**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **CONTENT** | **PAGE NO** |
| 1 | INTRODUCTION  1.1Business Management System  1.2 Project Objective  1.3 Project Specification  1.4 Hardware specification  1.5 Software specification | 3  3  5  6  6 |
| 2 | ER DIAGRAM  2.1 Entities  2.2 Attributes  2.3 Relationship  2.4 Key Concepts  2.5 Clarity and Communication  2.6 Database Design  2.7 Normalization  2.8 ER DIAGRAM | 8  8  8  9  9  9  10  10 |
| 3 | SCHEMA DIAGRAM | 11 |
| 4 | DEVELOPMENT | 13 |
| 5 | APPENDIX  5.1 Source Code  5.1.1 HOME PAGE  5.1.2 ADDITEMS PAGE  5.1.3 BILLING PAGE  5.1.4 DASHBOARD PAGE  5.1.5 DB CONNECT  5.1.6 CUSTOMER  5.1.7 PRODUCT  5.1.8 PRODUCT CATEGORY  5.1.9 SHOPPING CART | 16  16  23  29  30  33  35  36  37  39 |
| 6 | SCREENSHOTS | 41 |
| 7  8  9 | CONCLUSION    FUTURE WORK  REFERENCES | 47  48  50 |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **Figure No** | **Figure Caption** | **Page Number** |
| 1 | HOME PAGE | 41 |
| 2 | LOGIN PAGE | 41 |
| 3 | DASHBOARD | 42 |
| 4 | PURCHASING PAGE | 43 |
| 5 | BILLING PAGE | 44 |
| 6 | ADDING ADDITIONAL ITEMS | 45 |
| 7 | ADDED ITEMS IN DATABASE | 46 |
| 8 | END PAGE | 46 |

**CHAPTER 1**

**INTRODUCTION**

**1.1** **Business Management System:**

A Business Management System serves as a pivotal tool for businesses seeking efficiency and accuracy in their financial operations. By automating the invoicing process, maintaining a centralized customer database, and offering robust payment tracking, the system not only streamlines administrative tasks but also significantly reduces the likelihood of errors in billing calculations. The integration with other business systems ensures data consistency and provides a holistic view of the organization's financial landscape, enabling better decision-making and improved customer relations.

The benefits of a Billing Management System extend beyond operational efficiency. Businesses gain access to insightful reporting and analytics, empowering them to make informed decisions based on real-time financial data. The system's ability to enhance accuracy, streamline processes, and provide valuable financial insights contributes to improved regulatory compliance, reduced costs associated with manual labor, and scalability, making it an indispensable asset for organizations of all sizes in managing their billing processes effectively.

**1.2 Project Objective:**

The objective of the Billing Management System is to create an efficient and user-friendly platform for managing billing processes within a business. This system aims to streamline the billing workflow, improve accuracy in financial transactions, and provide a comprehensive solution for tracking and managing customer invoices. The primary goals of the Billing Management System include:

**1. Automation of Billing Processes:**

**-** Automate the generation and distribution of invoices to customers.

- Implement an automated billing cycle to reduce manual intervention and errors.

**2.Accuracy and Precision:**

- Ensure accurate calculation of bills, taxes, and other financial transactions.

- Minimize billing discrepancies and enhance the precision of financial data.

**3.Customer Management:**

- Maintain a centralized database of customer information.

- Facilitate easy retrieval of customer details for billing and communication purposes.

**4. Reporting and Analytics:**

- Provide reporting tools for analyzing billing data, revenue trends, and outstanding payments.

- Enable data-driven decision-making through insightful analytics.

**5. Integration with Other Systems:**

- Integrate with existing business systems, such as inventory management or CRM, to ensure seamless data flow.

- Enhance overall business efficiency through system interoperability.

**6. User Accessibility and Experience:**

- Develop an intuitive user interface for easy navigation and use.

- Implement role-based access controls to secure sensitive billing information.

**1.3 Project Specification:**

The Billing Management System will have the following key features:

**1. Customer Management:**

- Add, edit, and delete customer information.

- Maintain a customer database with relevant details.

**2. Invoice Generation:**

- Create invoices automatically based on predefined billing cycles or manually as needed.

- Include itemized details, quantities, rates, and taxes in the invoices.

**3. Billing History:**

- Maintain a history of all generated invoices.

- Allow users to view, search, and filter billing records.

**4. Payment Tracking:**

- Record and track customer payments.

- Generate payment receipts and update payment status.

**5. Reporting and Analytics:**

- Provide standard reports on revenue, outstanding payments, and billing trends.

- Allow users to customize reports based on specific criteria.

**6. Security and Authentication:**

- Implement secure login mechanisms.

- Ensure data privacy and protection through proper access controls.

**1.4 Hardware Specification:**

The system is designed to be hardware-agnostic, enabling deployment on various configurations. The basic hardware specifications include:

**1. Server:**

- Sufficient processing power and memory to handle concurrent user requests.

- Adequate storage for storing customer data, invoices, and system logs.

**2. Database Server:**

- A robust database server to manage the customer database and billing records efficiently.

- Backup mechanisms to prevent data loss.

**3. Networking:**

- Stable network connectivity to ensure seamless communication between the server and client devices.

- Security measures to protect data during transmission.

**1.5 Software Specification:**

The Billing Management System is built on a technology stack that ensures reliability, scalability, and security. The software specifications include:

**1. Programming Language:**

- Server-side: PHP for backend logic.

- Client-side: HTML, CSS, JavaScript for the user interface.

**2. Database Management System:**

- MySQL or another relational database system for data storage.

**3. Web Framework:**

- Utilize a web framework (e.g., Laravel, CodeIgniter) to streamline development and enhance security.

**4. Frontend Libraries:**

- Use frontend libraries or frameworks (e.g., Bootstrap) for responsive and visually appealing user interfaces.

**5. Security Measures:**

- Implement secure coding practices to prevent common vulnerabilities.

- Use HTTPS for secure data transmission.

**6. Version Control:**

- Employ version control (e.g., Git) for collaborative development and code management.

**CHAPTER 2**

**E-R DIAGRAM:**

An Entity-Relationship Diagram (ERD) is a visual representation of the relationships among entities within a database. It is a powerful tool used by database designers and developers to model the structure of a database and define how different entities relate to each other. The ERD typically consists of entities, attributes, and relationships, providing a clear and concise overview of the database's logical structure.

**Components of an ERD:**

**2.1. Entities:**

- Entities are objects or concepts in the real world that are represented in the database. Each entity is depicted as a rectangle in the diagram, and examples could include "Customer," "Product," or "Order."

**2.2. Attributes:**

- Attributes are the properties or characteristics of entities. They are represented within ovals and describe the details of each entity. For instance, a "Customer" entity may have attributes such as "CustomerID," "Name," and "Email."

**2.3. Relationships:**

- Relationships illustrate how entities are connected or associated with each other. Lines connecting entities indicate the type and strength of the relationship. Common relationship types include "One-to-One," "One-to-Many," and "Many-to-Many."

**2.4 Key Concepts:**

**1. Primary Keys:**

- Each entity typically has a primary key, a unique identifier that distinguishes one record from another within the entity. Primary keys are crucial for establishing relationships between entities.

**2. Foreign Keys:**

- Foreign keys are attributes in one entity that refer to the primary key of another entity. They establish the connections between entities in a relational database.

**3. Cardinality:**

- Cardinality defines the numerical relationship between two entities in a relationship. It indicates how many instances of one entity are related to a single instance of another. Common cardinalities include "1" (one), "M" (many), or "0..1" (zero or one).

**Benefits of ERD:**

**2.5. Clarity and Communication:**

- ERDs provide a visual representation that is easily understandable by both technical and non-technical stakeholders. They serve as a communication tool between database designers, developers, and other project stakeholders.

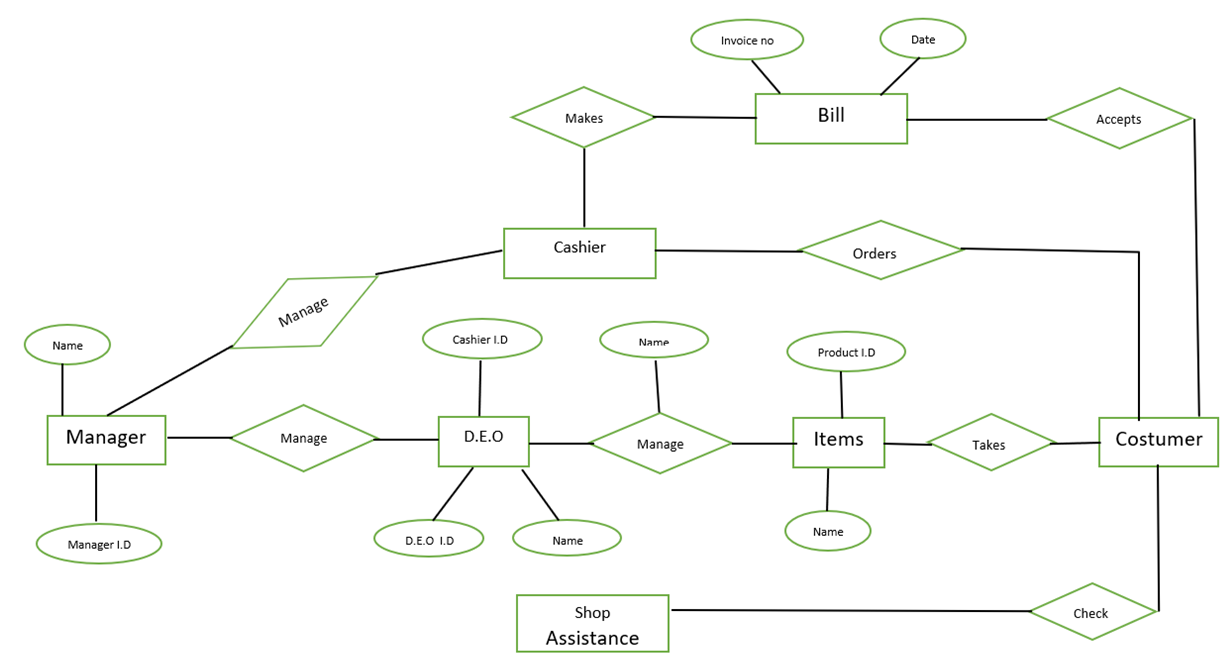
**2.6 Database Design:**

- ERDs guide the process of designing a database structure by mapping out entities, attributes, and relationships. They help ensure that the database model accurately represents the real-world scenario it aims to capture.

**2.7 Normalization:**

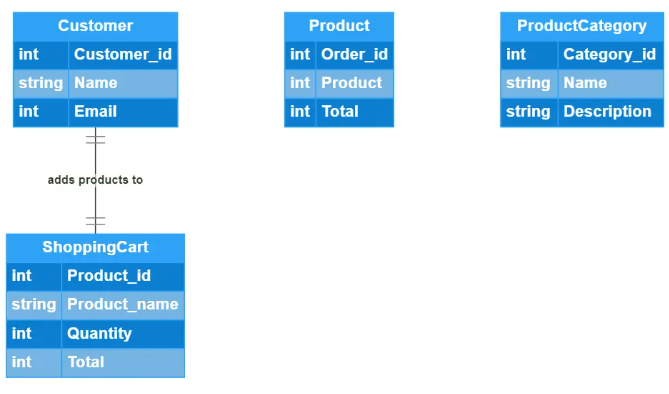
- ERDs aid in the normalization process by identifying relationships and dependencies between entities. Normalization helps eliminate data redundancy and ensures efficient data storage.

**2.8 ER DIAGRAM FOR HOSPITAL APPOINTMENT SYSTEM**



**CHAPTER 3**

**SCHEMA DIAGRAM:**

****

**Entity: Customer**

- Attributes:

- `customerID` (customer ID)

- `cemail` (customer email)

- `cpassword` (customer password)

**Entity: Product**

- Attributes:

- `productID` (product ID)

- `productName` (product name)

- `price` (product price)

- `description` (product description)

- `categoryID` (category ID)

**Entity: Product Category**

- Attributes:

- `categoryID` (category ID)

- `categoryName` (category name)

**Entity: Shopping Cart**

- Attributes:

- `cartID` (cart ID)

- `customerID` (customer ID)

- `creationDate` (date the cart was created)

- `totalPrice` (total price of items in the cart)

**CHAPTER 4**

**DEVELOPMENT**

**1. Home Page (home.html):**

- The home page serves as the entry point to the Business Management System.

- It includes a simple header with a navigation link to the "ulp.php" page.

- The main content provides a welcoming message and instructions to use the navigation menu.

**2. Add Items Page (additems.php):**

- This page allows users to add items to the system.

- The PHP code handles form submissions by retrieving input values, creating an item object, and storing it in a cookie named 'items'.

- The form uses a combination of HTML and PHP to gather information from the user.

- JavaScript is used to handle the form submission asynchronously, preventing the default form behavior and storing the item in the browser's localStorage.

**3. Billing Page (billing.php):**

- The billing page processes form submissions to insert billing information into a database.

- The PHP code checks if the form is submitted, retrieves form data, and attempts to insert the data into a 'billing' table in the database. However, there's a syntax error in the SQL query ('insert into' should be 'INSERT INTO').

- The connection to the database ('$conn') is assumed but not explicitly shown.

**4. Dashboard Page (dash.php):**

- This page displays a dashboard with a list of newly added items.

- The PHP code retrieves stored items from the 'items' cookie and generates a list in HTML.

- The dashboard is styled using simple CSS for a clean presentation**.**

**5. Error Page (index.html):**

- The error page is represented by an HTML document claiming a virus has been found.

- This appears to be a placeholder and is unrelated to the functionality of the Business Management System.

**1. Frontend Development:**

- Improve the user interface by enhancing the styling and layout of the pages.

- Consider adding consistent styling across pages for a cohesive look.

- Enhance user feedback and validation messages for a better user experience.

**2. Backend Development:**

- Address the syntax error in the billing page's SQL query.

- Establish a clear database connection and ensure proper error handling.

- Consider incorporating server-side validation and sanitation for user inputs.

**3. Security:**

- Implement security measures to prevent SQL injection and other vulnerabilities.

- Ensure sensitive information, such as database credentials, is handled securely.

**4. Testing:**

- Conduct thorough testing of the system, including form submissions and database interactions.

- Address any potential security vulnerabilities identified during testing.

**5. Documentation:**

- Provide clear documentation for developers, including instructions for setting up the database and understanding the code structure.

- Consider commenting the code to explain key functionalities.

**CHAPTER 5**

**APPENDIX**

**5.1 SOURCE CODE**

**5.1.1 Home Page:**

<!DOCTYPE html>

<html>

<head>

<title>Business Management System Home</title>

<link rel="stylesheet" type="text/css" href="css/style.css">

</head>

<body bgcolor="yellow">

<header>

<h1>Business Management System</h1>

<nav>

<ul>

<li><a href="ulp.php">Home</a></li>

</ul>

</nav>

</header>

<main>

<h2>Welcome to our Business Management System</h2>

<p>This is the home page of our system.</p>

<p>Use the navigation menu to access other pages.</p>

</main>

</body>

</html> additems page:

<?php

// Check if form is submitted

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

// Retrieve input values

$itemName = $\_POST['itemName'];

$itemPrice = $\_POST['itemPrice'];

$itemDescription = $\_POST['itemStock'];

// Create an object to store item details

$newItem = [

'name' => $itemName,

'price' => $itemPrice,

'description' => $itemDescription

];

// Check if localStorage already has items

$items = [];

if (isset($\_COOKIE['items'])) {

$items = json\_decode($\_COOKIE['items'], true);

}

// Add the new item to the array

$items[] = $newItem;

// Store the updated items array in a cookie

setcookie('items', json\_encode($items), time() + (86400 \* 30), '/'); // Cookie expires in 30 days

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Shop</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f1f1f1;

margin: 0;

padding: 20px;

}

h1 {

color: #333;

text-align: center;

}

form {

max-width: 400px;

margin: 0 auto;

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

label {

display: block;

margin-bottom: 8px;

color: #333;

}

input[type="text"],

input[type="number"],

textarea {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

input[type="submit"] {

background-color: #4caf50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 4px;

cursor: pointer;

margin-top: 10px;

}

input[type="submit"]:hover {

background-color: #45a049;

}

/\* Optional: Add some colorful elements \*/

.colorful-container {

background-color: yellow;

padding: 10px;

border-radius: 8px;

margin-top: 20px;

text-align: center;

}

.colorful-text {

color: green;

font-weight: bold;

}

/\* Additional colorful styling for submit button \*/

input[type="submit"].colorful-button {

background-color: pink;

}

input[type="submit"].colorful-button:hover {

background-color: lightpink;

}

</style>

</head>

<body>

<h1>Add Item</h1>

<form id="addItemForm" action="shop.php" method="post">

<label for="itemName">Item Name:</label>

<input type="text" id="itemName" name="itemName" required>

<br>

<label for="itemPrice">Item Price:</label>

<input type="number" id="itemPrice" name="itemPrice" required>

<br>

<label for="itemDescription">Item Stock:</label>

<textarea id="itemDescription" name="itemStock" required></textarea>

<br>

<input type="submit" value="Add Item" class="colorful-button">

</form>

<div class="colorful-container">

<p class="colorful-text">ITEMS VIEW</p>

</div>

<script>

// Event listener for form submission

document.getElementById('addItemForm').addEventListener('submit', function(event) {

event.preventDefault(); // Prevent form submission

// Retrieve input values

const itemName = document.getElementById('itemName').value;

const itemPrice = document.getElementById('itemPrice').value;

const itemDescription = document.getElementById('itemDescription').value;

// Create an object to store item details

const newItem = {

name: itemName,

price: itemPrice,

description: itemDescription

};

// Check if localStorage already has items

let items = [];

if (localStorage.getItem('items')) {

items = JSON.parse(localStorage.getItem('items'));

}

// Add the new item to the array

items.push(newItem);

// Store the updated items array in localStorage

localStorage.setItem('items', JSON.stringify(items));

});

</script>

</body>

</html>

**Description**:

**Home Page (home.html):**

The home page serves as the starting point for the Business Management System. It includes a simple header with the system name, a navigation link to the "ulp.php" page, and a main section with a welcome message. Users are instructed to use the navigation menu to access other pages.

**5.1.2 Additems Page:**

<?php

// Check if form is submitted

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

// Retrieve input values

$itemName = $\_POST['itemName'];

$itemPrice = $\_POST['itemPrice'];

$itemDescription = $\_POST['itemStock'];

// Create an object to store item details

$newItem = [

'name' => $itemName,

'price' => $itemPrice,

'description' => $itemDescription

];

// Check if localStorage already has items

$items = [];

if (isset($\_COOKIE['items'])) {

$items = json\_decode($\_COOKIE['items'], true);

}

// Add the new item to the array

$items[] = $newItem;

// Store the updated items array in a cookie

setcookie('items', json\_encode($items), time() + (86400 \* 30), '/'); // Cookie expires in 30 days

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Shop</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f1f1f1;

margin: 0;

padding: 20px;

}

h1 {

color: #333;

text-align: center;

}

form {

max-width: 400px;

margin: 0 auto;

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

label {

display: block;

margin-bottom: 8px;

color: #333;

}

input[type="text"],

input[type="number"],

textarea {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

input[type="submit"] {

background-color: #4caf50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 4px;

cursor: pointer;

margin-top: 10px;

}

input[type="submit"]:hover {

background-color: #45a049;

}

/\* Optional: Add some colorful elements \*/

.colorful-container {

background-color: yellow;

padding: 10px;

border-radius: 8px;

margin-top: 20px;

text-align: center;

}

.colorful-text {

color: green;

font-weight: bold;

}

/\* Additional colorful styling for submit button \*/

input[type="submit"].colorful-button {

background-color: pink;

}

input[type="submit"].colorful-button:hover {

background-color: lightpink;

}

</style>

</head>

<body>

<h1>Add Item</h1>

<form id="addItemForm" action="shop.php" method="post">

<label for="itemName">Item Name:</label>

<input type="text" id="itemName" name="itemName" required>

<br>

<label for="itemPrice">Item Price:</label>

<input type="number" id="itemPrice" name="itemPrice" required>

<br>

<label for="itemDescription">Item Stock:</label>

<textarea id="itemDescription" name="itemStock" required></textarea>

<br>

<input type="submit" value="Add Item" class="colorful-button">

</form>

<div class="colorful-container">

<p class="colorful-text">ITEMS VIEW</p>

</div>

<script>

// Event listener for form submission

document.getElementById('addItemForm').addEventListener('submit', function(event) {

event.preventDefault(); // Prevent form submission

// Retrieve input values

const itemName = document.getElementById('itemName').value;

const itemPrice = document.getElementById('itemPrice').value;

const itemDescription = document.getElementById('itemDescription').value;

// Create an object to store item details

const newItem = {

name: itemName,

price: itemPrice,

description: itemDescription

};

// Check if localStorage already has items

let items = [];

if (localStorage.getItem('items')) {

items = JSON.parse(localStorage.getItem('items'));

}

// Add the new item to the array

items.push(newItem);

// Store the updated items array in localStorage

localStorage.setItem('items', JSON.stringify(items));

});

</script>

</body>

</html>

**Description:**

**Add Items Page (additems.php):**

This page allows users to add items to the business management system. Upon form submission, the PHP script retrieves input values for the item name, price, and description. It then creates an object representing the item and stores it in a cookie named 'items'. The form is styled with CSS, and additional JavaScript is used to prevent the default form submission and store the item details in the browser's localStorage.

**5.1.3 Billing Page:**

<?php

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

// Get the form data

$name = $\_POST['Name'];

$address = $\_POST['Address'];

$email = $\_POST['Email'];

$number = $\_POST['Phone'];

$tot=$\_POST['Totalamount'];

$conn=mysqli\_connect('localhost','root','','project');

$qu=insert into billing values($name,$address,$email,$number,$tot);

$a=mysqli\_query($conn,$qu);

}

?>

**Description:**

**Billing Page (billing.php):**

The billing page processes form submissions to insert billing information into a database. It retrieves form data such as name, address, email, phone number, and total amount. However, there is a syntax error in the SQL query (should be 'INSERT INTO' instead of 'insert into'). The connection to the database is assumed but not explicitly shown.

**5.1.4** **Dashboard Page:**

<?php

// Retrieve stored items from localStorage

$items = json\_decode($\_COOKIE['items'], true);

?>

<!DOCTYPE html>

<html>

<head>

<title>Dashboard</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

padding: 20px;

}

h1 {

color: #333;

}

h2 {

color: #666;

}

#itemList {

list-style-type: none;

padding: 0;

}

li {

background-color: #fff;

margin-bottom: 10px;

padding: 10px;

border-radius: 5px;

}

</style>

</head>

<body>

<h1>Dashboard</h1>

<h2>Newly Added Items:</h2>

<ul id="itemList">

<?php

// Check if there are items to display

if ($items && count($items) > 0) {

// Iterate through each item and create list elements

foreach ($items as $item) {

$name = $item['name'];

$price = $item['price'];

$description = $item['description'];

echo '<li>Name: ' . $name . ', Price: ' . $price . ', Description: ' . $description . '</li>';

}

}

?>

</ul>

</body>

</html>

**Dashboard Page:**

<html><head><meta content="text/html; charset=UTF-8" http-equiv="content-type"><title>Ramco Institute of Technology Mail - Download error</title></head><body>Virus found</body></html>:write a description for the above code.

**Description:**

**Dashboard Page (dash.php):**

The dashboard page retrieves stored items from the 'items' cookie and displays them in a list format. The HTML structure is simple, with a heading, a list element, and styling to create a visually appealing dashboard. The PHP code decodes the 'items' cookie and generates list elements for each item, showing the name, price, and description.

**5.1.5 DB CONNECT**

<?php

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

// Get the form data

$name = $\_POST['Name'];

$address = $\_POST['Address'];

$email = $\_POST['Email'];

$number = $\_POST['Phone'];

$tot=$\_POST['Totalamount'];

$conn=mysqli\_connect('localhost','root','','project');

$qu=insert into billing values($name,$address,$email,$number,$tot);

$a=mysqli\_query($conn,$qu);

}

?>

**Description:**

**1. Form Submission Check:**

- `if ($\_SERVER['REQUEST\_METHOD'] === 'POST'):` checks if the form is submitted using the POST method.

**2. Form Data Retrieval:**

- `$name = $\_POST['Name'];`: Retrieves the value of 'Name' from the submitted form.

- `$address = $\_POST['Address'];`: Retrieves the value of 'Address' from the submitted form.

- `$email = $\_POST['Email'];`: Retrieves the value of 'Email' from the submitted form.

- `$number = $\_POST['Phone'];`: Retrieves the value of 'Phone' from the submitted form.

- `$tot=$\_POST['Totalamount'];`: Retrieves the value of 'Totalamount' from the submitted form.

**3. Database Connection:**

- `$conn=mysqli\_connect('localhost','root','','project');`: Establishes a connection to the MySQL database located at 'localhost' with the username 'root' and an empty password. The specific database used is 'project'.

**4. SQL Query Preparation:**

- `$qu=insert into billing values($name,$address,$email,$number,$tot);`: Attempts to prepare an SQL query to insert values into the 'billing' table. However, there's a syntax error here. It should be corrected to `INSERT INTO` like: `$qu = "INSERT INTO billing VALUES ('$name', '$address', '$email', '$number', '$tot')";`.

**5. Query Execution:**

- `$a=mysqli\_query($conn,$qu);`: Executes the SQL query using the established database connection. However, due to the syntax error mentioned above, this line may result in an error.

**5.1.6 Customer:**

**Description:**

**1. Entity Type:**

- The "Customer" entity is a fundamental component of the online platform's database.

**2. Unique Identifier:**

- Each customer is assigned a unique identification through the "customerID."

**3. Authentication Credentials:**

- Customers utilize an "cemail" (customer email) and "cpassword" (customer password) for secure and personalized authentication.

**4. Secure Authentication:**

- The use of email and password ensures a secure login process, safeguarding customer accounts.

**5. Individual Representation:**

- This entity represents individual users who engage with the online platform.

**6. User Account Management:**

- The system leverages the "Customer" entity to effectively manage user accounts.

**7. Personalization**:

- The entity enables a personalized experience for each customer based on their unique identification.

**8. Customer Interaction:**

- Customers interact with the platform through their respective "customerID."

**9. Central Role**:

- The "Customer" entity plays a central role in user-centric operations within the system.

**10. Database Foundation**:

- This entity serves as a foundational element within the database, contributing to the overall structure and functionality of the online platform.

**5.1.7 Product:**

**Description:**

**1. Entity Type:**

- The "Product" entity is a key component of the system's database, representing items available for purchase.

**2. Unique Identifier:**

- Each product is uniquely identified by a "productID," facilitating precise differentiation among items.

**3. Descriptive Attributes:**

- The entity includes descriptive attributes such as "productName," "price," and "description" to comprehensively define each product's characteristics.

**4. Characteristic Definition**:

- The "productName" attribute specifies the name of the product, while "price" indicates the cost associated with the item.

**5. Detailed Description:**

- The "description" attribute provides a detailed overview or information about the product.

**6. Category Linkage:**

- The "categoryID" attribute establishes a linkage between the product and a specific category, contributing to effective catalog organization.

**7. Category Organization:**

- This linkage aids in systematically organizing products within the catalog based on their assigned categories.

**8. Catalog Management:**

- The "Product" entity plays a crucial role in the management and organization of the overall product catalog.

**9. Inventory Differentiation:**

- The "productID" ensures unique identification, allowing for precise differentiation and tracking of individual products in the inventory.

**10. Database Integration:**

- As a core element of the database, the "Product" entity forms the foundation for product-related operations and interactions within the system.

**5.1.8 Product Category:**

**Description:**

**1. Entity Type:**

- The "Product Category" entity is a crucial component designed for categorizing products within the system.

**2. Unique Identifier:**

- Each category is uniquely identified by a "categoryID," ensuring a distinct label for organizational purposes.

**3. Descriptive Labeling:**

- The "categoryName" attribute provides a descriptive label for each product category, offering clarity and understanding.

**4. Categorization Purpose:**

- The primary function of this entity is to categorize products systematically based on their shared characteristics or features.

**5. Enhanced Organization**:

- By utilizing product categories, the system enhances the overall organization of the product catalog, making it more user-friendly.

**6. Browsing Experience**:

- Customers benefit from an improved browsing experience as they can navigate and explore products more efficiently within well-defined categories.

**7. Systematic Catalog Management:**

- The entity contributes to the systematic management of the product catalog, streamlining operations related to product presentation and accessibility.

**8. Logical Grouping:**

- Products with similar attributes or purposes are logically grouped together under specific categories, aiding in logical organization.

**9. Facilitates Search and Discovery:**

- The categorization facilitates a more straightforward search and discovery process for customers, allowing them to find products of interest with greater ease.

**10. Foundation for Navigation:**

- As an integral part of the system, the "Product Category" entity forms the foundation for navigation and exploration, influencing the structure and presentation of products within the platform.

**5.1.9 Shopping Cart:**

**Description:**

**1. Entity Type:**

- The "Shopping Cart" entity is designed to manage the temporary storage of products chosen by customers during their shopping sessions.

**2. Unique Identifier:**

- Each shopping cart is uniquely identified by a "cartID," ensuring a distinct record for each customer's cart.

**3. Customer Association:**

- The "customerID" attribute associates each shopping cart with a specific customer, enabling personalized cart management.

**4. Timestamp Capture:**

- The "creationDate" attribute captures the timestamp when a customer initiates their shopping cart, providing a record of the cart's creation time.

**5. Total Price Calculation:**

- The "totalPrice" attribute aggregates the costs of all items added to the cart, offering a real-time calculation of the total cost for the customer.

**6. Checkout Facilitation:**

- This entity plays a pivotal role in facilitating the checkout process, allowing customers to review and manage their selected items before completing a purchase.

**7. Temporary Storage:**

- The shopping cart serves as a temporary storage space for selected products, enabling customers to explore the platform and add items at their convenience.

**8. Convenient Review:**

- Customers can conveniently review the contents of their shopping cart, including product details and accumulated costs, providing transparency in the purchase decision-making process.

**9. User-Friendly Experience:**

- The entity contributes to a user-friendly experience by offering a centralized and organized space for customers to manage their potential purchases.

**10. Enhanced Customer Engagement**:

- By providing a mechanism for users to collect and review items before finalizing a purchase, the shopping cart enhances customer engagement and satisfaction, contributing to a positive overall shopping experience.

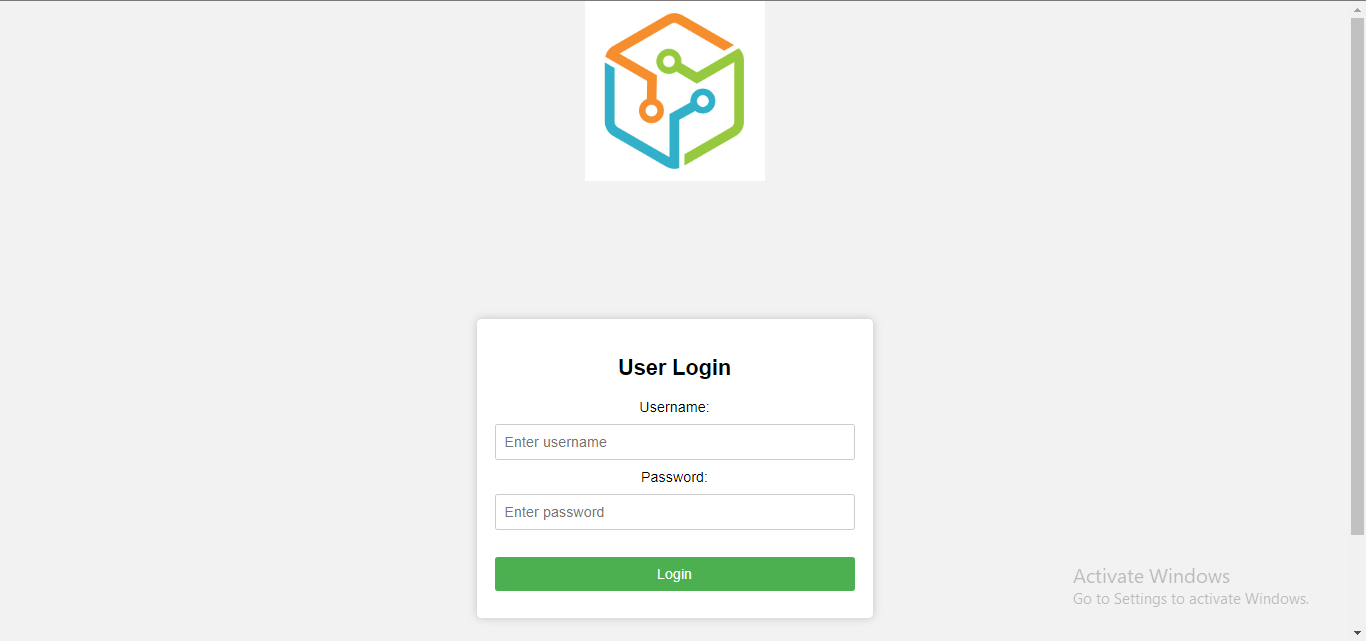
**CHAPTER 6**

**SCREENSHOT**

**FIGURE-1 (Home Page)**

****

**FIGURE-2 (Login Page)**



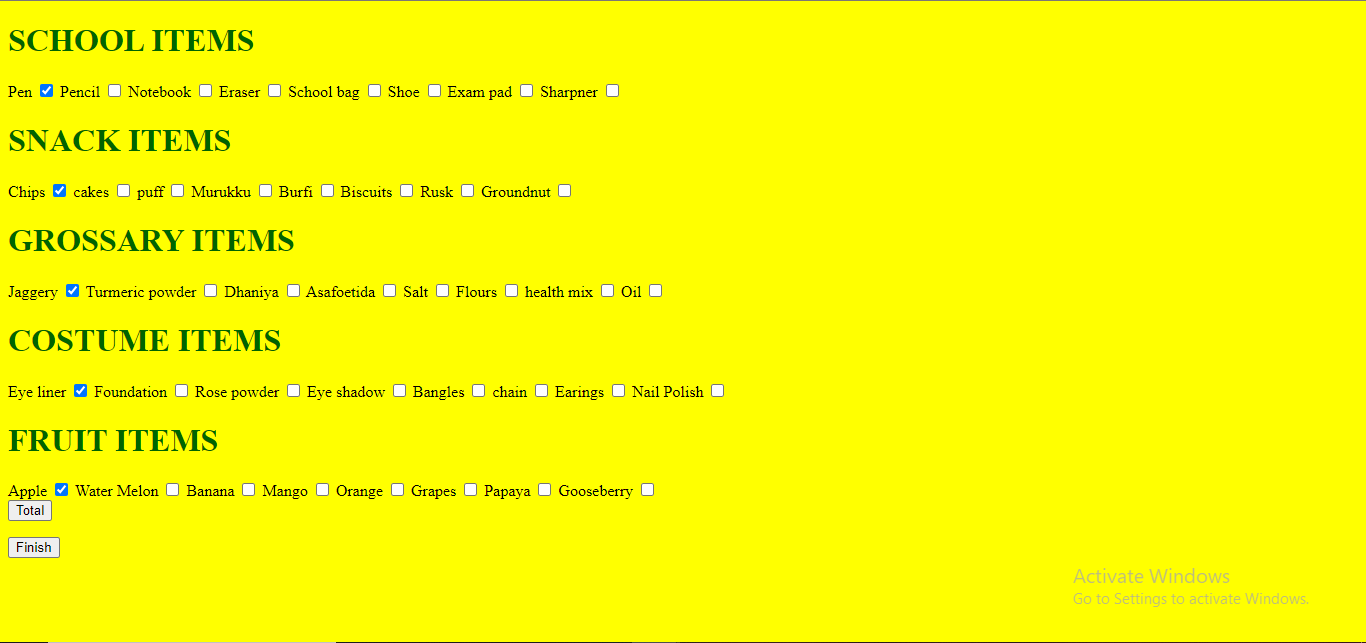
**FIGURE-3 (Dash Board)**

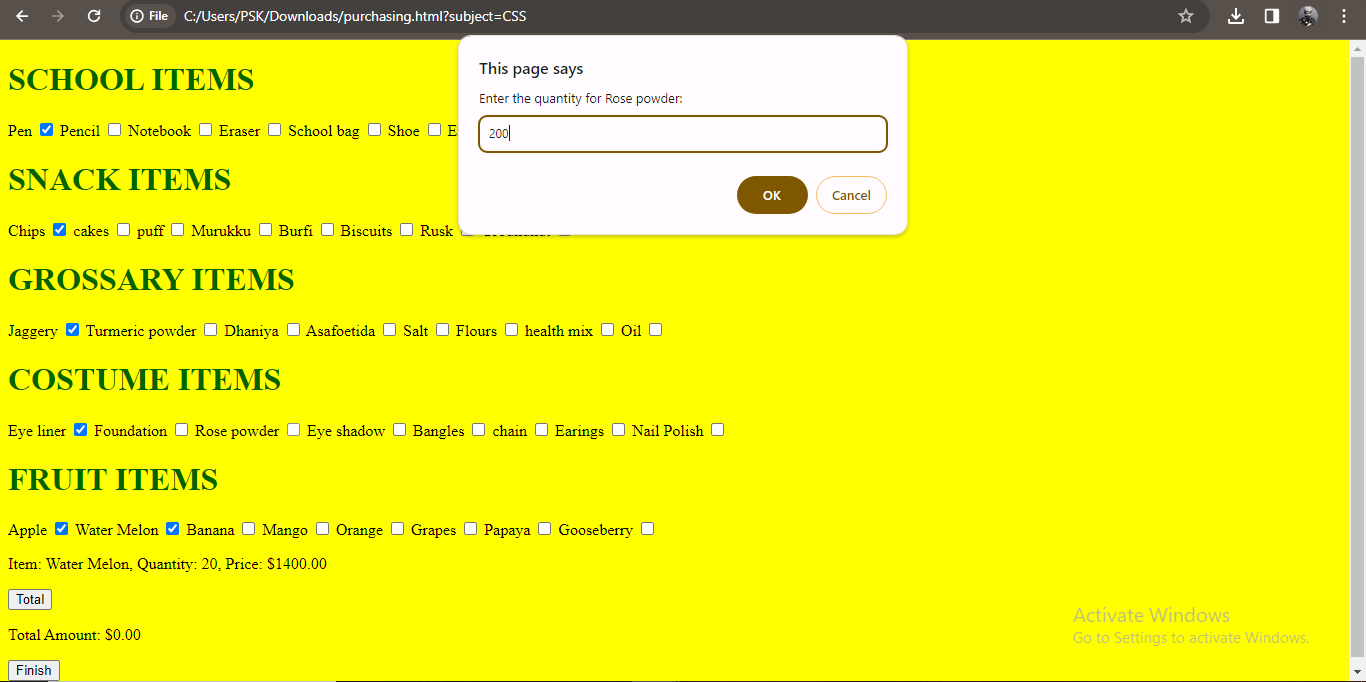






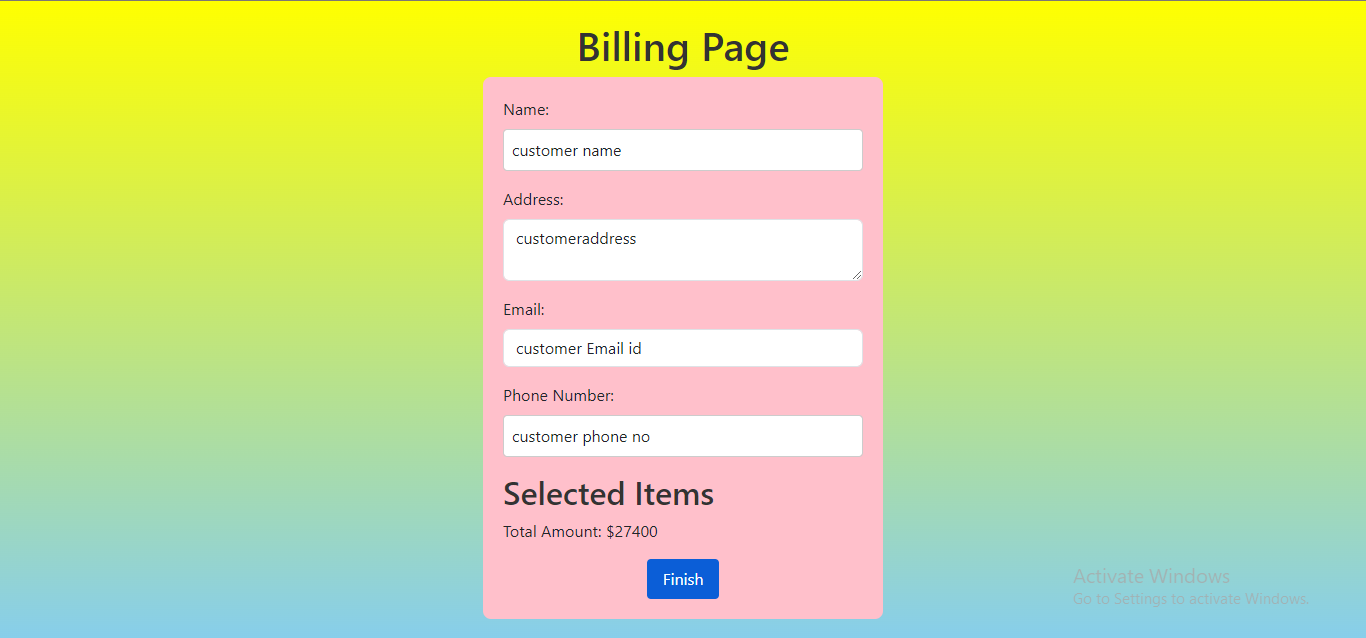
**FIGURE-4 (Purchasing Page)**



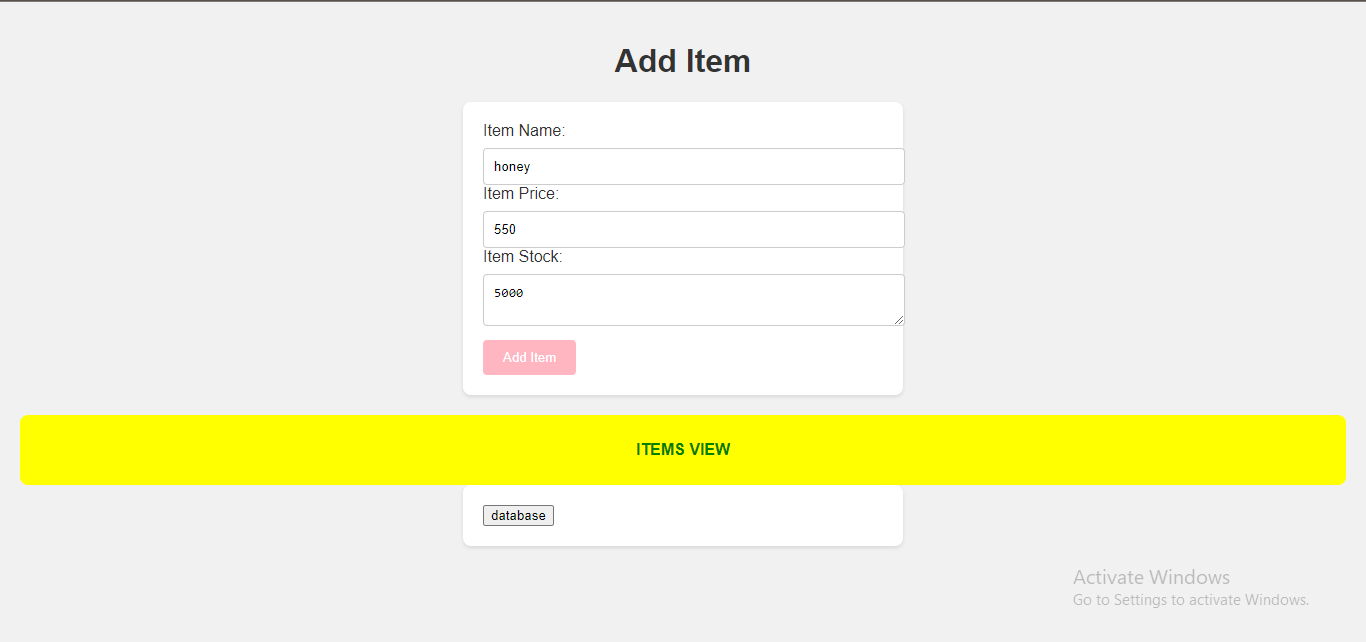


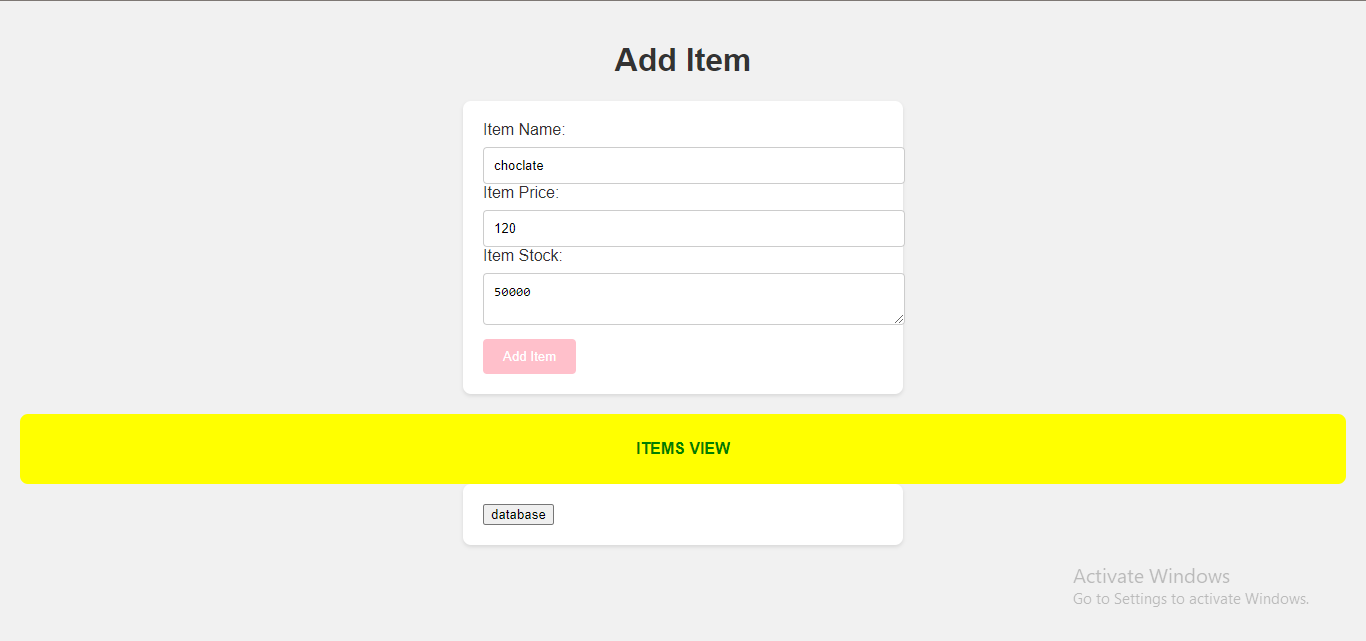


**FIGURE-5 (Billing Page)**

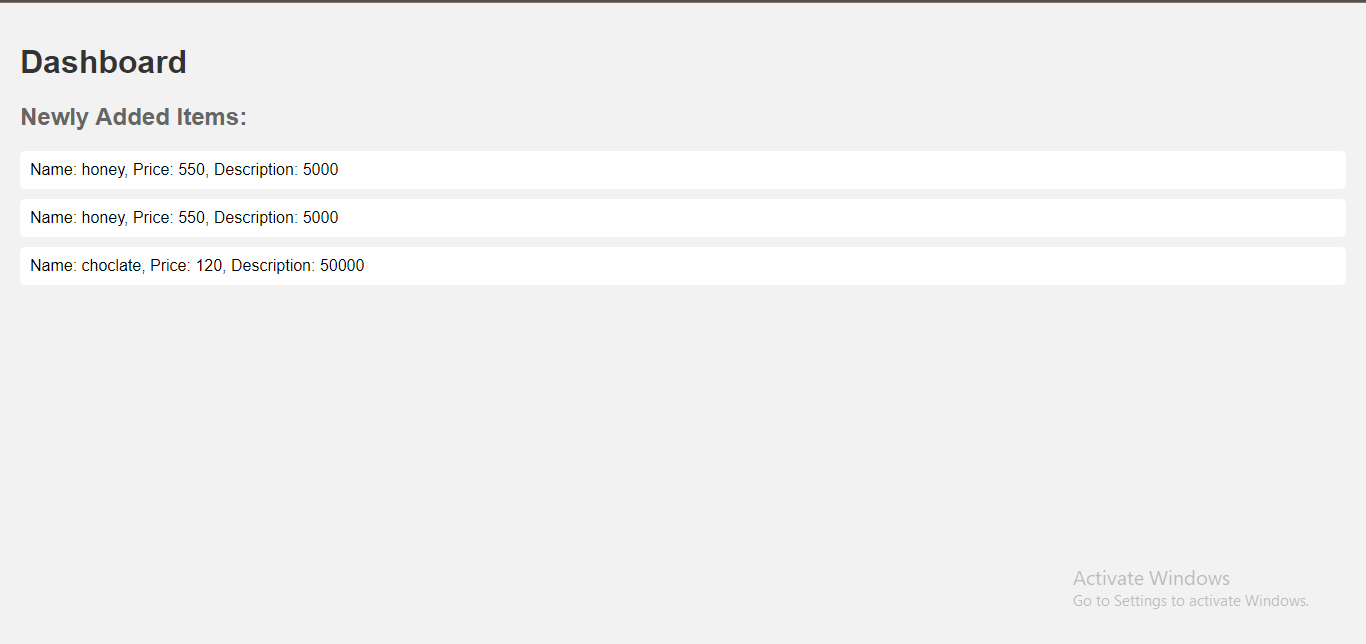


**FIGURE-6 (Adding Additional Items)**

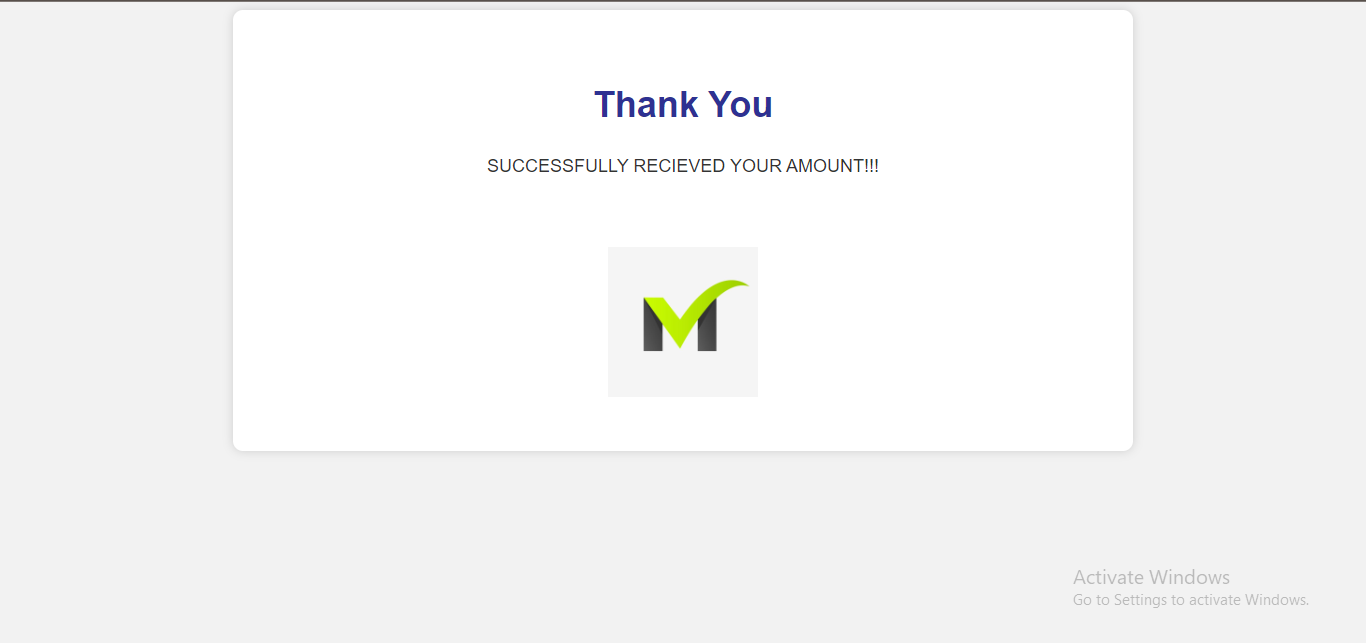




**FIGURE-7 (Added items in Database)**



**FIGURE-8** **(End PAGE)**

****

**CHAPTER 7**

**CONCLUSION:**

* In conclusion, the development of the Business Management System encompasses various key components that contribute to its functionality and user experience. The Home Page serves as the entry point, welcoming users and providing essential navigation. The Add Items page enables users to input and store new items into the system, utilizing both client-side (JavaScript) and server-side (PHP) technologies for seamless integration. The Billing Page captures customer information during the checkout process, ensuring accurate records of transactions.
* The Dashboard Page offers a comprehensive view of newly added items, providing users and administrators with valuable insights into the system's data. Throughout the system, the use of cookies and local storage enhances data persistence, enabling a seamless and efficient user experience. Additionally, the integration of PHP and MySQL facilitates database interactions, ensuring the secure and structured storage of information.
* In terms of the user interface, thoughtful design elements, such as colorful buttons and containers, contribute to an aesthetically pleasing and engaging environment. The Shopping Cart entity plays a crucial role in the system, allowing customers to manage their selected items before completing a purchase, enhancing the overall shopping experience.
* Overall, the Business Management System demonstrates a holistic approach to managing items, transactions, and user interactions. The integration of various technologies, from front-end development to back-end functionalities, results in a cohesive and user-friendly system that meets the needs of both customers and administrators. The emphasis on data organization, security, and a visually appealing interface collectively contributes to a robust and effective Business Management System.

**CHAPTER 8**

**FUTURE WORK:**

**1. User Authentication and Authorization:** Implement a robust user authentication system to secure user accounts. Introduce roles and permissions to differentiate between regular users and administrators, ensuring that sensitive functionalities are accessible only to authorized personnel.

**2. Dynamic Product Categories**: Enhance the Product Category entity to support dynamic creation and modification of categories. This could involve allowing administrators to add, edit, or remove categories based on evolving business needs.

**3. Interactive Dashboard:** Expand the Dashboard page to provide more interactive features, such as data visualizations, search functionalities, and sorting options. This would empower administrators with a more comprehensive understanding of item trends and user activities.

**4. Order History and Tracking:** Develop a feature that maintains a comprehensive order history for customers. Include order tracking capabilities, allowing users to monitor the status and shipment details of their purchases.

**5. Payment Gateway Integration:** Integrate a secure payment gateway to enable real transactions. This would involve connecting the system to external payment services to facilitate actual financial transactions and enhance the system's practical utility.

**6. User Feedback and Ratings:** Implement a feedback and rating system for products. Allow customers to leave reviews and ratings for items they've purchased, providing valuable insights for other users and enhancing the credibility of the platform.

**7. Responsive Design:** Optimize the system's design for responsiveness across various devices, including mobile phones and tablets. This ensures a consistent and user-friendly experience regardless of the device used to access the system.

**8. Enhanced Security Measures:** Strengthen security measures by incorporating encryption protocols, input validation, and additional layers of security to protect against potential vulnerabilities and data breaches.

**9. Notification System:** Implement a notification system to keep users informed about order confirmations, shipment updates, and other relevant events. This enhances user engagement and provides a more transparent and communicative platform.

**10. Localization and Globalization:** Extend the system's reach by implementing localization and globalization features. This would involve adapting the system to support multiple languages and currencies, making it accessible to a more diverse user base.

**CHAPTER 9**

**REFERENCE**

* [HTML Styles CSS (w3schools.com)](https://www.w3schools.com/html/html_css.asp)
* [HTML & CSS - GeeksforGeeks](https://www.geeksforgeeks.org/web-technology/html-css/)
* [PHP Tutorial (w3schools.com)](https://www.w3schools.com/php/default.asp)
* [Learn PHP Tutorial - javatpoint](https://www.javatpoint.com/php-tutorial)

|  |  |
| --- | --- |
| **PERFORMANCE** |  |
| **VIVAVOCE** |  |
| **MINI PROJECT** |  |
| **TOTAL** |  |