- Represents the items available for purchase on the platform.
- Attributes: product\_id (primary key), name, description, price, quantity.

### 3. Cart:

- Represents the shopping cart where users can add products before checkout.
- Attributes: cart\_id (primary key), user\_id (foreign key referencing User), product\_id (foreign key referencing Product), quantity.

### 4. Order:

- Represents a purchase made by a user.
- Attributes: order\_id (primary key), user\_id (foreign key referencing User), order\_date, payment.

### 5. Order Detail:

- Represents individual items within an order.
- Attributes: order\_detail\_id (primary key), order\_id (foreign key referencing Order), product\_id (foreign key referencing Product), quantity, price.

## Relationships:

### 1. User - Cart (owns):

- One user can have multiple carts (e.g., active and saved carts).

- Cardinality: One User can have Zero or Many Carts (1:M).
- Participation: Mandatory on the User side (Every User must have at least one cart).

## 2. User - Order (places):

- One user can place multiple orders.
- Cardinality: One User can have Zero or Many Orders (1:M).
- Participation: Mandatory on the User side (Every User must place at least one order).

## 3. Product - Cart (added to):

- One product can be added to multiple carts.
- Cardinality: One Product can be in Zero or Many Carts (1:M).
- Participation: Optional on both sides (A product may or may not be added to a cart, and a cart may or may not contain products).

## 4. Product - Order Detail (ordered in):

- One product can be included in multiple order details (across different orders).
- Cardinality: One Product can be in Zero or Many Order Details (1:M).
- Participation: Optional on both sides (A product may or may not be included in an order detail, and an order detail may or may not contain products).

## 5. Order - Order Detail (includes):

- One order can include multiple order details.
- Cardinality: One Order can have Zero or Many Order Details (1:M).
- Participation: Mandatory on the Order side (Every order must include at least one order detail).

These relationships and entities form the backbone of Cartopia's data model, enabling users to browse, select, purchase, and manage products effectively.

## 3.5 NORMALISATION

```
mysql> desc order_cust;
```

Field	Type	Null	Key	Default	Extra
ORDER_ID	int(11)	NO	PRI	NULL	auto_increment
PRODUCT_NAME	varchar(255)	NO		NULL	
QUANTITY	int(11)	NO		NULL	
TOTAL_PRICE	decimal(10,2)	NO		NULL	
order_date	timestamp	NO		current_timestamp()	
id	int(11)	YES	MUL	NULL	
username	varchar(50)	YES		NULL	

```
mysql> desc accounts;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(50)	NO		NULL	
password	varchar(255)	NO		NULL	

Normalization is the process of organizing data in a database to reduce redundancy and dependency, Thereby improving data integrity and efficiency.

Normalization ensures that the database schema is structured optimally to minimize data redundancy and anomalies.

### Normalization Levels:

The database schema for the Online Shopping System (Cartopia) Project :

Third Normal Form (3NF) to ensure data integrity and efficiency. Each normalization level addresses

specific types of data redundancies and dependencies, as outlined below:

#### 1. First Normal Form (1NF):

- ☐ Eliminate repeating groups within tables.
- ☐ Ensure each attribute contains atomic values.
- ☐ Example: Splitting a table with repeating sensor readings into separate tables.

#### 2. Second Normal Form (2NF):

- ☐ Meet the requirements of 1NF.
- ☐ Remove partial dependencies by creating separate tables for related attributes.
- ☐ Example: Ensuring all non-key attributes are fully functionally dependent on the primary key.

#### 3. Third Normal Form (3NF):

- ☐ Meet the requirements of 2NF.
- ☐ Eliminate transitive dependencies by moving attributes dependent on non-key



attributes to separate tables.

☐ Example: Ensuring that attributes like vibration levels are not dependent on non-key attributes like humidity.

### **Normalization Process:**

#### **1. Identify Entities and Attributes:**

☐ Identify the entities and attributes within the database schema.

#### **2. Apply First Normal Form (1NF):**

☐ Ensure that each table contains only atomic values, and there are no repeating groups of attributes.

☐ Split tables with repeating groups into separate tables to eliminate data redundancy.

#### **3. Apply Second Normal Form (2NF):**

☐ Ensure that each non-key attribute is fully functionally dependent on the primary key.

☐ Create separate tables for attributes that are not fully dependent on the primary key to remove partial dependencies.

#### **4. Apply Third Normal Form (3NF):**

☐ Eliminate transitive dependencies by moving attributes dependent on non-key attributes to separate tables.

☐ Ensure that all attributes are directly dependent on the primary key, and there are no indirect dependencies.

### **Benefits of Normalization:**

#### **1. Data Integrity:**

☐ Normalization reduces data redundancy and dependency, minimizing the risk of data anomalies such as insertion, update, and deletion anomalies.

#### **2. Efficiency:**

☐ Normalized databases are more efficient in terms of storage space and query performance, as they require fewer resources to maintain and query data.

#### **3. Scalability:**

☐ Normalized databases are easier to scale and maintain as they are structured in a way that accommodates changes and additions to the database schema.

By following the normalization process and ensuring the database schema meets the requirements of

at least the Third Normal Form, the Online Shopping System (Cartopia) Project achieves a well-structured

and optimized database design, enhancing data integrity, efficiency, and scalability.

## First Normal Form (1NF)

### ACCOUNTS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ACCOUNTS	ID	NUMBER	22	-	0	1	-	-	-
	USERNAME	VARCHAR2	50	-	-	-	✓	-	-

### ORDER\_TOTALS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDER_TOTALS	ORDER_ID	NUMBER	22	-	0	1	-	-	-
	TOTAL_PRICE	NUMBER	-	10	2	-	✓	-	-
	ORDER_DATE	DATE	7	-	-	2	-	-	-

### ORDER\_ITEMS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDER_ITEMS	ORDER_ID	NUMBER	22	-	0	1	-	-	-
	PRODUCT_NAME	VARCHAR2	50	-	-	2	-	-	-
	QUANTITY	NUMBER	22	-	0	-	✓	-	-

### ORDERS

Table	Column	Data Type	Length	Precision	Scale	Primary Key
ORDERS	ORDER_ID	NUMBER	22	-	0	1

## SECOND NORMALISATION

### ACCOUNTS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default
ACCOUNTS	ID	NUMBER	22	-	0	1	-	-
	USERNAME	VARCHAR2	50	-	-	-	-	-
	PASSWORD	VARCHAR2	50	-	-	-	-	-

### ORDERS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default
ORDERS	ORDER_ID	NUMBER	22	-	0	1	-	-
	ORDER_DATE	TIMESTAMP(6)	11	-	6	-	✓	CURRENT_TIMESTAMP
	ID	NUMBER	22	-	0	-	✓	-

## ORDER\_ITEMS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default
ORDER_ITEMS	ORDER_ID	NUMBER	22	-	0	1	-	-
	PRODUCT_NAME	VARCHAR2	50	-	-	2	-	-
	QUANTITY	NUMBER	22	-	0	-	✓	-

## ORDER\_TOTALS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default
ORDER_TOTALS	ORDER_ID	NUMBER	22	-	0	1	-	-
	TOTAL_PRICE	NUMBER	-	10	2	-	✓	-

## Ch 4. PROGRAM CODE

### 4.1. CODE DETAILS AND CODE EFFICIENCY

#### HOME PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home page</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH"
crossorigin="anonymous">
  <style>
    body {
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
      background: linear-gradient(to right, orange, silver);
      font-family: Arial, sans-serif; /* Professional font */
      color: 333; /* Professional color */
    }
    .header {
      position: absolute;
      top: 20px;
      width: 100%;
      text-align: center;
    }
    .icon {
      position: absolute;
      top: 20px;
      left: 20px;
    }
    .icon img {
      height: 80px; /* Adjust as needed */
    }
    .cart-icon {
      position: absolute;
      top: 20px;
      right: 20px;
```

```

    }
    .cart-icon img {
        height: 50px; /* Adjust as needed */
        cursor: pointer;
    }
    .welcome-text h1 {
        margin: 10px 0;
        font-size: 2em; /* Adjust as needed */
    }
    .slogan p {
        margin: 5px 0;
        font-size: 1.2em; /* Adjust as needed */
    }
    .button-container {
        text-align: center;
    }
    .button {
        display: inline-block;
        margin: 10px;
        padding: 15px 30px;
        font-size: 18px;
        color: fff;
        background-color: ff8c00; /* Darker shade of orange */
        border: none;
        border-radius: 5px;
        cursor: pointer;
        transition: background-color 0.3s;
    }
    .button:hover {
        background-color: e07b00; /* Even darker shade of orange for hover effect */
    }
</style>
</head>
<body>
    <div class="icon">
        
    </div>

    <div class="header">
        <div class="welcome-text">
            <h1>WELCOME TO CARTOPIA</h1>
        </div>
        <div class="slogan">
            <p>Shop Smarter Not Harder</p>
        </div>
    </div>

```

```

<div class="button-container">
    <button class="button" onclick="window.location.href='login.php'">
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-cart-fill" viewBox="0 0 16 16">

            </svg>
            LOGIN

        </button>

        <button class="button" onclick="window.location.href='register.php'">
            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-cart-fill" viewBox="0 0 16 16">

                </svg>

            </button>

            <button class="button" style="color: fff;">
                <a href="/cart1.html">
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-cart-fill" viewBox="0 0 16 16">
                        <path d="M0 1.5A.5.5 0 0 1 .5 1H2a.5.5 0 0 1 .485 3.79L2.89 3H14.5a.5.5 0 0 1
.491 5.921-1.5 8A.5.5 0 0 1 13 12H4a.5.5 0 0 1 -.491-.408L2.01 3.607 1.61 2H.5a.5.5 0 0 1-.5-
.5M5 12a2 2 0 1 0 0 4 2 2 0 0 0 0-4m7 0a2 2 0 1 0 0 4 2 2 0 0 0 0-4m-7 1a1 1 0 1 1 0 2 1 1 0
0 1 0-2m7 0a1 1 0 1 1 0 2 1 1 0 0 1 0-2"/>
                    </svg>
                </a>
            </button>
        </div>
</body>
</html>

```

## **REGISTER PAGE**

```

<?php
// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "ecommerce";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

```

```

}

// Process registration form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $user_name = $_POST['username'];
    $password = $_POST['password'];

    // Encrypt the password using bcrypt
    $hashed_password = password_hash($password, PASSWORD_BCRYPT);

    // Prepare and execute SQL statement to insert user into database
    $stmt = $conn->prepare("INSERT INTO accounts (username, password) VALUES (?, ?)");
    $stmt->bind_param("ss", $user_name, $hashed_password);

    // Execute the statement
    if ($stmt->execute()) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $stmt->error;
    }
    $stmt->close();
}

$conn->close();
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration</title>
</head>
<body>
    <h2>Register</h2>
    <form action="" method="POST"> <!-- Corrected form action and method -->
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <button type="submit">Register</button>
    </form>
</body>
</html>

```

## **LOGIN PAGE**

```
<?php
// Start the session
session_start();

// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "ecommerce";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $user_name = $_POST['username'];
    $password = $_POST['password'];

    // Retrieve the hashed password and username from the database
    $stmt = $conn->prepare("SELECT password FROM accounts WHERE username = ?");
    $stmt->bind_param("s", $user_name);
    $stmt->execute();
    $stmt->bind_result($stored_hash);
    $stmt->fetch();
    $stmt->close();

    // Verify the password
    if (password_verify($password, $stored_hash)) {
        // Set username in the session
        $_SESSION['username'] = $user_name;

        // Redirect to the desired page
        header("Location: index.html");
        exit();
    } else {
        $error_message = "Invalid username or password";
    }
}

$conn->close();
?>
```



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
</head>
<body>
  <h2>Login</h2>
  <?php if(isset($error_message)) { ?>
    <p><?php echo $error_message; ?></p>
  <?php } ?>
  <form action="login.php" method="POST">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <button type="submit">LOGIN</button>
  </form>
</body>
</html>
```

## **CATEGORY - MAIN PAGE**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" integrity="sha512-
iecdLmaskl7CVkqkXNq/ZH/XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/E0iPtmFIB46Zmdt
Ac9eNBvH0H/ZpiBw==" crossorigin="anonymous" referrerpolicy="no-referrer" />

  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      background: eaeaea;
      overflow: hidden;
    }
  </style>
```

```
.container {  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  transform: translate(-50%, -50%);  
  width: 1000px;  
  height: 600px;  
  background: f5f5f5;  
  box-shadow: 0 30px 50px dbdbdb;  
}
```

```
.container .slide .item {  
  width: 200px;  
  height: 300px;  
  position: absolute;  
  top: 50%;  
  transform: translate(0, -50%);  
  border-radius: 20px;  
  box-shadow: 0 30px 50px 505050;  
  background-position: 50% 50%;  
  background-size: cover;  
  display: inline-block;  
  transition: 0.5s;  
}
```

```
.slide .item:nth-child(1),  
.slide .item:nth-child(2) {  
  top: 0;  
  left: 0;  
  transform: translate(0, 0);  
  border-radius: 0;  
  width: 100%;  
  height: 100%;  
}
```

```
.slide .item:nth-child(3) {  
  left: 50%;  
}
```

```
.slide .item:nth-child(4) {  
  left: calc(50% + 220px);  
}
```

```
.slide .item:nth-child(5) {  
  left: calc(50% + 440px);  
}
```

```

}

/* here n = 0, 1, 2, 3,... */
.slide .item:nth-child(n + 6) {
    left: calc(50% + 660px);
    opacity: 0;
}

.item .content {
    position: absolute;
    top: 50%;
    left: 100px;
    width: 300px;
    text-align: left;
    color: eee;
    transform: translate(0, -50%);
    font-family: system-ui;
    display: none;
}

.slide .item:nth-child(2) .content {
    display: block;
}

.content .name {
    font-size: 40px;
    text-transform: uppercase;
    font-weight: bold;
    opacity: 0;
    animation: animate 1s ease-in-out 1 forwards;
}

.content .des {
    margin-top: 10px;
    margin-bottom: 20px;
    opacity: 0;
    animation: animate 1s ease-in-out 0.3s 1 forwards;
}

.content button {
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    opacity: 0;
    animation: animate 1s ease-in-out 0.6s 1 forwards;
}

```

```
@keyframes animate {
  from {
    opacity: 0;
    transform: translate(0, 100px);
    filter: blur(33px);
  }
  to {
    opacity: 1;
    transform: translate(0);
    filter: blur(0);
  }
}
```

```
.button {
  width: 100%;
  text-align: center;
  position: absolute;
  bottom: 20px;
}
```

```
.button {
  width: 40px;
  height: 35px;
  border-radius: 8px;
  border: none;
  cursor: pointer;
  margin: 0 5px;
  border: 1px solid 000;
  transition: 0.3s;
}
```

```
.button button:hover {
  background: ababab;
  color: fff;
}
```

```
.image-btn {
  background-color: ffffff;
  color: white;
  border: none;
  padding: 10px 20px;
  font-size: 16px;
  cursor: pointer;
  border-radius: 5px;
}
```



```

        </div>
    </div>
</div>
<div class="button">
    <button class="prev"><i class="fa-solid fa-arrow-left"></i></button>
    <button class="next"><i class="fa-solid fa-arrow-right"></i></button>
</div>
</div>

<script>
    let next = document.querySelector('.next')
    let prev = document.querySelector('.prev')

    next.addEventListener('click', function(){
        let items = document.querySelectorAll('.item')
        document.querySelector('.slide').appendChild(items[0])
    })

    prev.addEventListener('click', function(){
        let items = document.querySelectorAll('.item')
        document.querySelector('.slide').prepend(items[items.length - 1])
    })
</script>
</body>
</html>

```

## **CATEGORY 1 - GROCERIES SECTION:**

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>GROCERIES</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@6.4.0/css/all.min.css" />
    <style>
        body {
            margin: 0;
            padding: 0;
            background: linear-gradient(135deg, #e30ee7, #ffffff);
        }
        .icon-container {
            position: fixed;

```

```

    top: 20px;
    right: 20px;
    z-index: 9999;
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 10px;
}
.cart-icon, .report-btn, .logout-btn {
    width: 150px;
    height: 40px;
    background-color: 007bff;
    border-radius: 20px;
    display: flex;
    justify-content: center;
    align-items: center;
    color: fff;
    font-size: 18px;
    text-align: center;
    text-decoration: none;
}
.cart-icon a, .report-btn a, .logout-btn a {
    color: white;
    text-decoration: none;
}
.cart-icon:hover, .report-btn:hover, .logout-btn:hover {
    background-color: 0056b3;
    cursor: pointer;
}
.product-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    color: fff;
}
.product-box {
    position: relative;
    width: 200px;
    margin: 10px;
    border: 1px solid ccc;
    border-radius: 15px;
    overflow: hidden;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.product-image {
    width: 100%;

```

```

        height: 70%;
        overflow: hidden;
    }
    .product-image img {
        width: 100%;
        height: 100%;
        object-fit: cover;
    }
    .product-description {
        padding: 10px;
        background-color: fff;
        color: 333;
    }
    .options {
        display: flex;
        justify-content: space-between;
        align-items: center;
        padding: 5px;
        background-color: fff;
        border-bottom-left-radius: 15px;
        border-bottom-right-radius: 15px;
    }
    .options input[type="number"] {
        width: 50px;
    }
    .add-btn, .delete-btn {
        background-color: 007bff;
        color: fff;
        border: none;
        padding: 5px 10px;
        cursor: pointer;
    }
    .add-btn:hover, .delete-btn:hover {
        background-color: 0056b3;
    }
</style>
</head>
<body>
    <div class="icon-container">
        <div class="cart-icon">
            <a href="cart1.html">
                <i class="fas fa-shopping-cart"></i> Cart
            </a>
        </div>
        <div class="report-btn">
            <a href="report.php">

```



```

        <i class="fas fa-file-alt"></i> Report
    </a>
</div>
<div class="logout-btn">
    <a href="logout.php">
        <i class="fas fa-sign-out-alt"></i> Logout
    </a>
</div>
</div>
<div class="product-container">
    <div class="product-box" data-product-id="1" data-product-name="CABBAGE" data-
product-price="15">
        <div class="product-image">
            
        </div>
        <p class="product-description">CABBAGE Price: Rs.15/kg</p>
        <div class="options">
            <input type="number" value="1" min="0">
            <button class="add-btn">+</button>
            <button class="delete-btn">-</button>
        </div>
    </div>
    <div class="product-box" data-product-id="2" data-product-name="BASMATI RICE"
data-product-price="170">
        <div class="product-image">
            
        </div>
        <p class="product-description">BASMATI RICE Price: Rs.170</p>
        <div class="options">
            <input type="number" value="1" min="0">
            <button class="add-btn">+</button>
            <button class="delete-btn">-</button>
        </div>
    </div>
    <div class="product-box" data-product-id="3" data-product-name="FORTUNE
SUNFLOWER OIL" data-product-price="150">
        <div class="product-image">
            
        </div>
        <p class="product-description">FORTUNE SUNFLOWER OIL Price: Rs.150</p>
        <div class="options">
            <input type="number" value="1" min="0">
            <button class="add-btn">+</button>
            <button class="delete-btn">-</button>
        </div>
    </div>
</div>

```

</div>

<script>

```
document.addEventListener('DOMContentLoaded', function () {  
  const addButtons = document.querySelectorAll('.add-btn');  
  const deleteButtons = document.querySelectorAll('.delete-btn');  
  const quantityInputs = document.querySelectorAll('.options input[type="number"]');
```

```
  
  function updateCart(productId, productName, productPrice, quantity) {  
    let cart = JSON.parse(localStorage.getItem('cart')) || [];  
    const productIndex = cart.findIndex(item => item.productId === productId);
```

```
  
    console.log('Updating cart for product ID:', productId);  
    console.log('Product name:', productName);  
    console.log('Product price:', productPrice);  
    console.log('Quantity:', quantity);
```

```
  
    if (productIndex !== -1) {  
      if (quantity > 0) {  
        cart[productIndex].quantity = quantity;  
      } else {  
        cart.splice(productIndex, 1);  
      }  
    } else if (quantity > 0) {  
      cart.push({ productId, productName, productPrice, quantity });  
    }  
  }
```

```
  
  localStorage.setItem('cart', JSON.stringify(cart));  
  console.log('Updated cart:', cart);  
}
```

```
  
function updateQuantityInput(index, increment = true) {  
  const productBox = document.querySelectorAll('.product-box')[index];  
  const productId = parseInt(productBox.dataset.productId);  
  const productName = productBox.dataset.productName;  
  const productPrice = parseInt(productBox.dataset.productPrice);  
  const input = quantityInputs[index];  
  const currentQuantity = parseInt(input.value);
```

```
  
  if (increment) {  
    input.value = currentQuantity + 1;  
  } else {  
    input.value = Math.max(currentQuantity - 1, 0);  
  }  
}
```

```
  
  updateCart(productId, productName, productPrice, parseInt(input.value));  
}
```

```

    }

    addButtons.forEach((button, index) => {
        button.addEventListener('click', function () {
            updateQuantityInput(index, true);
        });
    });

    deleteButtons.forEach((button, index) => {
        button.addEventListener('click', function () {
            updateQuantityInput(index, false);
        });
    });

    quantityInputs.forEach((input, index) => {
        input.addEventListener('change', function () {
            if (input.value < 0) input.value = 0;
            updateQuantityInput(index, false);
        });
    });

    function initializeQuantities() {
        const cart = JSON.parse(localStorage.getItem('cart')) || [];
        document.querySelectorAll('.product-box').forEach((productBox, index) => {
            const productId = parseInt(productBox.dataset.productId);
            const cartItem = cart.find(item => item.productId === productId);
            if (cartItem) {
                quantityInputs[index].value = cartItem.quantity;
            } else {
                quantityInputs[index].value = 0;
            }
        });
    }

    initializeQuantities();
});
</script>
</body>
</html>

```

## **CATEGORY 2 - SKINCARE SECTION:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>SKINCARE</title>
  <!-- <link rel="stylesheet" href="style.css"> -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" />
  <style>
    body {
      margin: 0;
      padding: 0;
      background: linear-gradient(135deg, e30ee7, ffffff);
    }
    .icon-container {
      position: fixed;
      top: 20px;
      right: 20px;
      z-index: 9999;
      display: flex;
      flex-direction: column;
      align-items: center;
      gap: 10px;
    }
    .cart-icon, .report-btn, .logout-btn {
      width: 150px;
      height: 40px;
      background-color: 007bff;
      border-radius: 20px;
      display: flex;
      justify-content: center;
      align-items: center;
      color: fff;
      font-size: 18px;
      text-align: center;
      text-decoration: none;
    }
    .cart-icon a, .report-btn a, .logout-btn a {
      color: white;
      text-decoration: none;
    }
  </style>
</head>
<body>
```

```
.cart-icon:hover, .report-btn:hover, .logout-btn:hover {
    background-color: 0056b3;
    cursor: pointer;
}

.product-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    color: fff;
}

.product-box {
    position: relative;
    width: 200px;
    margin: 10px;
    border: 1px solid ccc;
    border-radius: 15px;
    overflow: hidden;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.product-image {
    width: 100%;
    height: 70%;
    overflow: hidden;
}

.product-image img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.product-description {
    padding: 10px;
    background-color: fff;
    color: 333;
}

.options {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 5px;
    background-color: fff;
    border-bottom-left-radius: 15px;
    border-bottom-right-radius: 15px;
}

.options input[type="number"] {
    width: 50px;
}
```

```

.add-btn, .delete-btn {
    background-color: 007bff;
    color: fff;
    border: none;
    padding: 5px 10px;
    cursor: pointer;
}
.add-btn:hover, .delete-btn:hover {
    background-color: 0056b3;
}
</style>
</head>
<body>
<div class="icon-container">
<div class="cart-icon">
<a href="cart1.html">
<i class="fas fa-shopping-cart"></i> Cart
</a>
</div>
<div class="report-btn">
<a href="report.php">
<i class="fas fa-file-alt"></i> Report
</a>
</div>
<div class="logout-btn">
<a href="logout.php">
<i class="fas fa-sign-out-alt"></i> Logout
</a>
</div>
</div>

<div class="product-container">
<div class="product-box" data-product-id="4" data-product-name="SHAMPOO" data-
product-price="300">
<div class="product-image">

</div>
<p class="product-description">SERUM Rs.300</p>
<div class="options">
<input type="number" value="1" min="0">
<button class="add-btn">+</button>
<button class="delete-btn">-</button>
</div>
</div>
<div class="product-box" data-product-id="5" data-product-name="SKIN CREAM"
data-product-price="150">

```

```

<div class="product-image">
  
</div>
<p class="product-description">SUNSCREEN Price: Rs.150</p>
<div class="options">
  <input type="number" value="1" min="0">
  <button class="add-btn">+</button>
  <button class="delete-btn">-</button>
</div>
</div>
<div class="product-box" data-product-id="6" data-product-name="LOTION" data-
product-price="200">
  <div class="product-image">
    
  </div>
  <p class="product-description">LOTION Price: Rs.200</p>
  <div class="options">
    <input type="number" value="1" min="0">
    <button class="add-btn">+</button>
    <button class="delete-btn">-</button>
  </div>
</div>
</div>
<script>
document.addEventListener('DOMContentLoaded', function () {
  const addButtons = document.querySelectorAll('.add-btn');
  const deleteButtons = document.querySelectorAll('.delete-btn');
  const quantityInputs = document.querySelectorAll('.options input[type="number"]');

  function updateCart(productId, productName, productPrice, quantity) {
    let cart = JSON.parse(localStorage.getItem('cart')) || [];
    const productIndex = cart.findIndex(item => item.productId === productId);

    console.log('Updating cart for product ID:', productId);
    console.log('Product name:', productName);
    console.log('Product price:', productPrice);
    console.log('Quantity:', quantity);

    if (productIndex !== -1) {
      if (quantity > 0) {
        cart[productIndex].quantity = quantity;
      } else {
        cart.splice(productIndex, 1);
      }
    } else if (quantity > 0) {

```

```

    cart.push({ productId, productName, productPrice, quantity });
  }

  localStorage.setItem('cart', JSON.stringify(cart));
  console.log('Updated cart:', cart);
}

function updateQuantityInput(index, increment = true) {
  const productBox = document.querySelectorAll('.product-box')[index];
  const productId = parseInt(productBox.dataset.productId);
  const productName = productBox.dataset.productName;
  const productPrice = parseInt(productBox.dataset.productPrice);
  const input = quantityInputs[index];
  const currentQuantity = parseInt(input.value);

  if (increment) {
    input.value = currentQuantity + 1;
  } else {
    input.value = Math.max(currentQuantity - 1, 0);
  }

  updateCart(productId, productName, productPrice, parseInt(input.value));
}

addButtons.forEach((button, index) => {
  button.addEventListener('click', function () {
    updateQuantityInput(index, true);
  });
});

deleteButtons.forEach((button, index) => {
  button.addEventListener('click', function () {
    updateQuantityInput(index, false);
  });
});

quantityInputs.forEach((input, index) => {
  input.addEventListener('change', function () {
    if (input.value < 0) input.value = 0;
    updateQuantityInput(index, false);
  });
});

function initializeQuantities() {
  const cart = JSON.parse(localStorage.getItem('cart')) || [];
  document.querySelectorAll('.product-box').forEach((productBox, index) => {

```



```

        const productId = parseInt(productBox.dataset.productId);
        const cartItem = cart.find(item => item.productId === productId);
        if (cartItem) {
            quantityInputs[index].value = cartItem.quantity;
        } else {
            quantityInputs[index].value = 0;
        }
    });
}

initializeQuantities();
});
</script>
</body>
</html>

```

### **SHOPPING CART PAGE:**

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cart</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            margin: 0;
            padding: 0;
            background: linear-gradient(135deg, e30ee7, ffffff);
        }
        .cart-container {
            max-width: 600px;
            margin: 50px auto;
            background: fff;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        }
        .cart-item {
            display: flex;
            align-items: center;
            justify-content: space-between;
            margin-bottom: 20px;
            padding-bottom: 10px;
        }
    </style>

```

```
border-bottom: 1px solid ccc;
}
.cart-item img {
  width: 100px;
  height: 100px;
  object-fit: cover;
  border-radius: 10px;
}
.cart-item-details {
  flex: 1;
  margin-left: 20px;
}
.cart-item-actions {
  display: flex;
  flex-direction: column;
  align-items: center;
}
.cart-item-actions input[type="number"] {
  width: 60px;
  margin-bottom: 10px;
  text-align: center;
}
.remove-btn {
  background-color: ff4d4d;
  color: fff;
  border: none;
  padding: 5px 10px;
  cursor: pointer;
}
.remove-btn:hover {
  background-color: ff1a1a;
}
.total-cost {
  text-align: right;
  font-size: 1.2em;
  font-weight: bold;
}
.pay-now-btn {
  display: block;
  width: 100%;
  background-color: 28a745;
  color: fff;
  border: none;
  padding: 10px;
  font-size: 1.2em;
  cursor: pointer;
}
```

```

        text-align: center;
        border-radius: 5px;
        margin-top: 20px;
    }
    .pay-now-btn:hover {
        background-color: #218838;
    }
    .payment-success-message {
        text-align: center;
        font-size: 1.2em;
        color: #28a745;
        display: none;
    }
</style>
</head>
<body>
    <div class="cart-container">
        <h3 class="text-center">Shopping Cart</h3>
        <div id="cart-items"></div>
        <div class="cart-total">
            <span>Total:</span>
            <span id="total-amount">0</span>
        </div>
        <!-- Added Pay Now Button and Payment Success Message -->
        <button id="pay-now-btn" class="pay-now-btn">Pay Now</button>
        <div id="payment-success-message" class="payment-success-message">Payment
Successful!</div>
    </div>

    <script>
        document.addEventListener('DOMContentLoaded', function () {
            const cartItemsContainer = document.getElementById('cart-items');
            const totalAmountElement = document.getElementById('total-amount');
            const payNowButton = document.getElementById('pay-now-btn'); // Correctly
referenced the pay now button
            const paymentSuccessMessage = document.getElementById('payment-success-
message');

            // Your JavaScript code
            function loadCartItems() {
                const cart = JSON.parse(localStorage.getItem('cart')) || [];
                cartItemsContainer.innerHTML = "";
                let totalAmount = 0;

                cart.forEach(item => {
                    const itemTotal = item.productPrice * item.quantity;

```

```

totalAmount += itemTotal;

const cartItem = document.createElement('div');
cartItem.classList.add('cart-item');

cartItem.innerHTML = `
  <div class="item-info">
    <h5>${item.productName}</h5>
    <p>Price: Rs.${item.productPrice}</p>
  </div>
  <div class="item-quantity">
    <input type="number" value="${item.quantity}" min="0" data-product-
id="${item.productId}">
    <span>Rs.${itemTotal}</span>
  </div>
`;

cartItemsContainer.appendChild(cartItem);
});

totalAmountElement.textContent = Rs.${totalAmount};
}

function updateCartItem(productId, quantity) {
  let cart = JSON.parse(localStorage.getItem('cart')) || [];
  const productIndex = cart.findIndex(item => item.productId === productId);

  if (productIndex !== -1) {
    if (quantity > 0) {
      cart[productIndex].quantity = quantity;
    } else {
      cart.splice(productIndex, 1);
    }

    localStorage.setItem('cart', JSON.stringify(cart));
    loadCartItems();
  }
}

cartItemsContainer.addEventListener('change', function (event) {
  if (event.target.tagName === 'INPUT' && event.target.type === 'number') {
    const productId = event.target.dataset.productId;
    const quantity = parseInt(event.target.value);

    if (quantity >= 0) {
      updateCartItem(productId, quantity);
    }
  }
});

```

```

    }
  }
});

payNowButton.addEventListener('click', function () {
  const cart = JSON.parse(localStorage.getItem('cart')) || [];
  if (cart.length > 0) {
    fetch('save_order.php', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ items: cart })
    })
    .then(response => response.text())
    .then(data => {
      console.log('Server response:', data);
      paymentSuccessMessage.style.display = 'block';
      localStorage.removeItem('cart'); // Clear cart after payment
      loadCartItems();
      alert('Payment successful');
      window.location.href = 'save_order.php';
    })

    .catch(error => console.error('Error:', error));
  }
});

loadCartItems();
});
</script>
</body>
</html>

```

## **REPORT PAGE:**

```

<?php
// Start the session
session_start();

// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "ecommerce";

```

```

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Calculate the date range for the current week (from Sunday to Saturday)
$start_date = date('Y-m-d', strtotime('last sunday'));
$end_date = date('Y-m-d', strtotime('next saturday'));

// Prepare the SQL statement
$stmt = $conn->prepare("
    SELECT a.username, o.PRODUCT_NAME, o.QUANTITY, o.TOTAL_PRICE,
    o.order_date
    FROM order_cust o
    JOIN accounts a ON a.username = o.user_name
    WHERE DATE(o.order_date) BETWEEN ? AND ?
");
$stmt->bind_param("ss", $start_date, $end_date);
$stmt->execute();
$result = $stmt->get_result();

```

```

$report_data = [];
while ($row = $result->fetch_assoc()) {
    $report_data[] = $row;
}

```

```

$stmt->close();
$conn->close();
?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Weekly Report</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
    <div class="container mt-5">
        <h2>Weekly Report</h2>
        <table class="table table-bordered">
            <thead>

```

```

        <tr>
            <th>Username</th>
            <th>Product Name</th>
            <th>Quantity</th>
            <th>Total Price</th>
            <th>Order Date</th>
        </tr>
    </thead>
    <tbody>
        <?php foreach ($report_data as $row): ?>
            <tr>
                <td><?php echo htmlspecialchars($row['username']); ?></td>
                <td><?php echo htmlspecialchars($row['PRODUCT_NAME']); ?></td>
                <td><?php echo htmlspecialchars($row['QUANTITY']); ?></td>
                <td><?php echo htmlspecialchars($row['TOTAL_PRICE']); ?></td>
                <td><?php echo htmlspecialchars($row['order_date']); ?></td>
            </tr>
        <?php endforeach; ?>
    </tbody>
</table>
</div>
</body>
</html>

```

## **LOGOUT PAGE:**

```

<?php
session_start();

if (isset($_SESSION['username'])) {
    // Unset all session variables
    session_unset();
    // Destroy the session
    session_destroy();
    $loggedOut = true;
} else {
    $loggedOut = false;
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Logged Out</title>

```

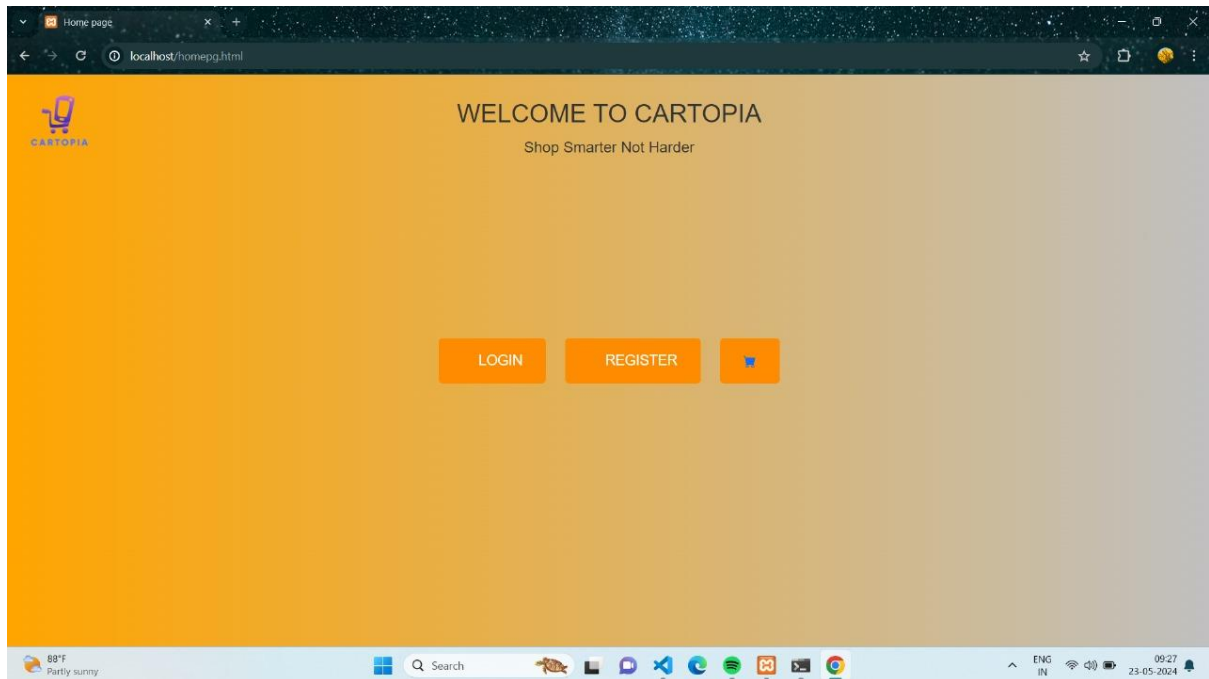
```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
<style>
  body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: linear-gradient(135deg, e30ee7, ffffff);
    margin: 0;
  }
  .message-box {
    text-align: center;
    padding: 20px;
    background-color: ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  }
</style>
</head>
<body>
  <div class="message-box">
    <?php if ($loggedOut): ?>
      <h1>LOGGED OUT SUCCESSFULLY</h1>
    <?php else: ?>
      <h1>NO ACTIVE SESSION</h1>
    <?php endif; ?>
    <a href="./homepg.html" class="btn btn-primary mt-3">Go to Home</a>
  </div>
</body>
</html>
```



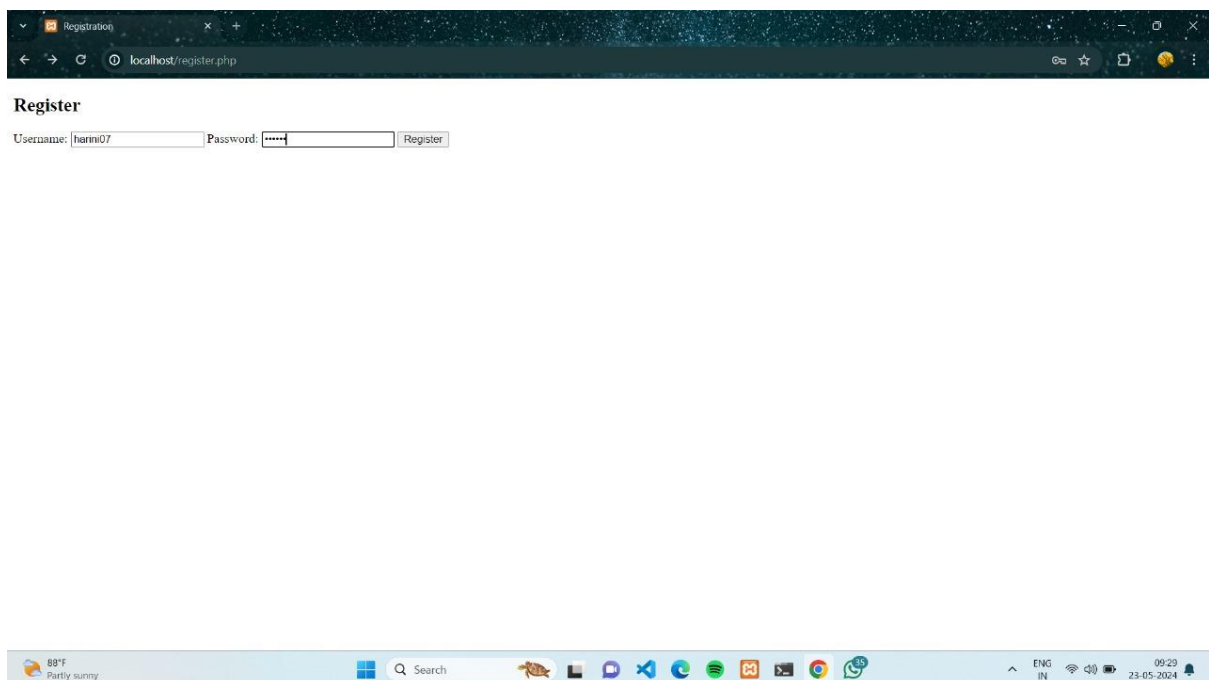
# CHAPTER 5 - RESULTS AND DISCUSSION:

## 5.1 OUTPUT

### HOME PAGE:

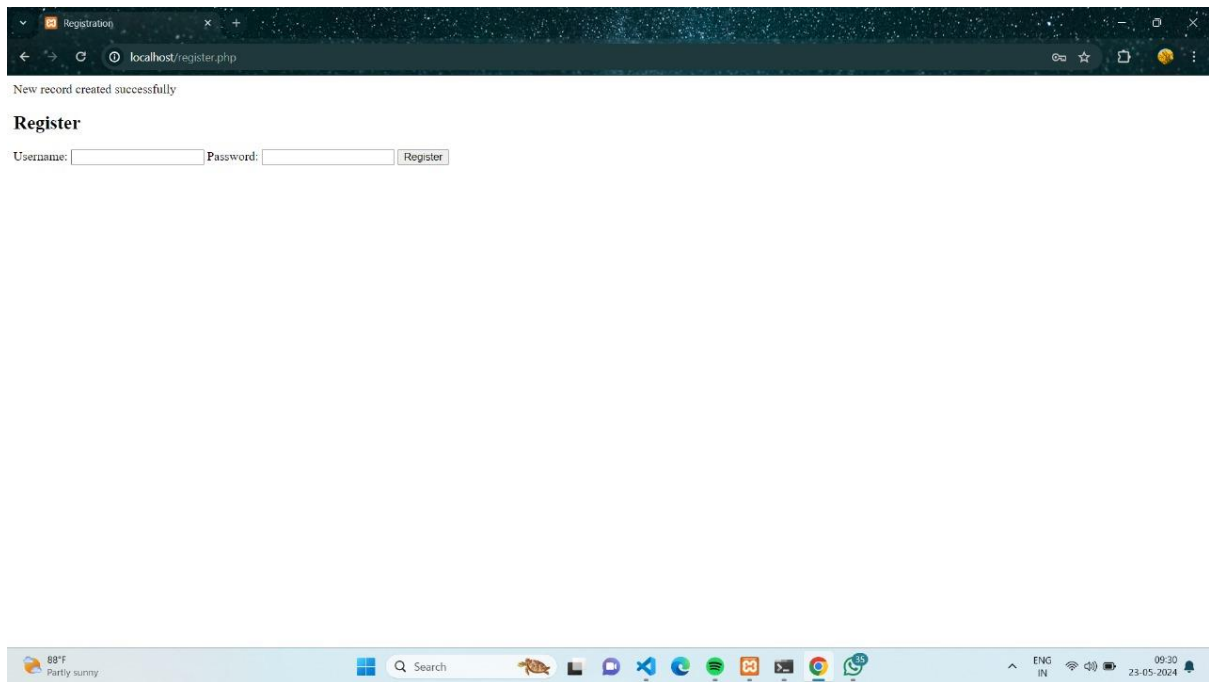


### REGISTER PAGE:

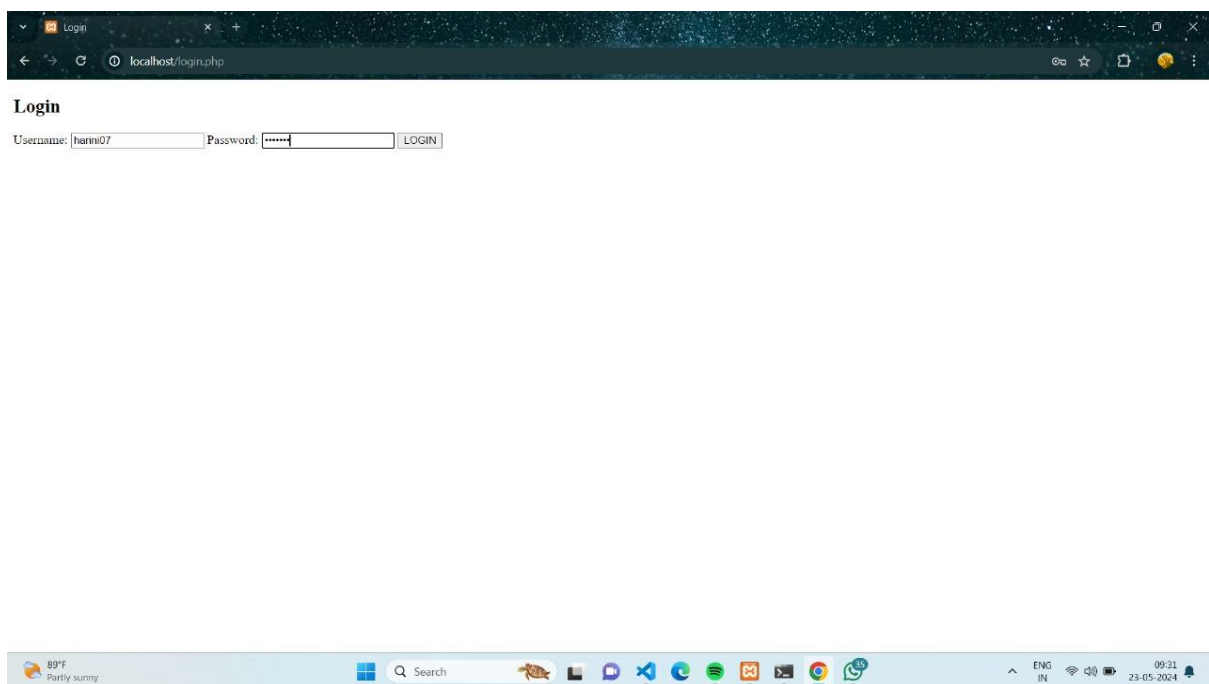




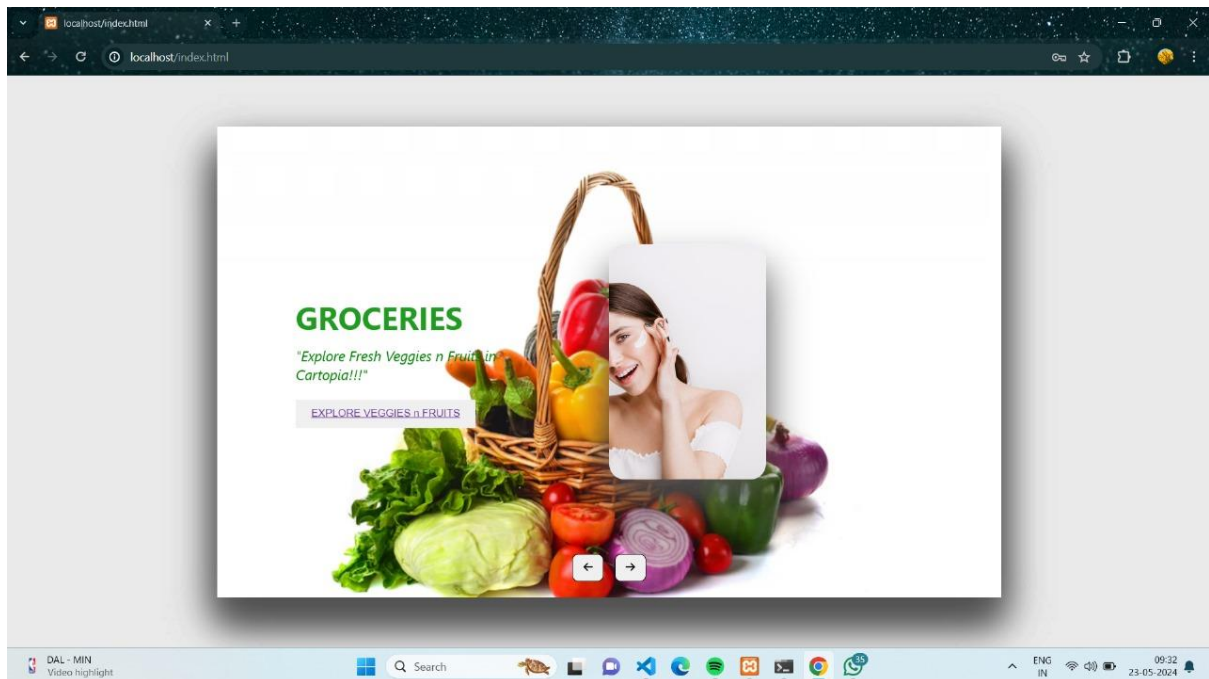
## AFTER REGISTRATION:



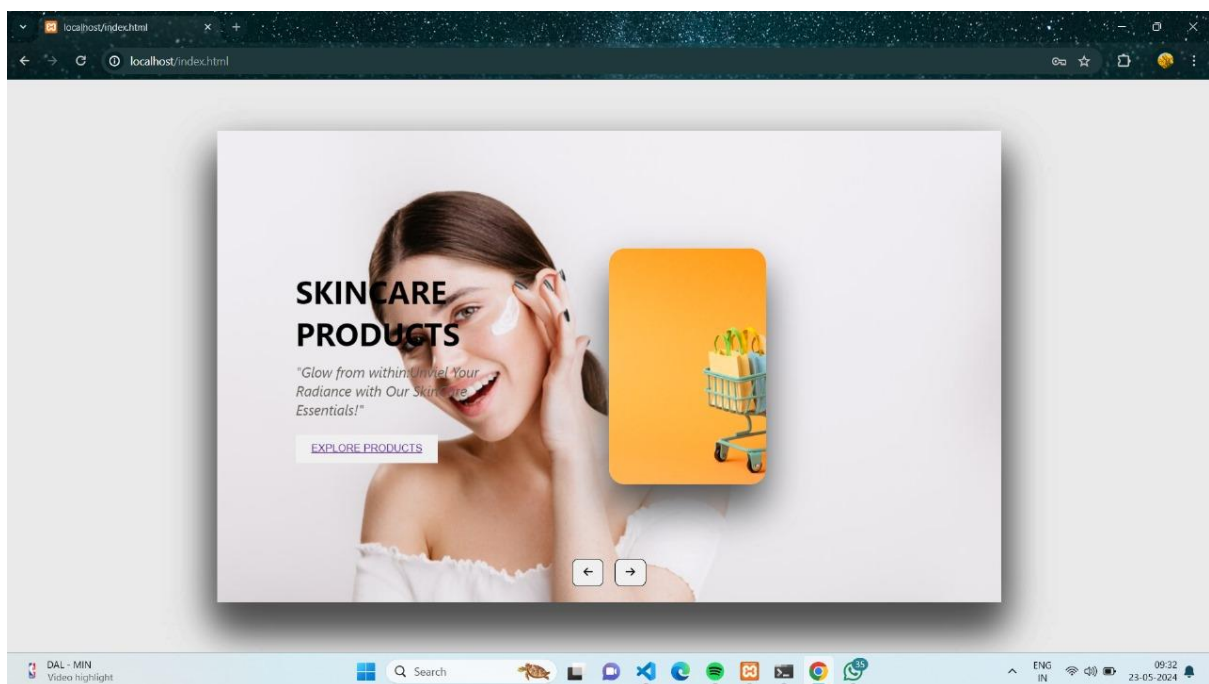
## LOGIN PAGE:



## GROCERIES PAGE:




## SKINCARE PAGE:



GROCERY PRODUCTS:

GROCERIES

localhost/groceries.php




CABBAGE Price: Rs.15/kg

1

+

-




BASMATI RICE Price: Rs.170

0

+

-



FORTUNE SUNFLOWER OIL Price: Rs.150

1

+

-

Cart

Report

Logout

89°F  
Partly sunny

Search

ENG  
IN


23-05-2024

09:33

SKINCARE PRODUCT:

SKINCARE

localhost/SC.php




SERUM Rs.300

1

+

-




SUNSCREEN Price: Rs.150

0

+

-



LOTION Price: Rs.200

1

+

-

Cart

Report

Logout

89°F  
Partly sunny

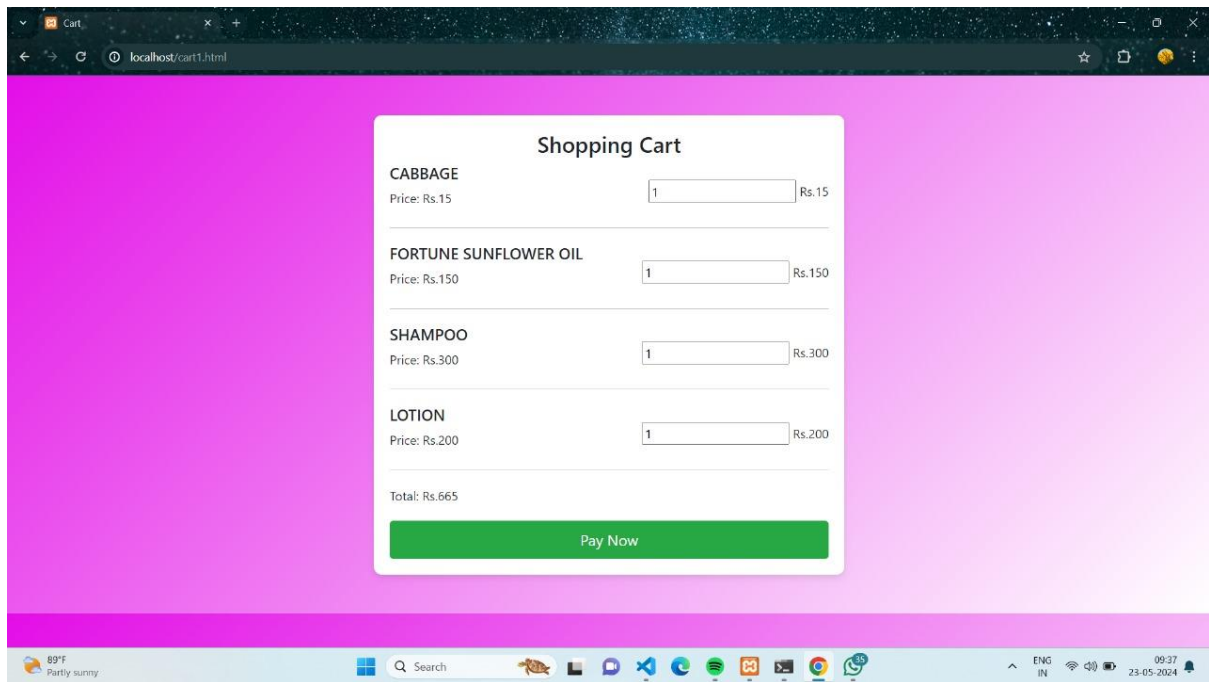
Search

ENG  
IN

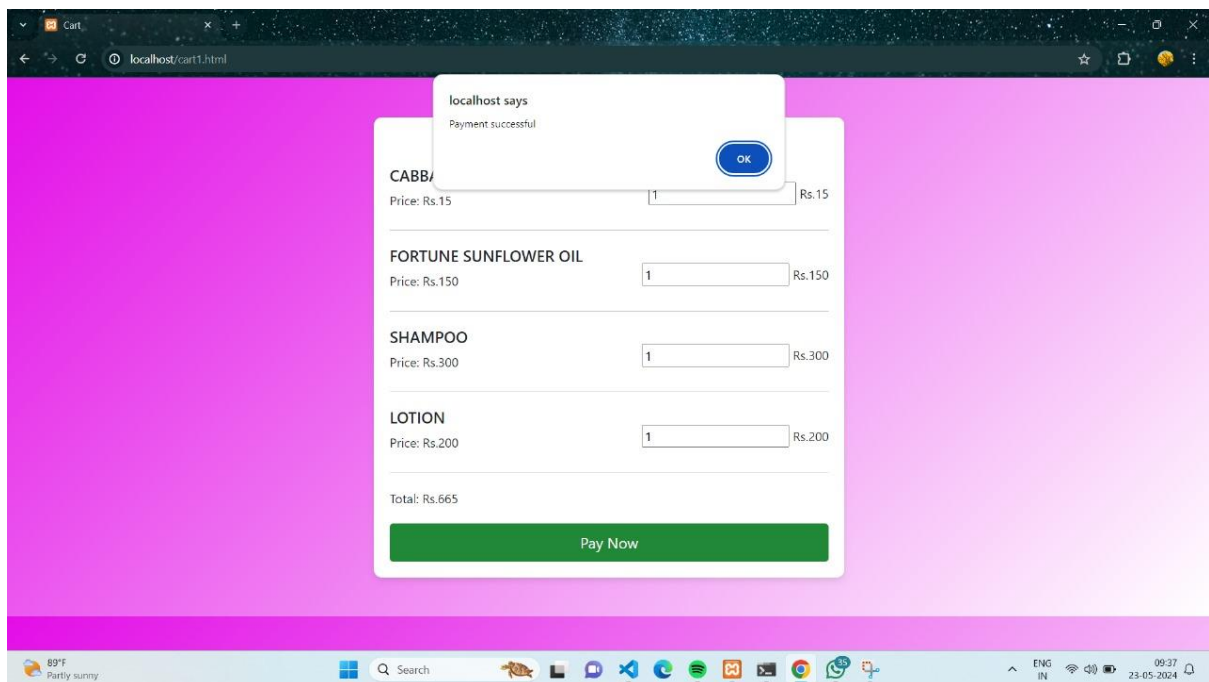
23-05-2024

09:35

## CART PAGE:



## PAYMENT PAGE:



ORDER PAGE:

localhost/save\_order.php

localhost/save\_order.php

Order saved successfully

REPORT PAGE:

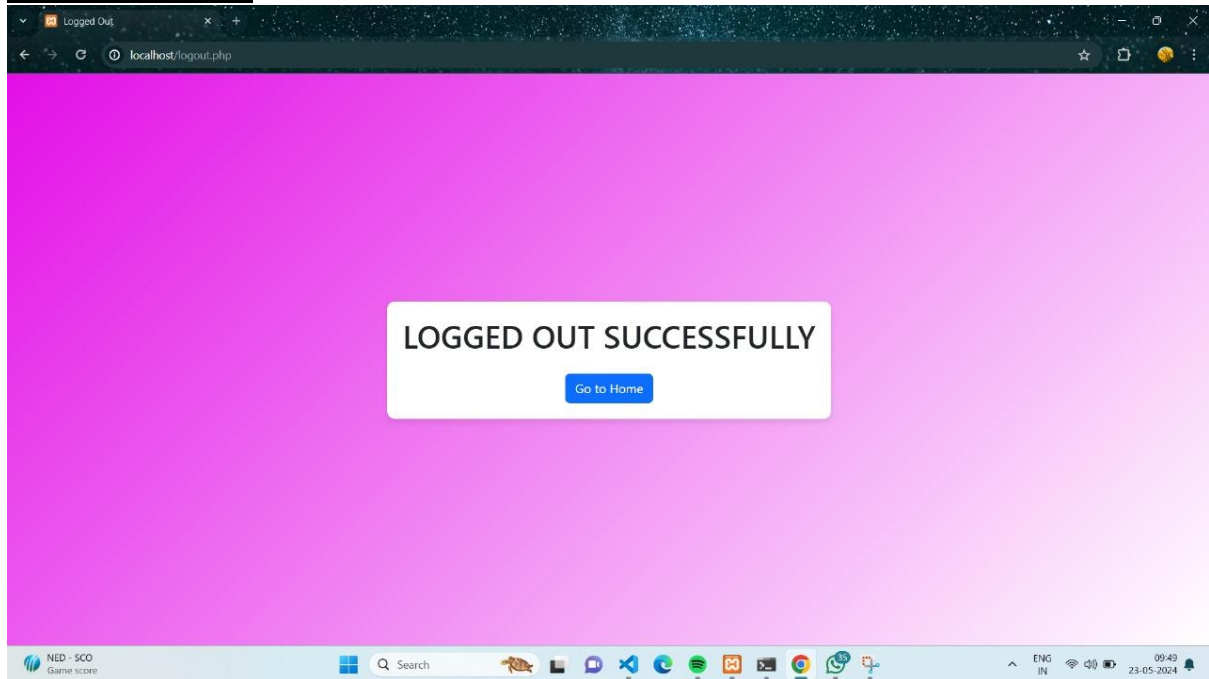
Weekly Report

localhost/report.php

Weekly Report

Username	Product Name	Quantity	Total Price	Order Date
annamalai	basmati	120	240.00	2024-05-22 21:55:20
annamalai	SHAMPOO	2	240.00	2024-05-22 22:05:22
annamalai	SHAMPOO	2	200.00	2024-05-22 23:01:51
annamalai	sunflower oil	1	100.00	2024-05-22 23:01:51
deepak05	sunflower oil	1	120.00	2024-05-21 12:42:26
deepak05	basmati	1	100.00	2024-05-21 12:42:26
deepak05	sunflower oil	2	240.00	2024-05-21 12:42:26
deepak05	basmati	2	400.00	2024-05-23 00:03:27
harini07	cabbage	1	15.00	2024-05-23 09:40:24
harini07	Fortune Sunflower Oil	1	150.00	2024-05-23 09:40:24
harini07	Shampoo	1	300.00	2024-05-23 09:40:24
harini07	Lotion	1	200.00	2024-05-23 09:40:24
harini07	cabbage	1	15.00	2024-05-23 09:40:24
harini07	Fortune Sunflower Oil	1	150.00	2024-05-23 09:40:24
harini07	Shampoo	1	300.00	2024-05-23 09:40:24
harini07	Lotion	1	200.00	2024-05-23 09:40:24

## **LOGOUT PAGE:**





## **5.2. TESTING**

### **6.1 Unit Testing**

Unit testing is a testing technique in which modules are tested individually. Small individual units of source code are tested to determine whether it is fit to use or not. Different modules are put to test while the modules are being developed.

### **6.2 Integration Testing**

Integration testing is the technique in which individual components or modules are grouped together and tested. It occurs after testing. The inputs for the integrated testing are the modules that have already been unit tested.

### **6.3 System Testing**

System testing is conducted on the entire system as a whole to check whether the system meets its requirements or not. software was installed on different systems and any errors or bugs that occurred were fixed.

### **6.4 Acceptance Testing**

User Acceptance is defined as a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment.

# Ch 6. CONCLUSION

## 7.1 Conclusion

In conclusion, Cartopia offers a seamless and convenient online shopping experience for users, providing a wide variety of products categorized neatly for easy browsing.

With features like user registration, login, and personalized account management, Cartopia ensures that users can access their preferences and past purchases effortlessly.

The platform allows users to explore groceries and skincare products, with detailed product descriptions and the ability to add items to their carts. The shopping cart functionality enables users to manage their selections, adjust quantities, and proceed to checkout securely with various payment options available.

After completing a purchase, users receive order confirmations and can track their order status. Additionally, Cartopia generates detailed order reports, including usernames, product details, quantities, prices, and transaction dates, providing users with insights into their purchase history.

Overall, Cartopia replicates the convenience of traditional brick-and-mortar stores while offering the benefits of online shopping, such as a broader product range and 24/7 accessibility. With its user-friendly interface and robust features, Cartopia aims to enhance the online shopping experience and meet the diverse needs of its customers.

## Ch 7. REFERENCES

### WEBSITES :

1. HTML - <https://www.w3schools.com/html/>
2. CSS - <https://www.w3schools.com/css/>
3. JAVASCRIPT - <https://www.w3schools.com/js/>
4. PHP - <https://www.php.net/manual/en/index.php>
5. MYSQL - <https://www.w3schools.com/MySQL/default.asp>
6. PHP YOUTUBE LINK -  
<https://www.youtube.com/watch?v=zZ6vybT1HQs>
7. MYSQL YOUTUBE LINK -  
<https://www.youtube.com/watch?reload=9&v=Hl4NZB1XR9c>
8. How to use PHP in HTML - <https://www.geeksforgeeks.org/how-to-use-php-in-html/>
9. XAMPP - <https://www.apachefriends.org/download.html>
10. phpMyAdmin -- <https://www.phpmyadmin.net/>

### BOOKS :

1. **MYSQL BOOK** - Efficient MySQL Performance: Best Practices and Techniques By Daniel Nichter
2. **PHP AND MYSQL** - Head First PHP & MySQL Paperback by [Lynn Beighley](#), [Michael Morrison](#)
3. Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites 3rd Edition by [Robin Nixon](#)