

North South University
Department of Electrical & Computer Engineering
Project Final Report

Course Code: CSE 327

Course Title: Software Engineering

Course Faculty: Mr. AKM Iqtidar Newaz [IqN]

Project Name:

**Digital Tech Hub (Hardware and
Electronics Selling Website system.)**

Date of Submission: 16/12/2025

Section: 06

Group Number: 04

Submitted To: Mr. AKM Iqtidar Newaz [IqN]

Submitted By

Member 1: Md Rakibul Hasan **ID:** 2212346042

Member 2: Md. Rokib Hasan Oli **ID:** 2211950642

Member 3: Tirtho Mojumdar **ID:** 2312536048

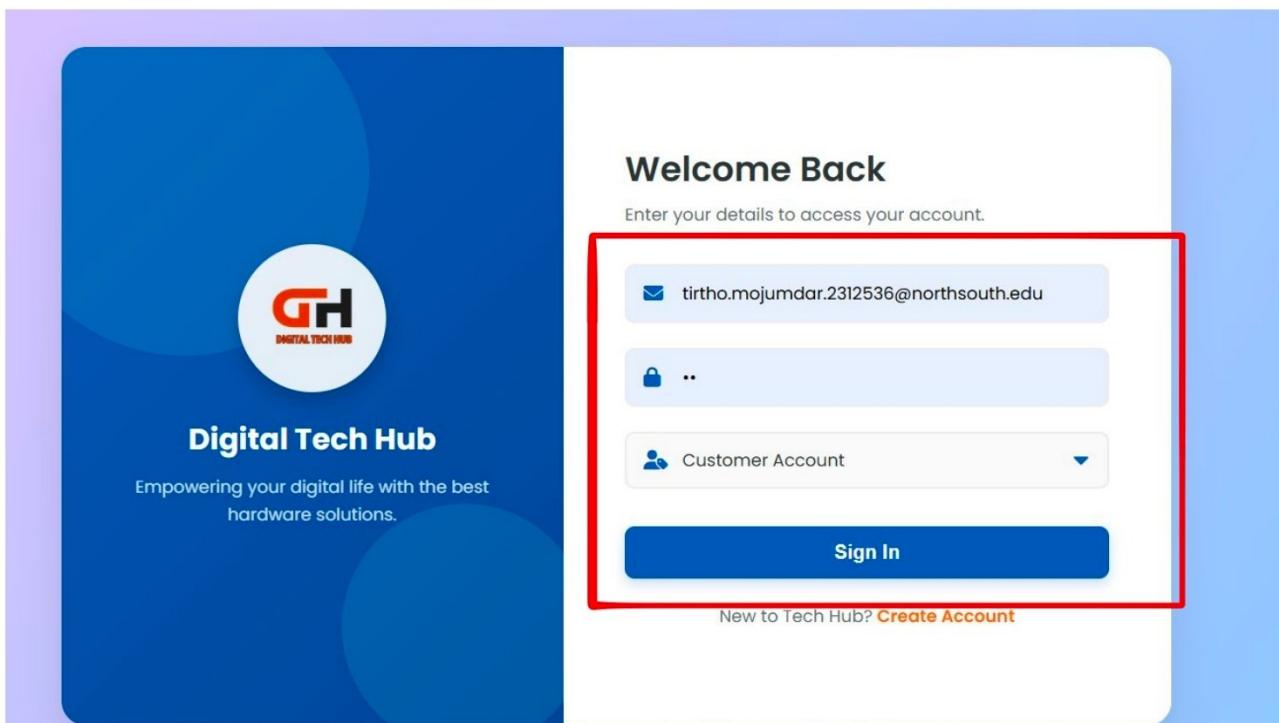
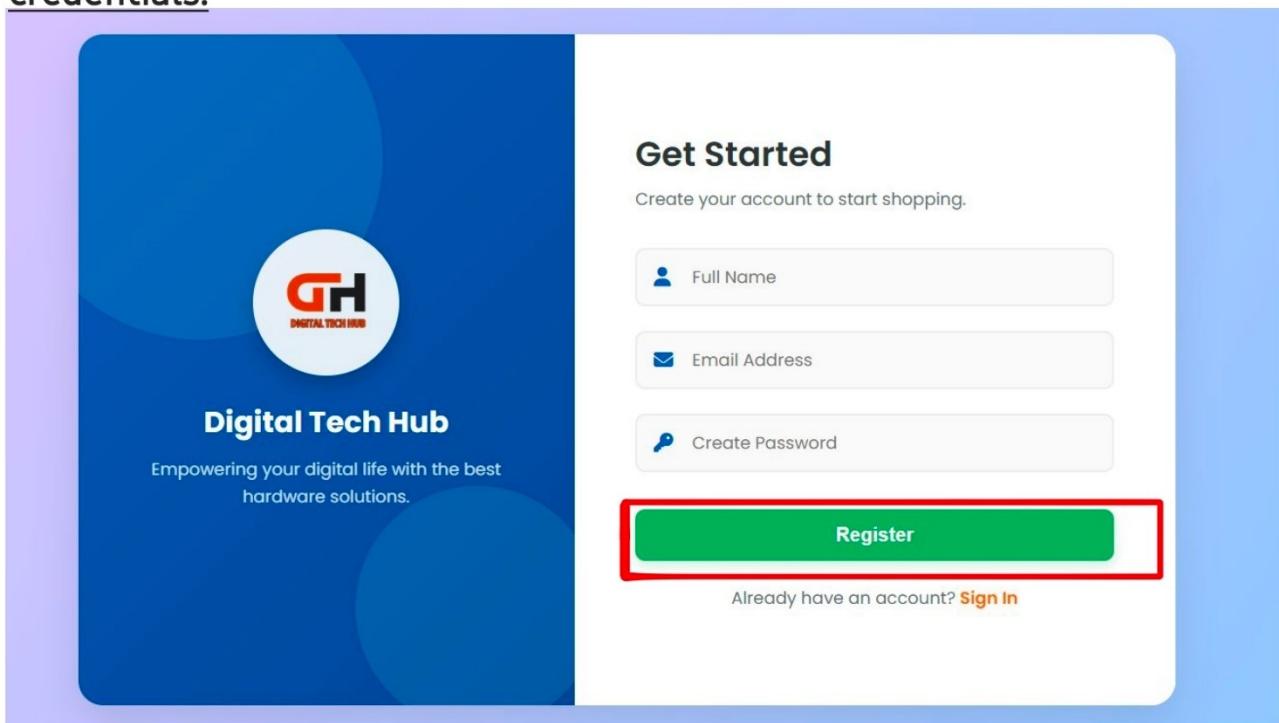
Member 4: Sabbir Ahmod **ID:** 2311501042

Git Repository: https://github.com/Rokib-Hasan-Oli/Digital_Tech_Hub_--CSE327_Sec-6/tree/main

TABLE OF CONTENTS

1. PROPOSED AND COMPLETED FUNCTIONAL REQUIREMENTS	03
FRU: User Functionalities Requirements/Specifications [Screenshots of Code + UI]	
FRA: Admin Functionalities Requirements/Specifications [Screenshots of Code + UI]	
2. PROPOSED AND COMPLETED NON-FUNCTIONAL REQUIREMENTS	25
2.1 Performance Specifications (NFR-P1: Load Time)	
2.2 Security Specifications (NFR-S1: Password Hashing)	
2.3 Security Specifications (NFR-S2: Input Validation)	
2.4 Usability Specifications (NFR-U1: Intuitive Interface)	
2.5 Usability Specifications (NFR-U2: Responsiveness)	
2.6 Reliability Specifications (NFR-R1: Data Integrity)	
2.7 Maintainability Specifications (NFR-M1: Documentation & Standards)	
3. IMPLEMENTATION OF DESIGN PATTERNS	33
3.1 Singleton Pattern (Database Connection)	
3.2 Factory Pattern (Payment Processing)	
3.3 Facade Pattern (Order Processing)	
3.4 Observer Pattern (Notification System)	
3.5 Strategy Pattern (Discounts)	
3.6 Decorator Pattern (Product Warranty)	
3.7 Proxy Pattern (Admin Security)	
4. DIAGRAM VS. IMPLEMENTATION ANALYSIS	41
4.1 Class Diagram Analysis	
4.2 Sequence Diagram Analysis	

FRU.0: The system shall allow users to register and log in using credentials.



```

13
14 // =====
15 // 1. HANDLE REGISTER (FIXED)
16 // =====
17 if (isset($_POST['register'])) {
18     $name = $db->real_escape_string($_POST['name']);
19     $email = $db->real_escape_string($_POST['email']);
20     $pass = password_hash($_POST['password'], PASSWORD_DEFAULT);
21
22     // --- FIX START: Check if email already exists ---
23     $check_query = "SELECT * FROM users WHERE email = '$email'";
24     $check_result = $db->query($check_query);
25
26     if ($check_result->num_rows > 0) {
27         // Email found, show red error message
28         $msg = "This account already exists on your device";
29         $msg_type = "error";
30     } else {
31         // Email is new, proceed with registration
32         $sql = "INSERT INTO users (full_name, email, password) VALUES ('$name', '$email', '$pass')";
33         if ($db->query($sql)) {
34             $msg = "Registration Successful! Please Login.";
35             $msg_type = "success";
36         } else {
37             $msg = "Error: " . $db->error;
38             $msg_type = "error";
39         }
40     }
41     // --- FIX END ---
42 }

```

```

47 if (isset($_POST['login'])) {
48     $email = $db->real_escape_string($_POST['email']);
49     $pass = $_POST['password'];
50     $role = $_POST['role'];
51
52     if ($role == 'admin') {
53         $res = $db->query("SELECT * FROM admins WHERE email='$email'");
54         $row = $res->fetch_assoc();
55
56         // Check password (supports both plain text sample data & hashed real data)
57         if ($row && ($pass === $row['password'] || password_verify($pass, $row['password']))) {
58             $_SESSION['role'] = 'admin';
59             $_SESSION['user_id'] = $row['admin_id'];
60             header("Location: admin_dashboard.php");
61             exit();
62         } else {
63             $msg = "Invalid Admin Credentials";
64             $msg_type = "error";
65         }
66     } else {
67         $res = $db->query("SELECT * FROM users WHERE email='$email'");
68         $row = $res->fetch_assoc();
69
70         if ($row && password_verify($pass, $row['password'])) {
71             if ($row['status'] == 'Blocked') {
72                 $msg = "Access Denied: Your account is blocked.";
73                 $msg_type = "error";
74             } else {
75                 $_SESSION['role'] = 'user';
76                 $_SESSION['user_id'] = $row['user_id'];
77                 $_SESSION['name'] = $row['full_name'];
78                 $_SESSION['photo'] = $row['profile_image'];
79                 header("Location: user_dashboard.php");
80                 exit();
81             }
82         }
83     }
84 }

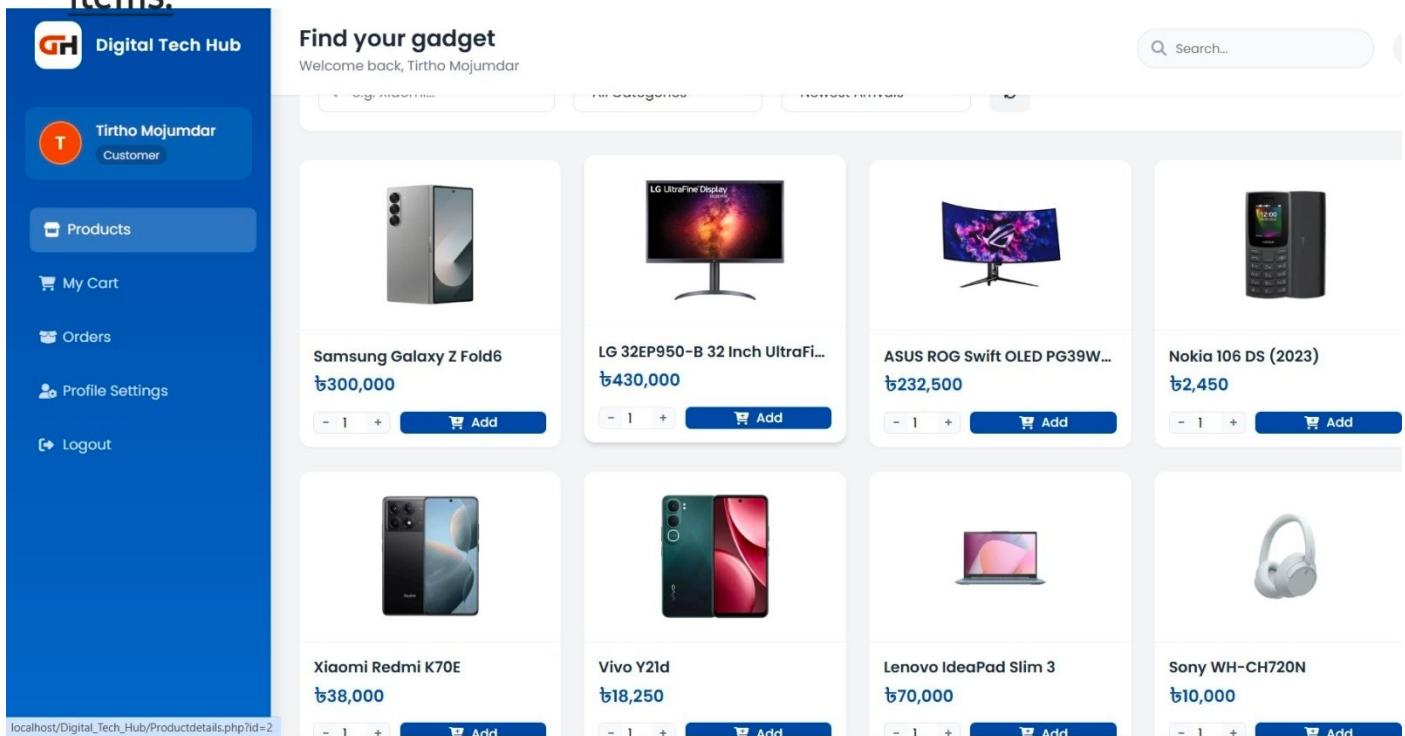
```

```

81     }
82     } else {
83         $msg = "Invalid Email or Password";
84         $msg_type = "error";
85     }
86 }
87 ?>
88
89

```

FRU.1: The system shall allow users to view all Hardware and Electronics items.

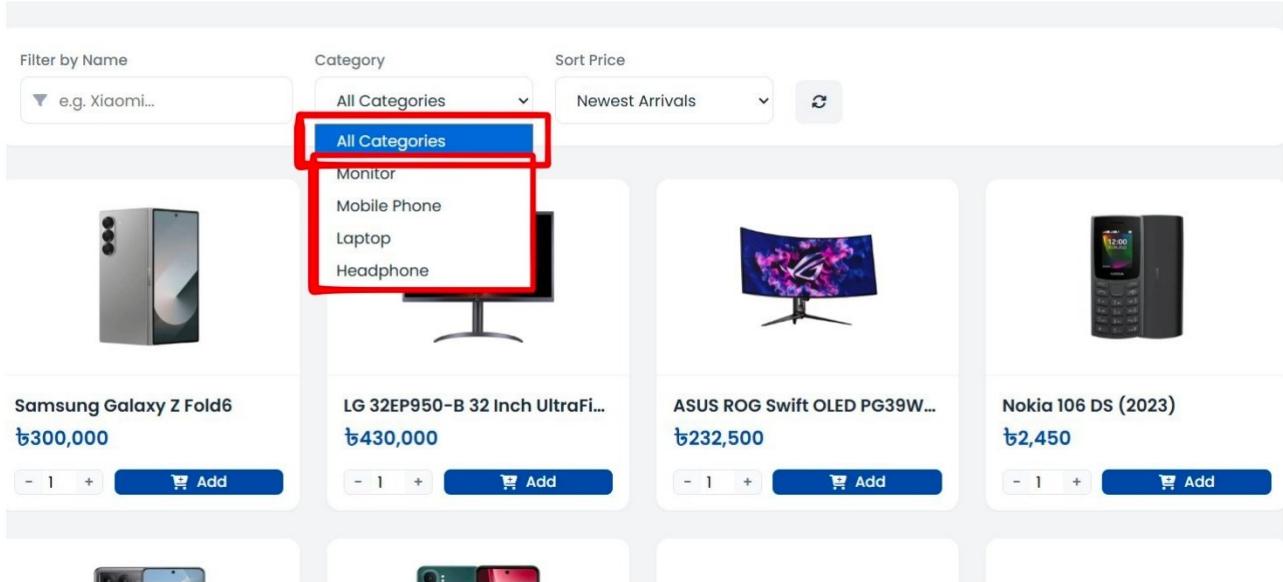


The screenshot shows a user interface for a shopping platform. On the left is a sidebar for 'Digital Tech Hub' with a logo, user info (Tirtho Mojumdar, Customer), and navigation links for Products, My Cart, Orders, Profile Settings, and Logout. The main area has a header 'Find your gadget' and a search bar. Below is a grid of eight product cards:

Product Image	Product Name	Price	Action
	Samsung Galaxy Z Fold6	৳300,000	Add
	LG 32EP950-B 32 Inch UltraFi...	৳430,000	Add
	ASUS ROG Swift OLED PG39W...	৳232,500	Add
	Nokia 106 DS (2023)	৳2,450	Add
	Xiaomi Redmi K70E	৳38,000	Add
	Vivo Y21d	৳18,250	Add
	Lenovo IdeaPad Slim 3	৳70,000	Add
	Sony WH-CH720N	৳10,000	Add

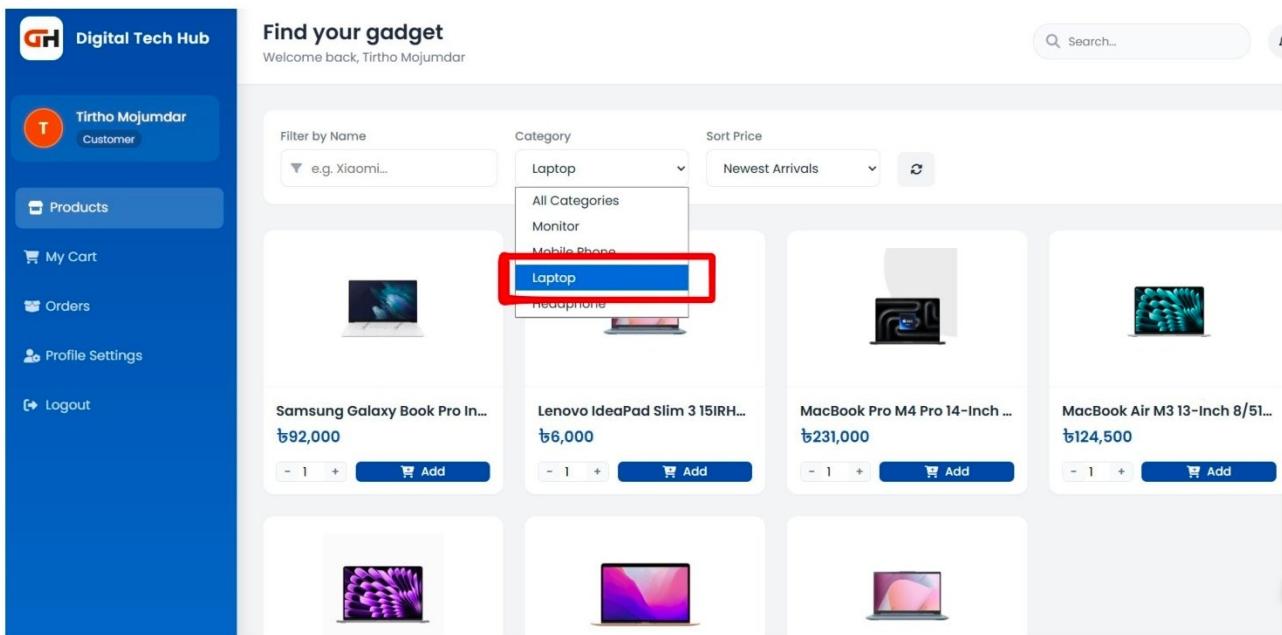
```
32 if (!isset($_GET['view']) || $_GET['view'] == 'products') {  
33     $cat_query = "SELECT * FROM categories";  
34     $categories = $db->query($cat_query);  
35  
36     // Filter System  
37     $conditions = [];  
38     $order_sql = "";  
39
```

FRU.2: The system shall allow users to view all product categories



```
33     $cat_query = "SELECT * FROM categories";
34     $categories = $db->query($cat_query);
35
```

FRU.3: The system shall allow users to view products filtered by category



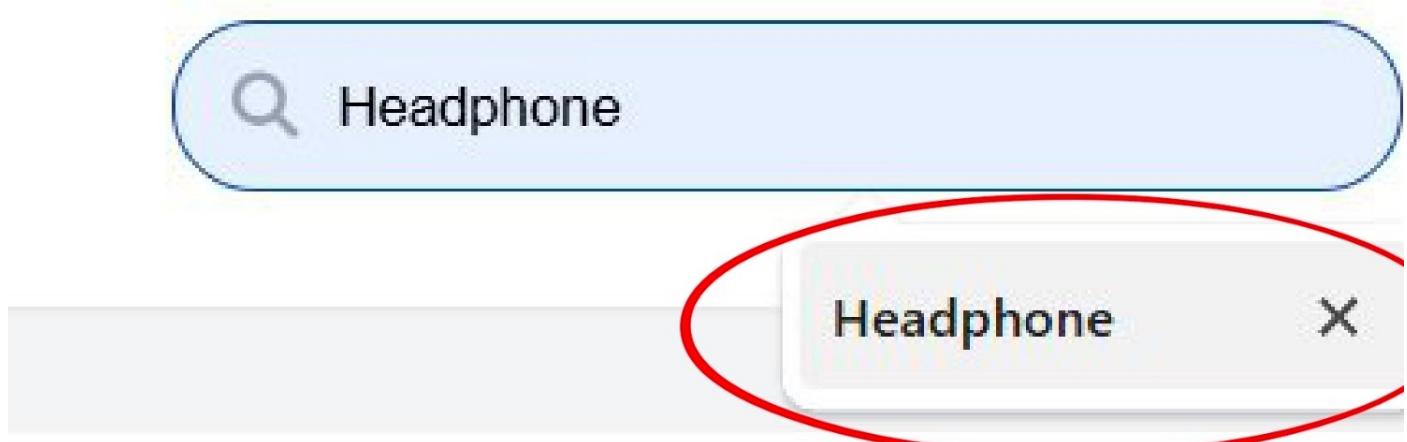
The screenshot shows a user interface for a shopping website called "Digital Tech Hub". On the left is a vertical sidebar with a blue background, displaying a user profile for "Tirtho Mojumdar" (Customer), navigation links for "Products", "My Cart", "Orders", "Profile Settings", and "Logout", and a search bar. The main content area has a white background. At the top, it says "Find your gadget" and "Welcome back, Tirtho Mojumdar". A search bar with a placeholder "Search..." is on the right. Below that is a filter section with three dropdown menus: "Filter by Name" (with a placeholder "e.g. Xiaomi..."), "Category" (set to "Laptop", with a dropdown menu showing "All Categories", "Monitor", and "Mobile Phone"), and "Sort Price" (set to "Newest Arrivals"). The main area displays a grid of four laptop products:

Product Image	Product Name	Price	Action
	Samsung Galaxy Book Pro In...	₹92,000	Add
	Lenovo IdeaPad Slim 3 15IRH...	₹6,000	Add
	MacBook Pro M4 Pro 14-Inch ...	₹231,000	Add

The "Category" dropdown menu is open, and the "Laptop" option is highlighted with a red box.

```
44
45     if (!empty($_GET['category'])) {
46         $cat_id = (int)$_GET['category'];
47         $conditions[] = "category_id = '$cat_id'";
48     }
49
```

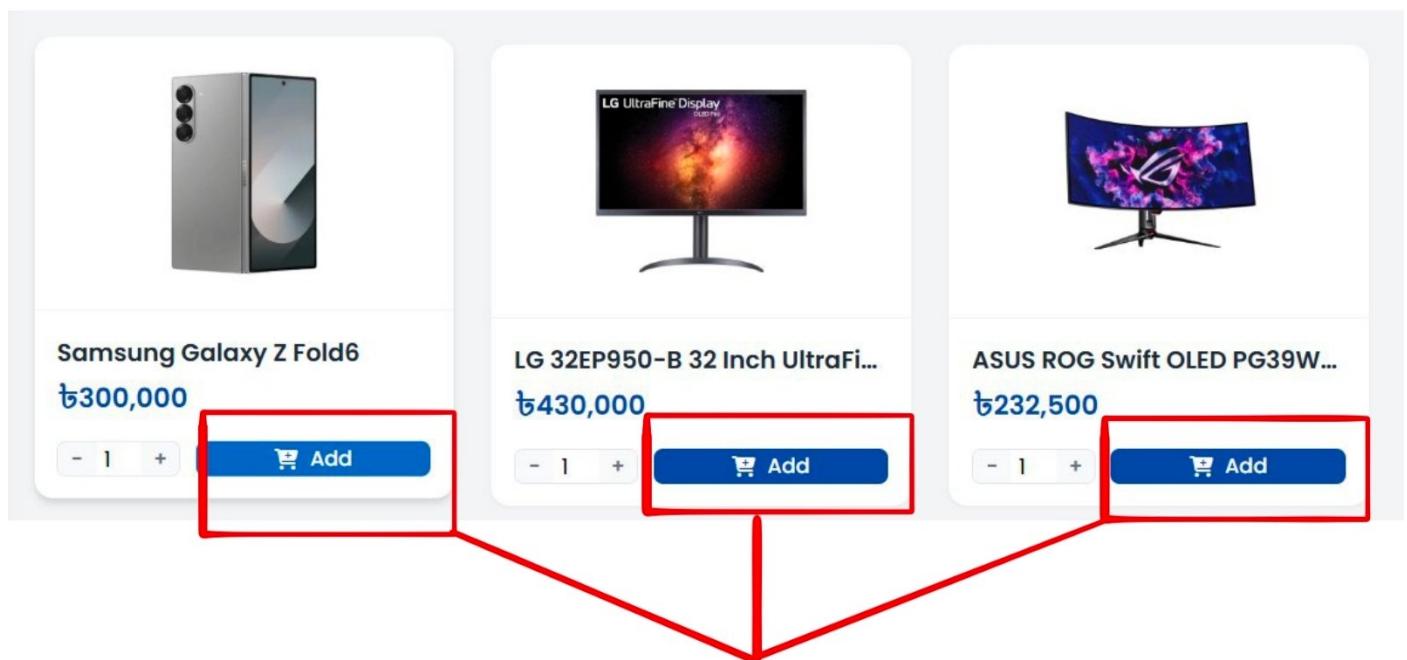
FRU.4: The system shall support a search function to find products



```
35
36     // Filter System
37     $conditions = [];
38     $order_sql = "";
39
40     if (!empty($_GET['search'])) {
41         $s = $db->real_escape_string($_GET['search']);
42         $conditions[] = "name LIKE '%$s%'";
43     }
44
```

13/28

FRU.5: The system shall allow users to add items to a shopping cart and view the cart contents.

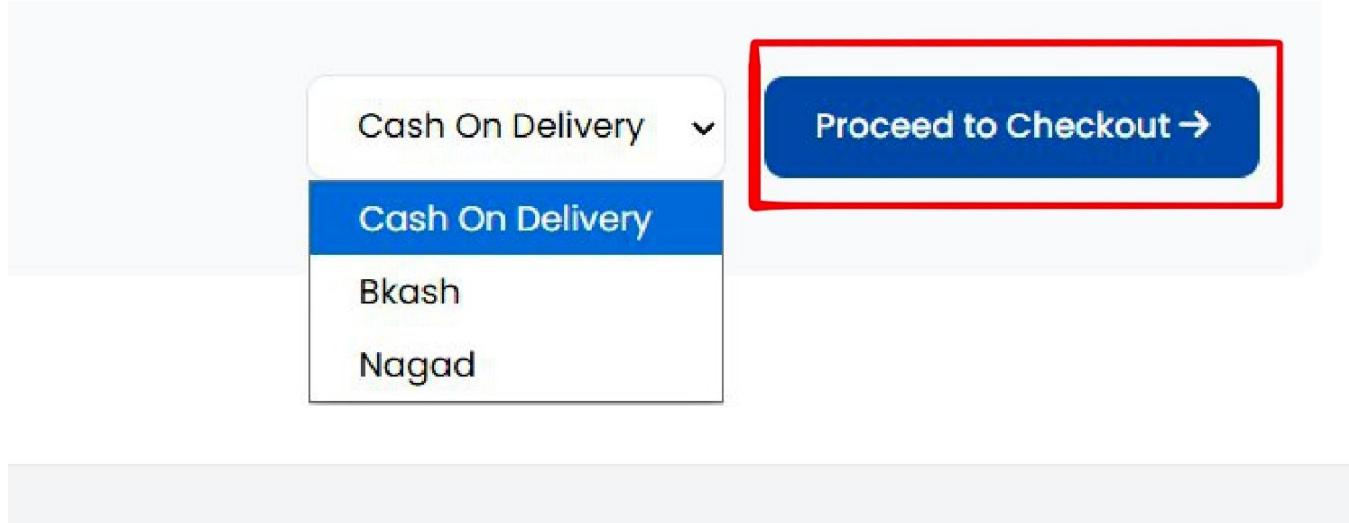


```
68 if (isset($_POST['add_cart'])) {
69     if (!isset($_SESSION['cart'])) {
70         $_SESSION['cart'] = [];
71     }
72     $_SESSION['cart'][] = [
73         'id'    => $_POST['pid'],
74         'name'  => $_POST['pname'],
75         'price' => $_POST['pprice'],
76         'qty'   => $_POST['qty']
77     ];
78     echo "<script>alert('Added to Cart');</script>";
79 }
```

```
178 if (isset($_POST['checkout']) && isset($_SESSION['cart'])) {
179     $facade = new OrderFacade();
180
181     $uid    = $_SESSION['user_id'];
182     $amount = $_POST['pay_amount'];
183     $method = $_POST['method'];
184     $items  = $_SESSION['cart'];
185
186     // Place order
187     $orderId = $facade->placeOrder($uid, $amount, $method, $items);
188     unset($_SESSION['cart']);
189
190     //  FIX: update order & payment status after successful payment
191     if (strtolower($method) !== 'cod') {
192         $db->query("
193             UPDATE orders
194                 SET payment_status = 'Paid',
195                     order_status = 'Processing'
196                 WHERE order_id = $orderId
197         ");
198     }
199 }
200
201 $products = $db->query("
202     SELECT p.name, oi.quantity, oi.unit_price
203     FROM order_items oi
204     JOIN products p ON p.product_id = oi.product_id
205     WHERE oi.order_id = $orderId
206 ");
```

16/28

FRU.6: The system shall facilitate a checkout process.



```
178 if (isset($_POST['checkout']) && isset($_SESSION['cart'])) {  
179     $facade = new OrderFacade();  
180  
181     $uid    = $_SESSION['user_id'];  
182     $amount = $_POST['pay_amount'];  
183     $method = $_POST['method'];  
184     $items  = $_SESSION['cart'];  
185  
186     // Place order  
187     $orderId = $facade->placeOrder($uid, $amount, $method, $items);  
188     unset($_SESSION['cart']);  
189  
190     //  FIX: update order & payment status after successful payment  
191     if (strtolower($method) !== 'cod') {  
192         $db->query(" UPDATE orders  
193             SET payment_status = 'Paid',  
194                 order_status = 'Processing'  
195             WHERE order_id = $orderId  
196         ");  
197     }  
198  
199     $products = $db->query(" SELECT p.name, oi.quantity, oi.unit_price  
200             FROM order_items oi  
201             JOIN products p ON p.product_id = oi.product_id  
202             WHERE oi.order_id = $orderId  
203         ");  
204  
205     $badge = strtolower($method);  
206  
207     ?>
```

FRU.7: The system shall offer online payment methods (e.g., Bkash, Nagad).

Cash On Delivery

Cash On Delivery

Bkash

Nagad

```
1  <?php
2  require_once 'Database.php';
3  interface IPayment {
4      public function pay($amount);
5  }
6
7  class BkashPayment implements IPayment {
8      public function pay($amount) { return "Paid $amount Tk via Bkash (Trans ID: BK-" . rand(1000,9999) . ")"; }
9  }
10
11 class NagadPayment implements IPayment {
12     public function pay($amount) { return "Paid $amount Tk via Nagad (Trans ID: NG-" . rand(1000,9999) . ")"; }
13 }
14
15 class CODPayment implements IPayment {
16     public function pay($amount) { return "Amount $amount Tk to be paid on Delivery."; }
17 }
18
19 class PaymentFactory {
20     public static function create($method) {
21         if ($method == 'Bkash') return new BkashPayment();
22         if ($method == 'Nagad') return new NagadPayment();
23         return new CODPayment(); // Default
24     }
25 }
26 ?>
```

FRU.8: The system shall allow users to view their order history and the current status of each order.

The screenshot shows a user profile header with a circular icon containing a 'T', the name 'Tirtho Mojumdar', and the title 'Customer'. Below the header is a sidebar with links: 'Products', 'My Cart', a highlighted 'Orders' link, 'Profile Settings', and 'Logout'. The main content area is titled 'Order History' and displays a table of eight orders. The columns are: ORDER ID, DATE, METHOD, PAYMENT STATUS, STATUS, and TOTAL. The data is as follows:

ORDER ID	DATE	METHOD	PAYMENT STATUS	STATUS	TOTAL
#11	15 Dec 2025	COD	Pending	Pending	₹600,000.00
#10	15 Dec 2025	COD	Pending	Pending	₹300,000.00
#9	13 Dec 2025	Bkash	Paid	Processing	₹300,000.00
#8	13 Dec 2025	Nagad	Paid	Processing	₹300,000.00
#7	13 Dec 2025	COD	Pending	Pending	₹300,000.00
#6	13 Dec 2025	Nagad	Paid	Processing	₹430,000.00
#5	13 Dec 2025	COD	Pending	Pending	₹300,000.00

```
144 <li>
145   <a href="?view=orders" class="<?php echo (isset($_GET['view']) && $_GET['view'] == 'orders') ? 'active' : ''; ?>">
146     <i class="fa fa-box-open"></i> Orders
147   </a>
</li>
```

FRU.9: The system shall provide Profile Management allowing users to edit their profile/address and change their password.

The screenshot displays a user interface for managing account information. On the left, a sidebar menu includes links for Shop, My Cart, Orders, and Profile Settings, with Profile Settings being the active tab. The main content area is titled 'Profile Information' and contains several input fields: 'Profile Picture' (with a placeholder image and a 'Choose Photo' button), 'Full Name' (Tirtho Mojumdar), 'Email Address' (tirtho.mojumdar.2312536@northsouth.edu), 'Phone Number' (+880 1234-567890), 'Member Since' (December 2025), and 'Address' (Enter your full address).

```
1 <?php
2 session_start();
3 require_once 'Database.php';
4
5 // Auth Check
6 if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'user') {
7     header("Location: login_register.php");
8     exit();
9 }
10
11 $db = Database::getInstance()->getConnection();
12 $logo = "https://ln.run/WLP6-";
13 $user_id = $_SESSION['user_id'];
14 $success_msg = "";
15 $error_msg = "";
16
17 // Fetch current user data
18 $user_sql = "SELECT * FROM users WHERE user_id = $user_id";
19 $user_result = $db->query($user_sql);
20 $user_data = $user_result->fetch_assoc();
21
22 // Handle Profile Update
23 if (isset($_POST['update_profile'])) {
24     $full_name = $db->real_escape_string($_POST['full_name']);
25     $email = $db->real_escape_string($_POST['email']);
26     $phone = $db->real_escape_string($_POST['phone'] ?? '');
27     $address = $db->real_escape_string($_POST['address'] ?? '');
28
29 // Handle profile photo upload
30 $profile_image = $user_data['profile_image'] ?? 'default-avatar.png';
31
32 if (isset($_FILES['profile_image']) && $_FILES['profile_image']['error'] == 0) {
33     $upload_dir = "uploads/profiles/";
34     if (!file_exists($upload_dir)) {
35         mkdir($upload_dir, 0777, true);
36     }
37 }
```

```

38     $file_ext = strtolower(pathinfo($_FILES['profile_image']['name'], PATHINFO_EXTENSION));
39     $allowed_ext = ['jpg', 'jpeg', 'png', 'gif'];
40
41     if (in_array($file_ext, $allowed_ext)) {
42         $new_filename = "user_" . $user_id . "_" . time() . "." . $file_ext;
43         $target_file = $upload_dir . $new_filename;
44
45         if (move_uploaded_file($_FILES['profile_image']['tmp_name'], $target_file)) {
46             $profile_image = $new_filename;
47         }
48     }
49 }
50
51 // Update database
52 $update_sql = "UPDATE users SET
53     full_name = '$full_name',
54     email = '$email',
55     phone = '$phone',
56     address = '$address',
57     profile_image = '$profile_image'
58 WHERE user_id = $user_id";
59
60 if ($db->query($update_sql)) {
61     $_SESSION['name'] = $full_name;
62     $_SESSION['email'] = $email;
63     $success_msg = "Profile updated successfully!";
64
65     // Refresh user data
66     $user_result = $db->query($user_sql);
67     $user_data = $user_result->fetch_assoc();
68 } else {
69     $error_msg = "Failed to update profile.";
70 }
71 }
72

```

```

73 // Handle Password Change
74 if (isset($_POST['change_password'])) {
75     $current_password = $_POST['current_password'];
76     $new_password = $_POST['new_password'];
77     $confirm_password = $_POST['confirm_password'];
78
79     // FIX 1: Use password_verify() to check the hash, not direct comparison
80     if (password_verify($current_password, $user_data['password'])) {
81
82         if ($new_password == $confirm_password) {
83             if (strlen($new_password) >= 6) {
84
85                 // FIX 2: Hash the NEW password before saving it to the database
86                 // If you don't do this, you won't be able to login next time!
87                 $hashed_password = password_hash($new_password, PASSWORD_DEFAULT);
88
89                 $update_pass = "UPDATE users SET password = '$hashed_password' WHERE user_id = $user_id";
90
91                 if ($db->query($update_pass)) {
92                     $success_msg = "Password changed successfully!";
93
94                         // Optional: Update the local user_data so consecutive changes don't fail immediately
95                         $user_data['password'] = $hashed_password;
96                 } else {
97                     $error_msg = "Failed to change password.";
98                 }
99             } else {
100                 $error_msg = "Password must be at least 6 characters.";
101             }
102         } else {
103             $error_msg = "New passwords do not match.";
104         }
105     } else {
106         $error_msg = "Current password is incorrect.";
107     }
108 }
109 ?>

```

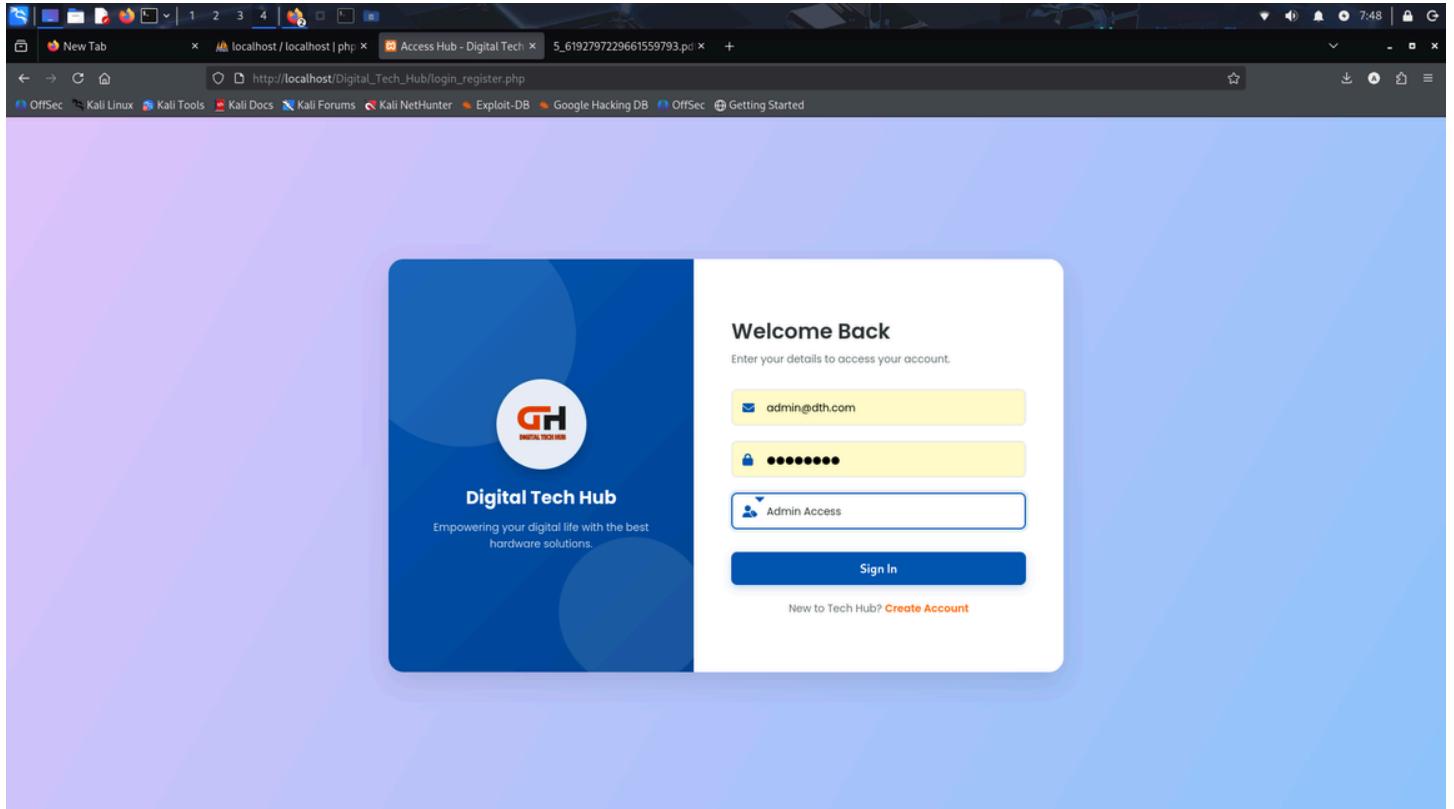
FRU.10: The system shall allow users to log out.

The screenshot shows a user interface for an e-commerce platform. On the left, a sidebar for a user named 'Tirtho Mojumdar' (Customer) lists navigation options: Products, My Cart, Orders, Profile Settings, and Logout. The 'Logout' button is highlighted with a red box. The main area displays a search bar ('Filter by Name') and a category dropdown ('Category: All Categories'). Below this, two products are listed: a Samsung Galaxy Z Fold6 (silver, triple-camera) and an LG 32EP950-B 32 inch monitor (black, curved screen). Each product card includes a price (₹300,000 and ₹430,000 respectively), quantity selector (- 1 +), and an 'Add' button.

```
C:\wamp\www\index.php - DigitalElectronics - log_out.php
1 <?php
2 session_start();
3 session_destroy();
4 header("Location: login_register.php");
5 ?>
```

Admin Functionalities (Owner-Facing)

- FRA.O: The system shall allow the Admin to log in

A screenshot of the Digital Tech Hub Admin Dashboard. The left sidebar shows navigation links for "Administrator", "Products", "Categories", "Users", "Orders", and "Logout". The main content area has two sections: "Add New Product" (with fields for Product Name, Category, Price, Stock Quantity, Description, Warranty, and Product Image) and "Product List" (a table showing three keyboard products: AJAZZ Ak980, T-Wolf T70, and Rapoo V500PRO-98).

ID	IMAGE	NAME	CATEGORY	PRICE	STOCK	ACTION
#23		AJAZZ Ak980 Wired Rainbow Hot Swap Mechanical Keyboard	Keyboard	3500.00	76	
#22		T-Wolf T70 Tri Mode RGB Mechanical Gaming Keyboard	Keyboard	3299.00	100	
#21		Rapoo V500PRO-98 Red Switch RGB Gaming Mechanical Keyboard	Keyboard	3350.00	198	

```
1 <?php
2 session_start();
3 require_once 'AdminProxy.php';
4
5 // 1. PROXY PATTERN: Security Check
6 $proxy = new AdminProxy();
7 $proxy->render();
```

- FRA.1: The system shall allow the Admin to Add, Edit, and Delete Hardware and Electronics categories.

The screenshot shows a web-based administration interface for a digital inventory system. On the left, a sidebar menu includes 'Administrator System Control', 'Products' (selected), 'Categories', 'Users', 'Orders', and 'Logout'. The main content area has two sections: 'Add New Product' and 'Product List'.

Add New Product: This section contains fields for Product Name (e.g. iPhone 15), Category (Monitor selected), Price (Tk) (0.00), Stock Quantity (0), Description (Product details...), a checkbox for 'Include 1 Year Official Warranty (+2000 Tk)', and a file upload field for 'Product Image' (No file selected). A blue 'Add Product' button is at the bottom.

Product List: This section displays a table of three keyboard products:

ID	IMAGE	NAME	CATEGORY	PRICE	STOCK	ACTION
#23		AJAZZ Ak980 Wired Rainbow Hot Swap Mechanical Keyboard	Keyboard	3500.00	76	Delete
#22		T-Wolf T70 Tri Mode RGB Mechanical Gaming Keyboard	Keyboard	3299.00	100	Delete
#21		Rapoo V500PRO-98 Red Switch RGB Gaming Mechanical Keyboard	Keyboard	3350.00	198	Delete

Code Snippet:

```

if (isset($_POST['add_cat'])) {
    $name = $db->real_escape_string($_POST['cat_name']);
    $desc = $db->real_escape_string($_POST['cat_desc']);
    if ($db->query("INSERT INTO categories (name, description) VALUES ('$name', '$desc')")) {
        $msg = "Category Added Successfully.";
    } else {
        $msg = "Error: " . $db->error;
    }
}
if (isset($_GET['delete_cat'])) {
    $id = intval($_GET['delete_cat']);
    $db->query("DELETE FROM categories WHERE category_id=$id");
    $msg = "Category Deleted.";
}

```

- FRA.2: The system shall allow the Admin to Add, Edit, and Delete Hardware and Electronics items (inventory management).

Screenshot of a web-based digital inventory management system (Admin Dashboard) running on a Kali Linux environment.

Header: Admin Dashboard | Digital Tech Hub - http://localhost/Digital_Tech_Hub/admin_dashboard.php?view=categories

Left Sidebar (Administrator System Control):

- Products
- Categories (selected)
- Users
- Orders
- Logout

Main Content Area:

Add Category

Category Name: e.g. Laptops

Description: Category details...

Existing Categories:

ID	NAME	DESCRIPTION	ACTION
#3	Monitor	Explore our extensive selection of high-performance monitors designed to meet every need.	
#4	Mobile Phone	Latest smartphones designed to connect your world.	
#5	Tablet	Tablet PCs or "Tabs" are modern-day super compact computers with touchscreen. These bridge a phone and laptop, with the perks of both. Tablets are typically operated with fingers and gestures. iPad OS and Android OS-based Tabs are the most popular. From Graphics Designers, and home users to kids - tabs are great as personal devices. Graphics Tabs and Drawing Tablets are also popular for professional-level sketching and 3D modeling.	
#6	Camera	Cameras play a vital role in both professional and personal photography, making them essential companions for photography enthusiasts, content creators, and professionals in Bangladesh. The latest Camera models from popular brands like Canon, Nikon, Sony, and Fujifilm are readily available at Star Tech. Whether you're capturing stunning weddings, creating engaging vlogs, documenting news events, or shooting breathtaking landscapes, Star Tech offers the latest camera price in Bangladesh with a wide range of options to suit every need and budget. From entry-level DSLRs for beginners to high-end mirrorless cameras for professionals, discover the best camera prices in Bangladesh at Star Tech, your one-stop shop for top-quality photography gear.	
#7	Keyboard	Computer keyboard is an input device that has arrangement of buttons or keys with printed characters & symbols are usually function through single press for typing text and numbers, actions	

```

if (isset($_POST['add_product'])) {
    $name = $db->real_escape_string($_POST['name']);
    $desc = $db->real_escape_string($_POST['description']);
    $price = $_POST['price'];
    $stock = $_POST['stock'];
    $cat_id = $_POST['category_id'];
    $has_warranty = isset($_POST['has_warranty']) ? 1 : 0;
    // Image Upload Handling
    $image = "default_product.png";
    if (isset($_FILES['product_image']['name']) && $_FILES['product_image']['name'] != "") {
        $target_dir = "uploads/";
        if (!file_exists($target_dir)) { mkdir($target_dir, 0777, true); }

        $image = basename($_FILES["product_image"]["name"]);
        $target_file = $target_dir . $image;
        move_uploaded_file($_FILES["product_image"]["tmp_name"], $target_file);
    }
    $sql = "INSERT INTO products (category_id, name, description, price, stock_quantity, product_image, has_warranty)
            VALUES ('$cat_id', '$name', '$desc', '$price', '$stock', '$image', '$has_warranty')";

    if ($db->query($sql)) {
        $msg = "Product Added Successfully.";
    } else {
        $msg = "Error: " . $db->error;
    }
}
if (isset($_GET['delete_prod'])) {
    $id = intval($_GET['delete_prod']);
    $db->query("DELETE FROM products WHERE product_id=$id");
    $msg = "Product Deleted.";
}

```

- FRA.3: The system shall allow the Admin to view a list of all registered users.

ID	NAME	EMAIL	STATUS	ACTION
#1	Rokib Hasan	rokib@nsu.com	Active	Block
#2	Rakibul Hasan	rakibul@nsu.com	Active	Block
#3	abc	abc@gmail.com	Active	Block

```

<?php if (isset($_GET['view']) && $_GET['view'] == 'users') { ?>
<div class="panel-section">
    <h3>Manage Users</h3>
    <div class="table-responsive">
        <table class="data-table">
            <thead>
                <tr><th>ID</th><th>Name</th><th>Email</th><th>Status</th><th>Action</th></tr>
            </thead>
            <tbody>
                <?php
                    $res = $db->query("SELECT * FROM users");
                    while ($row = $res->fetch_assoc()) {
                        $statusClass = ($row['status'] == 'Blocked') ? 'status-blocked' : 'status-active';
                        echo "<tr>";
                        echo "<td>" . $row['user_id'] . "</td>";
                        echo "<td>" . $row['full_name'] . "</td>";
                        echo "<td>" . $row['email'] . "</td>";
                        echo "<td><span class='status-badge $statusClass'>" . $row['status'] . "</span></td>";
                        echo "<td>";
                        // UPDATED LOGIC: Display Block or Unblock button
                        if ($row['status'] == 'Blocked') {
                            echo "<a href='?view=users&unblock_user={$row['user_id']}' class='btn-success btn-sm'>Unblock</a>";
                        } else {
                            echo "<a href='?view=users&block_user={$row['user_id']}' class='btn-danger btn-sm'>Block</a>";
                        }
                        echo "</td></tr>";
                    }
                </tbody>
            </table>
        </div>
    </div>
<?php } ?>

```

- FRA.4: The system shall provide the ability for the Admin to block specific users.

User Blocked.

ID	NAME	EMAIL	STATUS	ACTION
#1	Rokib Hasan	rokib@nsu.com	Active	Block
#2	Rakibul Hasan	rakibul@nsu.com	Active	Block
#3	abc	abc@gmail.com	Blocked	Unblock

```
if (isset($_GET['block_user'])) {
    $uid = intval($_GET['block_user']);
    $db->query("UPDATE users SET status='Blocked' WHERE user_id=$uid");
    $msg = "User Blocked.";
}

// 4. Unblock User (GET) - ADDED FOR UNBLOCK FUNCTIONALITY
if (isset($_GET['unblock_user'])) {
    $uid = intval($_GET['unblock_user']);
    $db->query("UPDATE users SET status='Active' WHERE user_id=$uid");
    $msg = "User Unblocked and set to Active.";
}
```

- FRA.5: The system shall allow the Admin to view all submitted orders.

Admin Panel
Manage your digital inventory and users.

Order #1 status updated to **Pending** successfully.

ORDER ID	USER ID	TOTAL	STATUS	ACTION
#1	User #1	300000.00 Tk	Pending	PDF

```
<?php if (isset($_GET['view']) && $_GET['view'] == 'orders') {
    // NEW: Define the possible order statuses
    $order_statuses = ["Pending", "Processing", "Shipped", "Delivered", "Cancelled"];
?>
<div class="panel-section">
    <h3>All Orders</h3>
    <div class="table-responsive">
        <table class="data-table">
            <thead>
                <tr><th>Order ID</th><th>User ID</th><th>Total</th><th>Status</th><th>Action</th></tr>
            </thead>
            <tbody>
                <?php
                    $res = $db->query("SELECT * FROM orders ORDER BY order_id DESC");
                    while ($row = $res->fetch_assoc()) {
                        echo "<tr>";
                        echo "<td>" . $row['order_id'] . "</td>";
                        echo "<td>" . $row['user_id'] . "</td>";
                        echo "<td>" . $row['total_amount'] . " Tk</td>";
                        echo "<td>";
                            echo "<select class='status-select' onchange='window.location.href = \?view=orders&order_id={$row['order_id']}&update_order_status=' + this.value'>";
                            foreach ($order_statuses as $status) {
                                $selected = ($row['order_status'] == $status) ? 'selected' : '';
                                echo "<option value='$status' $selected>$status</option>";
                            }
                            echo "</select>";
                        echo "</td>";
                        echo "<td> <a href='generate_pdf.php?order_id={$row['order_id']}' target='_blank' class='btn-primary btn-sm'><i class='fa fa-file-pdf'>/i PDF</a> </td>";
                    }
                    echo "</tr>";
                }
            </tbody>
        </table>
    </div>
</div> ?>
```

- FRA.6: The system shall allow the Admin to update the status of an order.

Admin Panel
Manage your digital inventory and users.

All Orders				
ORDER ID	USER ID	TOTAL	STATUS	ACTION
#3	User #1	42000.00 Tk	Pending	
#2	User #1	300000.00 Tk	Cancelled	
#1	User #1	300000.00 Tk	Processing	

```

if (isset($_GET['update_order_status']) && isset($_GET['order_id'])) {
    $order_id = intval($_GET['order_id']);
    $new_status = $db->real_escape_string($_GET['update_order_status']);

    // Check if the order_id is valid
    $check = $db->query("SELECT 1 FROM orders WHERE order_id=$order_id");

    if ($check->num_rows > 0) {
        $db->query("UPDATE orders SET order_status='$new_status' WHERE order_id=$order_id");
        $msg = "Order #$order_id status updated to **$new_status** successfully.";
    } else {
        $msg = "Error: Order #$order_id not found.";
    }
}

```

- FRA.7: The system shall enable the Admin to generate customer bills in PDF format.

All Orders

ORDER ID	USER ID	TOTAL	STATUS	ACTION
#3	User #1	42000.00 Tk	Pending	
#2	User #1	300000.00 Tk	Pending	
#1	User #1	300000.00 Tk	Pending	

```

</td>
<td>
    <a href='generate_pdf.php?order_id={$row['order_id']}' target='_blank' class='btn-primary btn-sm'><i class='fa fa-file-pdf'></i> PDF</a>
</td>

```

- FRA.8: The system shall allow the Admin to log out.

Admin Panel
Manage your digital inventory and users.

Add New Product

Product Name <input type="text" value="e.g. iPhone 15"/>	Category <input type="text" value="Monitor"/>	Price (Tk) <input type="text" value="0.00"/>	Stock Quantity <input type="text" value="0"/>
Description <input type="text" value="Product details..."/>			
<input type="checkbox"/> Include 1 Year Official Warranty (+2000 Tk)			
Product Image <input type="button" value="Browse..."/> No file selected.			
<input type="button" value="Add Product"/>			

Product List

ID	IMAGE	NAME	CATEGORY	PRICE	STOCK	ACTION
#23		AJAZZ Ak980 Wired Rainbow Hot Swap Mechanical Keyboard	Keyboard	3500.00	76	<input type="button" value="Edit"/>
#22		T-Wolf T70 Tri Mode RGB Mechanical Gaming Keyboard	Keyboard	3299.00	100	<input type="button" value="Edit"/>
#21		Rapoo V500PRO-98 Red Switch RGB Gaming Mechanical Keyboard	Keyboard	3350.00	198	<input type="button" value="Edit"/>

```

<li><a href="?view=products" class="<?php echo (!isset($_GET['view'])) || $_GET['view'] == 'products' ? 'active' : ''; ?><i class="fa fa-box"></i> Products</a></li>
<li><a href="?view=categories" class="<?php echo (isset($_GET['view'])) && $_GET['view'] == 'categories' ? 'active' : ''; ?><i class="fa fa-list"></i> Categories</a></li>
<li><a href="?view=users" class="<?php echo (isset($_GET['view'])) && $_GET['view'] == 'users' ? 'active' : ''; ?><i class="fa fa-users"></i> Users</a></li>
<li><a href="?view=orders" class="<?php echo (isset($_GET['view'])) && $_GET['view'] == 'orders' ? 'active' : ''; ?><i class="fa fa-shopping-cart"></i> Orders</a></li>
<li class="logout-item"><a href="logout.php"><i class="fa fa-sign-out-alt"></i> Logout</a></li>

```

Non-Functional Requirements/Specifications

2.1 Performance Specifications (NFR-P1: Load Time)

- Proposed Requirement (NFR-P1):** "The website's main pages (homepage, category pages) shall load within **10 seconds** under normal load conditions."
- Completed Implementation:** To achieve optimal performance, we implemented the **Singleton Design Pattern** for database management. This ensures that the application reuses a single database connection instance per request lifecycle instead of opening a new connection for every query, significantly reducing server overhead and latency.

Screenshot of Code :

```
public static function getInstance() {  
    if (self::$instance === null) {  
        self::$instance = new Database();  
    }  
    return self::$instance;  
}
```

Screenshot of UI :

The screenshot shows the Digital Tech Hub website interface. On the left, there's a sidebar with a user profile (Rokib Hasan Oli, Customer), navigation links for Products, My Cart, Orders, Profile Settings, and Logout. The main content area has a search bar and filters for Name and Category. It displays two products: a Samsung Galaxy Z Fold6 and an LG 32EP950-B 32 Inch UltraFine Display. The Network tab of the browser developer tools is open, showing a timeline of requests. The total load time is 950 ms, and the DOMContentLoaded event occurred at 62 ms. The network activity shows several requests for images, CSS files, and fonts, all completed very quickly, indicating a fast loading time.

2.2 Security Specifications (NFR-S1: Password Hashing)

- Proposed Requirement (NFR-S1):** "All user and admin login credentials shall be **hashed** (e.g., using PHP's password hashing functions) and stored securely in the MySQL database."
- Completed Implementation:** We strictly avoided storing plain-text passwords. During user registration, the system uses PHP's **password_hash()** function (BCRYPT algorithm) to encrypt the password before insertion. During login, **password_verify()** is used to authenticate the user against the stored hash.

- **Screenshot of Code :**

```

17 if (isset($_POST['register'])) {
18     $name = $db->real_escape_string($_POST['name']);
19     $email = $db->real_escape_string($_POST['email']);
20     $pass = password_hash($_POST['password'], PASSWORD_DEFAULT);
21     $check_query = "SELECT * FROM users WHERE email = '$email'";
22     $check_result = $db->query($check_query);
23
24     if ($check_result->num_rows > 0) {
25         $msg = "This account already exists on your device";
26         $msg_type = "error";
27     } else {
28         $sql = "INSERT INTO users (full_name, email, password) VALUES ('$name', '$email', '$pass')";
29         if ($db->query($sql)) {
30             $msg = "Registration Successful! Please Login.";
31             $msg_type = "success";
32         } else {
33             $msg = "Error: " . $db->error;
34             $msg_type = "error";
35         }
36     }
}

```

- **Screenshot of UI :**

The screenshot shows two parts of a web application. On the left, there is a table of users with columns: user_id, full_name, email, and password. The data includes:

	user_id	full_name	email	password
<input type="checkbox"/> Edit Copy Delete	1	Rokib Hasan	rokib@nsu.com	\$2y\$10\$XJ6pOsEDj/w3.g2CGVqXpODwGnL6VlOPhlnevAzRYk1...
<input type="checkbox"/> Edit Copy Delete	2	Rakibul Hasan	rakibul@nsu.com	\$2y\$10\$9hOwgt867nQdhY6RISbcgeMiwN4iWd8o5FbpzNThFz3...
<input type="checkbox"/> Edit Copy Delete	3	Md Rakibul Hasan	lol@gmail.com	\$2y\$10\$9Dk/0oORxK1if4NbXOMuZesEjDh.mf3k628A1g63h5d...
<input type="checkbox"/> Edit Copy Delete	10	Md Rakibul Hasan	Rock@gmail.com	\$2y\$10\$8Jyw4Q2ZbS2Xf7OUh5MAMu5hQuvcKSe5lqvErRgVaBu...

On the right, there is a login page titled "Welcome Back". It has fields for "Email" (Rock@gmail.com), "Password" (.....), and a dropdown for "Customer Account". A "Sign In" button is at the bottom. Below the sign-in form, it says "New to Tech Hub? [Create Account](#)".

2.3 Security Specifications (NFR-S2: Input Validation)

- **Proposed Requirement (NFR-S2):** "The system shall employ input validation on all forms (login, registration, profile update, product entry) to prevent SQL Injection and XSS attacks."

- **Completed Implementation:** We implemented defense-in-depth security. For critical transactions, we used Prepared Statements (in **OrderFacade.php**) which separate SQL code from user data. For general form inputs, we used `$db->real_escape_string()` to sanitize characters that could alter SQL commands.
- **Screenshot of Code :**

SQL Injection Prevention:

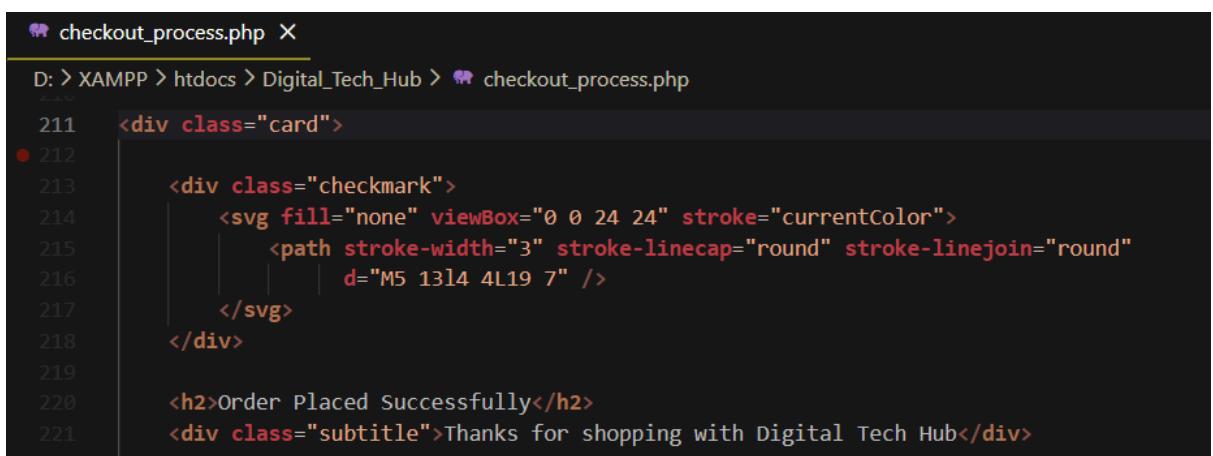
```
if (isset($_POST['register'])) {
    $name = $db->real_escape_string($_POST['name']);
    $email = $db->real_escape_string($_POST['email']);
```

XSS Prevention:

```
<div class="user-info">
    <h3><?php echo htmlspecialchars($_SESSION['name']); ?></h3>
    <span class="badge">Customer</span>
</div>
```

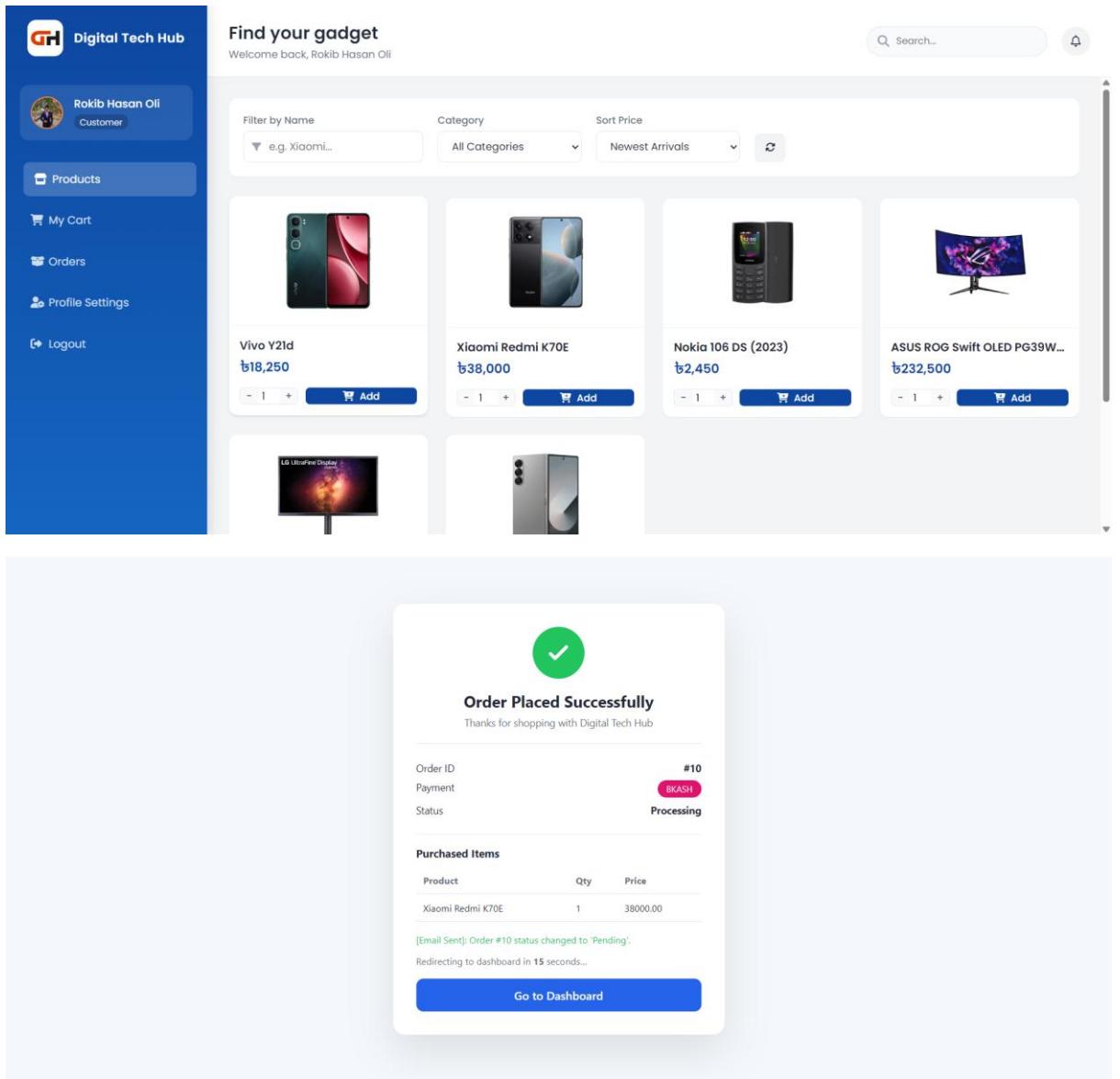
2.4 Usability Specifications (NFR-U1: Intuitive Interface)

- **Proposed Requirement (NFR-U1):** "The User interface shall be intuitive and easy to navigate, allowing first-time users to complete a purchase within 5 minutes of browsing."
- **Completed Implementation:** We designed a clean, grid-based layout for the User Dashboard with a persistent sidebar for easy navigation. Visual cues, such as the "Order Successful" animation and clear "Add to Cart" buttons, guide the user effortlessly through the purchasing process.
- **Screenshot of Code :**



```
checkout_process.php X
D: > XAMPP > htdocs > Digital_Tech_Hub > checkout_process.php
211 <div class="card">
212     <div class="checkmark">
213         <svg fill="none" viewBox="0 0 24 24" stroke="currentColor">
214             <path stroke-width="3" stroke-linecap="round" stroke-linejoin="round"
215                 d="M5 13l4 4L19 7" />
216         </svg>
217     </div>
218     <h2>Order Placed Successfully</h2>
219     <div class="subtitle">Thanks for shopping with Digital Tech Hub</div>
220
221
```

- **Screenshot of UI :**



2.5 Usability Specifications (NFR-U2: Responsiveness)

- Proposed Requirement (NFR-U2):** "The system shall be responsive, ensuring the interface is accessible and functional across various devices (desktop, Laptop, tablet)."
- Completed Implementation:** We utilized CSS3 Media Queries to ensure the layout adapts dynamically to different screen sizes. The dashboard sidebar and grids automatically stack or resize when the viewport width decreases (e.g., on mobile devices).

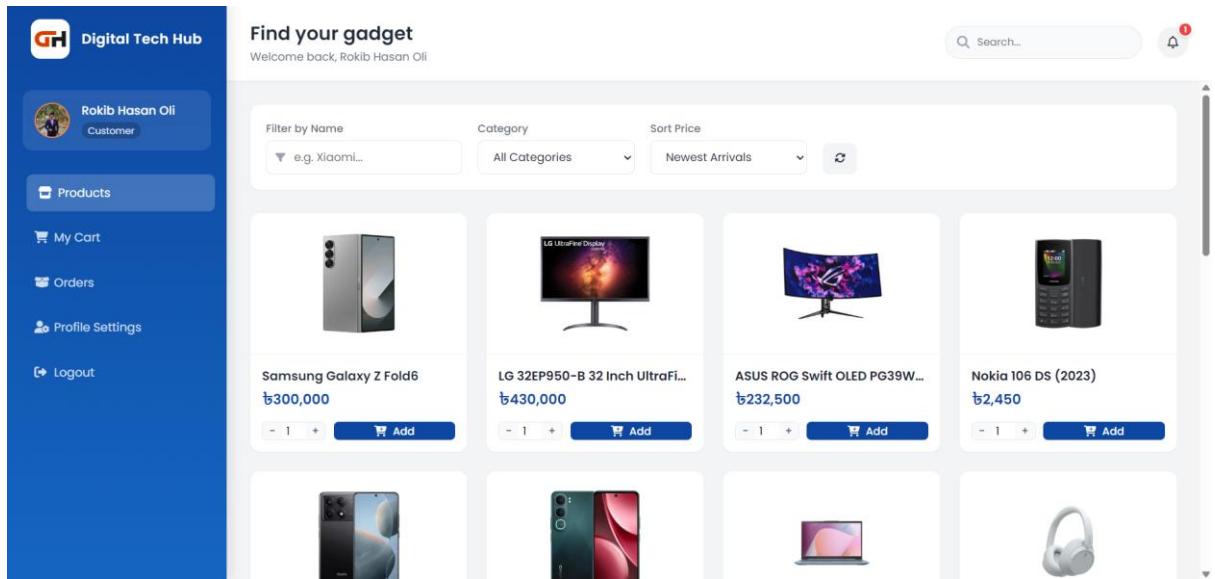
- Screenshot of Code :

```
user_dashboard.css 1 ×

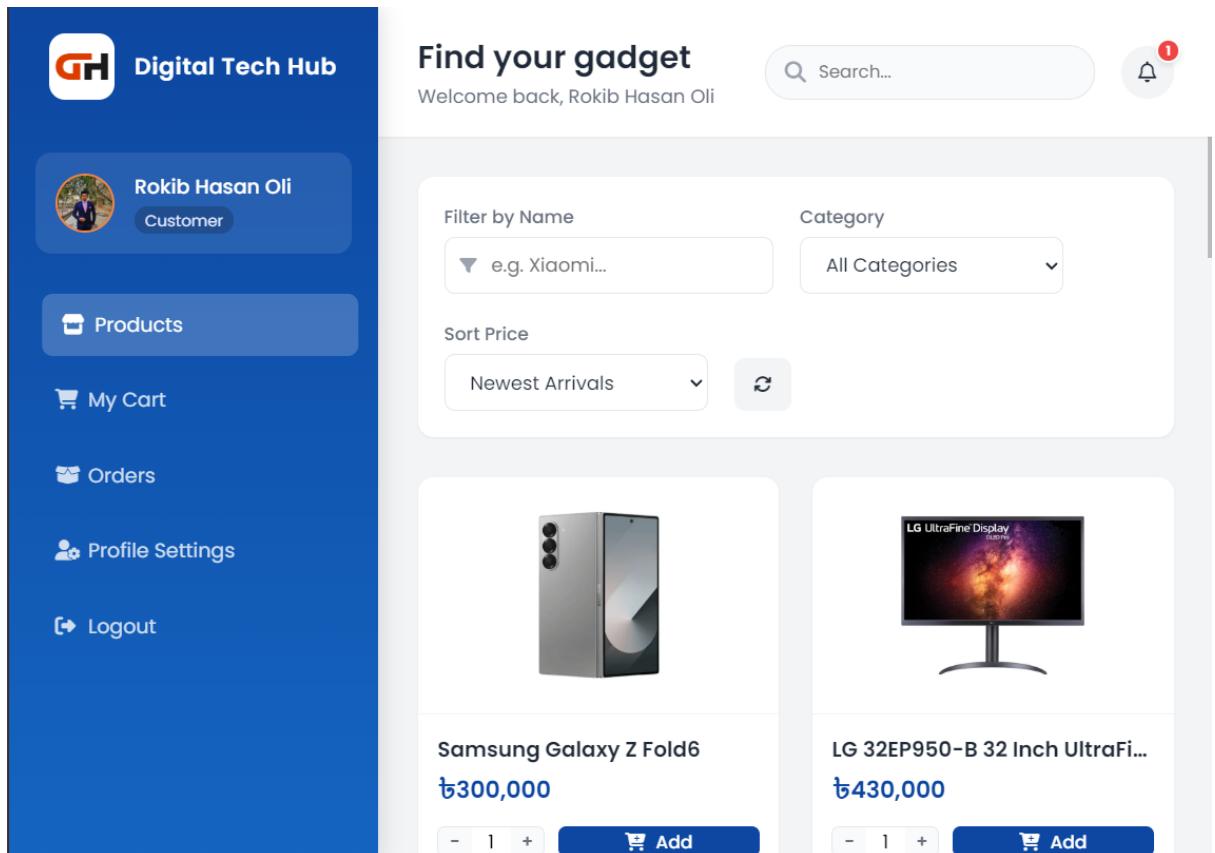
573  /* ===== */
574  |   RESPONSIVE
575  ===== */
576 @media (max-width: 900px) {
577   .sidebar {
578     width: 80px;
579     padding: 15px 10px;
580   }
581
582   .sidebar .logo-area span,
583   .user-profile-widget,
584   .side-nav a span {
585     display: none;
586   }
587
588   .side-nav a {
589     justify-content: center;
590     padding: 15px;
591   }
592
593   .side-nav a i {
594     width: auto;
595     font-size: 1.2rem;
596   }
597
598   .logo-area {
599     padding-left: 0;
600     justify-content: center;
601   }
602 }
```

- Screenshot of UI :

Desktop, Laptop:



Tablet:



2.6 Reliability Specifications (NFR-R1: Data Integrity)

- Proposed Requirement (NFR-R1):** "The database (MySQL) must ensure data integrity for all critical transactions (e.g., order placement, inventory updates)."

- **Completed Implementation:** We enforced Referential Integrity using **Foreign Keys** in our MySQL database. This ensures that an order cannot exist without a valid user, and order items must be linked to valid products. Additionally, the ON DELETE CASCADE constraint cleans up related data automatically to prevent orphaned records.

- **Screenshot of Code :**

```

D: > XAMPP > htdocs > Digital_Tech_Hub > Database > Digital_Tech_Hub.sql

45 CREATE TABLE IF NOT EXISTS orders (
46     order_id INT AUTO_INCREMENT PRIMARY KEY,
47     user_id INT,
48     total_amount DECIMAL(10, 2) NOT NULL,
49     order_status ENUM('Pending', 'Processing', 'Shipped', 'Delivered', 'Cancelled') DEFAULT 'Pending',
50     payment_method ENUM('COD', 'Bkash', 'Nagad', 'Online') NOT NULL,
51     payment_status ENUM('Pending', 'Paid', 'Failed') DEFAULT 'Pending',
52     order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
53     FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
54 );
55
56 CREATE TABLE IF NOT EXISTS order_items (
57     order_item_id INT AUTO_INCREMENT PRIMARY KEY,
58     order_id INT,
59     product_id INT,
60     quantity INT NOT NULL,
61     unit_price DECIMAL(10, 2) NOT NULL,
62     FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
63     FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
64 );

```

- **Screenshot of UI :**

The screenshot shows the phpMyAdmin interface for the 'orders' table in the 'digital_tech_hub' database. The table structure is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action	
1	order_id	int(11)			No	None	AUTO_INCREMENT			
2	user_id	int(11)			Yes	NULL				
3	total_amount	decimal(10,2)			No	None				
4	order_status	enum('Pending','Processing','Shipped','Delivered','Cancelled')	utf8mb4_general_ci		Yes	Pending				
5	payment_method	enum('COD','Bkash','Nagad','Online')	utf8mb4_general_ci		No	None				
6	payment_status	enum('Pending','Paid','Failed')	utf8mb4_general_ci		Yes	Pending				
7	order_date	timestamp			No	current_timestamp()				

Below the table structure, there are sections for 'Indexes' and 'Partitions'. The 'Indexes' section shows two primary keys: 'PRIMARY' (on column 10) and 'user_id' (on column 5). The 'Partitions' section is currently empty.

2.7 Maintainability Specifications (NFR-M1: Documentation & Standards)

- **Proposed Requirement (NFR-M1):** "The code base (PHP, HTML, CSS) shall be well-documented and follow standard coding practices to facilitate future updates and bug fixes."

- **Completed Implementation:** We adopted standard **Design Patterns** (Singleton, Factory, Facade) which inherently organize code into modular, maintainable structures. We also included comments explaining the purpose of complex logic, such as the OrderFacade and Database connection classes.

- **Screenshot of Code :**

The screenshot shows a code editor interface with two tabs open. The top tab is 'OrderFacade.php' and the bottom tab is 'Database.php'. Both tabs show PHP code with syntax highlighting. The 'Database.php' file is currently active and has a red rectangular selection box around its entire content.

```

OrderFacade.php
D: > XAMPP > htdocs > Digital_Tech_Hub > OrderFacade.php
1  <?php
2  require_once 'Database.php';
3  require_once 'PaymentFactory.php';
4  require_once 'OrderObserver.php';
5
6  class OrderFacade {
7      public function placeOrder($userId, $cartTotal, $paymentMethod, $cartItems) {
8          $db = Database::getInstance()->getConnection();
9
10         // 1. Handle Payment (Factory)
11         $paymentObj = PaymentFactory::create($paymentMethod);
12         $paymentStatus = $paymentObj->pay($cartTotal); // In real app, check return value
13
14         // 2. Create Order (Database)
15         $stmt = $db->prepare("INSERT INTO orders (user_id, total_amount, payment_method, order_status) VALUES (?, ?, ?, 'Pending')");
16         $stmt->bind_param("ids", $userId, $cartTotal, $paymentMethod);
17         $stmt->execute();
18         $orderId = $db->insert_id;
19
20         // 3. Save Items
21         foreach ($cartItems as $item) {
22             $stmtItem = $db->prepare("INSERT INTO order_items (order_id, product_id, quantity, unit_price) VALUES (?, ?, ?, ?)");
23             $stmtItem->bind_param("iiid", $orderId, $item['id'], $item['qty'], $item['price']);
24             $stmtItem->execute();
25         }
26
27         // 4. Notify (Observer)
28         $subject = new OrdersSubject();
29         $subject->attach(new EmailNotifier());
30         $subject->attach(new AdminLogObserver());
31         $subject->notify($orderId, 'Pending');
32
33         return $orderId;
34     }
35 }
36 ?>

```

```

Database.php
1  <?php
2  class Database {
3
4      private static $instance = null;
5      private $connection;
6
7      private $host = 'localhost';
8      private $username = 'root';
9      private $password = '';
10     private $database = 'digital_tech_hub';
11
12     private function __construct() {
13
14         $this->connection = new mysqli($this->host, $this->username, $this->password, $this->database);
15         if ($this->connection->connect_error) {
16             die("Database Connection Failed: " . $this->connection->connect_error);
17         }
18         $this->connection->set_charset("utf8");
19     }
20
21     public static function getInstance() {
22         if (self::$instance === null) {
23             self::$instance = new Database();
24         }
25         return self::$instance;
26     }
27
28     public function getConnection() {
29         return $this->connection;
30     }
31
32     private function __clone() {}
33
34     public function __wakeup() {}
35 }
36 ?>

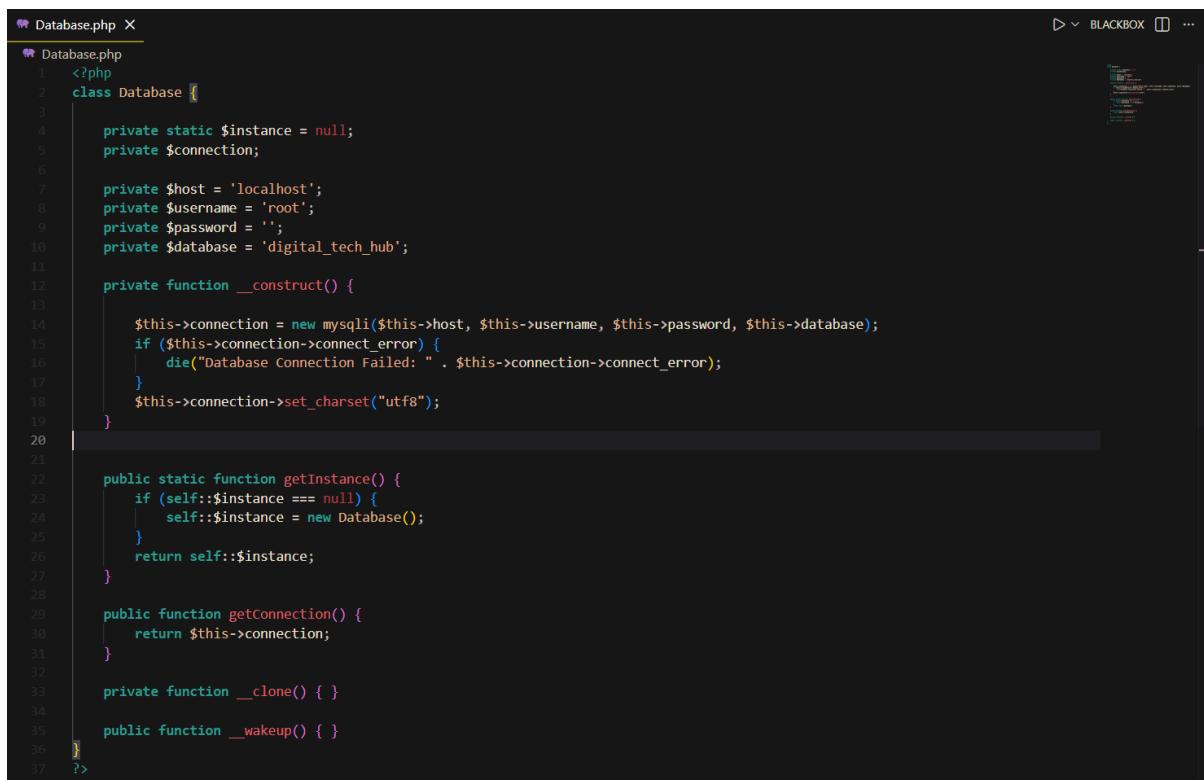
```

3. Implementation of Design Patterns

To ensure the system is maintainable, scalable, and secure, we implemented **7 standard Software Engineering Design Patterns**. Below is the explanation of how each pattern addresses a specific problem in the **Digital Tech Hub** system.

3.1 Singleton Pattern (Database Connection)

- **Problem:** Opening a new database connection for every single query consumes server memory and slows down the website (NFR-P1).
- **Implementation:** We created a Database class with a private constructor and a static getInstance() method. This ensures that only **one** connection object exists per user session, no matter how many times the database is accessed.
- **Screenshot of Code:**



The screenshot shows a code editor window with a dark theme. The file is named 'Database.php'. The code implements the Singleton design pattern for a database connection. It includes a private constructor, a static getInstance() method to return the single instance, and a getConnection() method to return the database connection object. The code also handles MySQLi errors and sets the character set to utf8.

```
<?php
class Database {
    private static $instance = null;
    private $connection;

    private $host = 'localhost';
    private $username = 'root';
    private $password = '';
    private $database = 'digital_tech_hub';

    private function __construct() {
        $this->connection = new mysqli($this->host, $this->username, $this->password, $this->database);
        if ($this->connection->connect_error) {
            die("Database Connection Failed: " . $this->connection->connect_error);
        }
        $this->connection->set_charset("utf8");
    }

    public static function getInstance() {
        if (self::$instance === null) {
            self::$instance = new Database();
        }
        return self::$instance;
    }

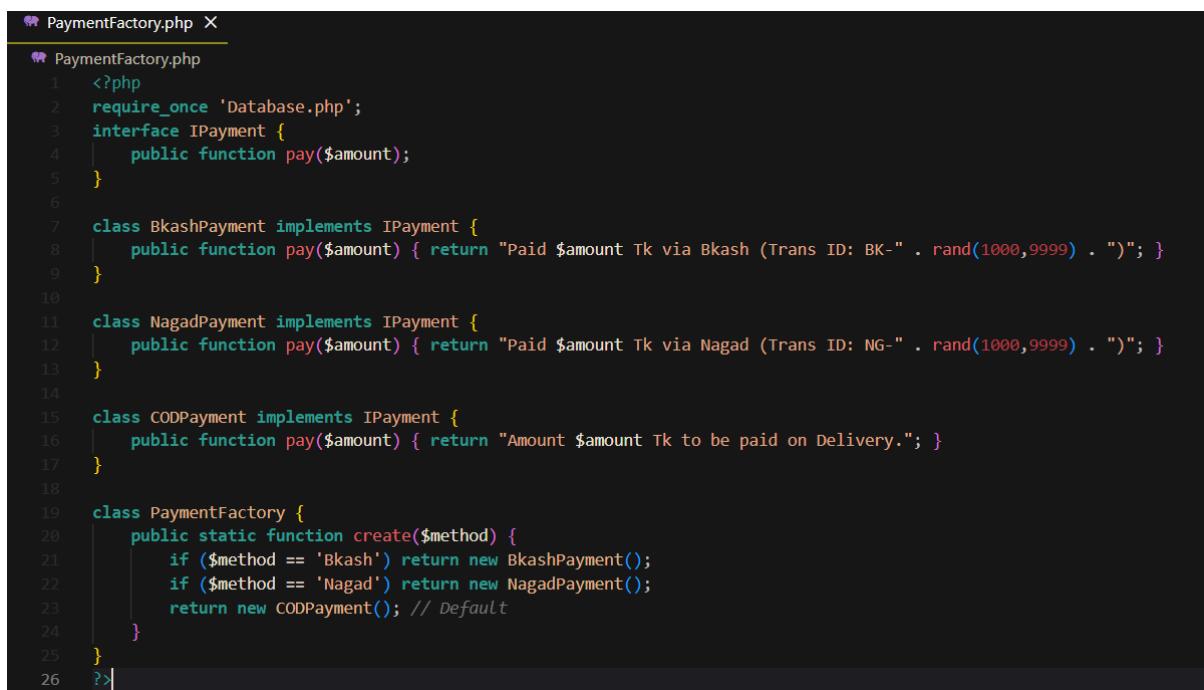
    public function getConnection() {
        return $this->connection;
    }

    private function __clone() { }

    public function __wakeup() { }
}
```

3.2 Factory Pattern (Payment Processing)

- **Problem:** The checkout system needs to handle multiple payment methods (Bkash, Nagad, COD). Hardcoding if-else logic for every method makes the code messy and hard to extend.
- **Implementation:** We used the **Factory Pattern** to create a PaymentFactory. The checkout logic simply asks the factory for a payment object based on the user's selection, without worrying about the specific class logic.
- **Screenshot of Code:**



```
PaymentFactory.php
1 <?php
2 require_once 'Database.php';
3 interface IPayment {
4     public function pay($amount);
5 }
6
7 class BkashPayment implements IPayment {
8     public function pay($amount) { return "Paid $amount Tk via Bkash (Trans ID: BK-" . rand(1000,9999) . ")"; }
9 }
10
11 class NagadPayment implements IPayment {
12     public function pay($amount) { return "Paid $amount Tk via Nagad (Trans ID: NG-" . rand(1000,9999) . ")"; }
13 }
14
15 class CODPayment implements IPayment {
16     public function pay($amount) { return "Amount $amount Tk to be paid on Delivery."; }
17 }
18
19 class PaymentFactory {
20     public static function create($method) {
21         if ($method == 'Bkash') return new BkashPayment();
22         if ($method == 'Nagad') return new NagadPayment();
23         return new CODPayment(); // Default
24     }
25 }
26 ?>
```

3.3 Facade Pattern (Order Processing)

- **Problem:** Placing an order involves a complex sequence: processing payment, saving to the database, and sending notifications. If the checkout page handles all this, it violates the "Single Responsibility Principle."
- **Implementation:** We introduced OrderFacade as a simplified interface. The client (checkout page) calls one method placeOrder(), and the Facade handles the complex interactions between the Payment, Database, and Notification subsystems.

- Screenshot of Code:

```

<?php
require_once 'database.php';
require_once 'PaymentFactory.php';
require_once 'OrderObserver.php';

class OrderFacade {
    public function placeOrder($userId, $cartTotal, $paymentMethod, $cartItems) {
        $db = Database::getInstance()->getConnection();

        $paymentObj = PaymentFactory::create($paymentMethod);
        $paymentObj->pay($cartTotal);

        $stmt = $db->prepare("INSERT INTO orders (user_id, total_amount, payment_method, order_status) VALUES (?, ?, ?, 'Pending')");
        $stmt->bind_param("ids", $userId, $cartTotal, $paymentMethod);
        $stmt->execute();
        $orderId = $db->insert_id;

        foreach ($cartItems as $item) {
            $stmtItem = $db->prepare("INSERT INTO order_items (order_id, product_id, quantity, unit_price) VALUES (?, ?, ?, ?)");
            $stmtItem->bind_param("iiid", $orderId, $item['id'], $item['qty'], $item['price']);
            $stmtItem->execute();
        }

        $subject = new OrdersSubject();

        $subject->attach(new DatabaseNotifier($userId));
        $subject->attach(new EmailNotifier());
        $subject->attach(new AdminLogObserver());

        $subject->notify($orderId, 'Pending');

        return $orderId;
    }
}
?>

```

3.4 Observer Pattern (Notifications)

- **Problem:** When an order is placed, we need to perform multiple independent actions (Email User, Log for Admin, Alert Dashboard). Hardcoding these into the order logic makes it tightly coupled.
- **Implementation:** We used the **Observer Pattern**. The Order system acts as the "Subject" and broadcasts a notification. Independent "Observers" (EmailNotifier, AdminLogObserver) listen for this event and react automatically.

- Screenshot of Code:

```
OrderObserver.php X
OrderObserver.php
1 <?php
2 require_once 'Database.php';
3
4 // 1. The Observer Interface
5 interface IObserver {
6     public function update($orderId, $status);
7 }
8
9 // 2. Concrete Observer: Database Notification (The Real Feature)
10 class DatabaseNotifier implements IObserver {
11     private $userId;
12
13     // We need the User ID to know WHO to notify
14     public function __construct($userId) {
15         $this->userId = $userId;
16     }
17
18     public function update($orderId, $status) {
19         $db = Database::getInstance()->getConnection();
20
21         $message = "Your Order #$orderId is now confirmed! Status: $status";
22
23         // Save to Database
24         $stmt = $db->prepare("INSERT INTO notifications (user_id, message) VALUES (?, ?)");
25         if ($stmt) {
26             $stmt->bind_param("is", $this->userId, $message);
27             $stmt->execute();
28         }
29     }
30 }
```

```
OrderObserver.php X
OrderObserver.php
32 // 3. Concrete Observer: Email (simulation)
33 class EmailNotifier implements IObserver {
34     public function update($orderId, $status) {
35         $_SESSION['msg_email'] = "Email sent to user for Order #$orderId";
36     }
37 }
38
39 // 4. Concrete Observer: Admin Log
40 class AdminLogObserver implements IObserver {
41     public function update($orderId, $status) {
42         // Keeps a Log for admins
43         error_log("Order #$orderId placed at " . date('Y-m-d H:i:s'));
44     }
45 }
46
47 // 5. The Subject (Observable)
48 class OrderSubject {
49     private $observers = [];
50
51     public function attach(IObserver $observer) {
52         $this->observers[] = $observer;
53     }
54
55     public function notify($orderId, $status) {
56         foreach ($this->observers as $observer) {
57             $observer->update($orderId, $status);
58         }
59     }
60 }
61 ?>
```

3.5 Strategy Pattern (Discounts)

- **Problem:** We need to apply different discount rules (e.g., Holiday Sale, No Discount) dynamically without rewriting the cart logic.
- **Implementation:** We defined a DiscountContext that can switch between different strategies (algorithms) at runtime. For example, if the coupon "HOLIDAY10" is used, the strategy switches to HolidayDiscount.
- **Screenshot of Code:**

```
DiscountStrategy.php X
1  <?php
2  interface IDiscountStrategy {
3      public function calculate($total);
4  }
5
6  class NoDiscount implements IDiscountStrategy {
7      public function calculate($total) { return $total; }
8  }
9
10 class HolidayDiscount implements IDiscountStrategy {
11     public function calculate($total) { return $total * 0.90; }
12 }
13
14 class DiscountContext {
15     private $strategy;
16
17     public function __construct() {
18         $this->strategy = new NoDiscount();
19     }
20
21     public function setStrategy(IDiscountStrategy $strategy) {
22         $this->strategy = $strategy;
23     }
24
25     public function getFinalPrice($rawTotal) {
26         return $this->strategy->calculate($rawTotal);
27     }
28 }
29 ?>
```

```

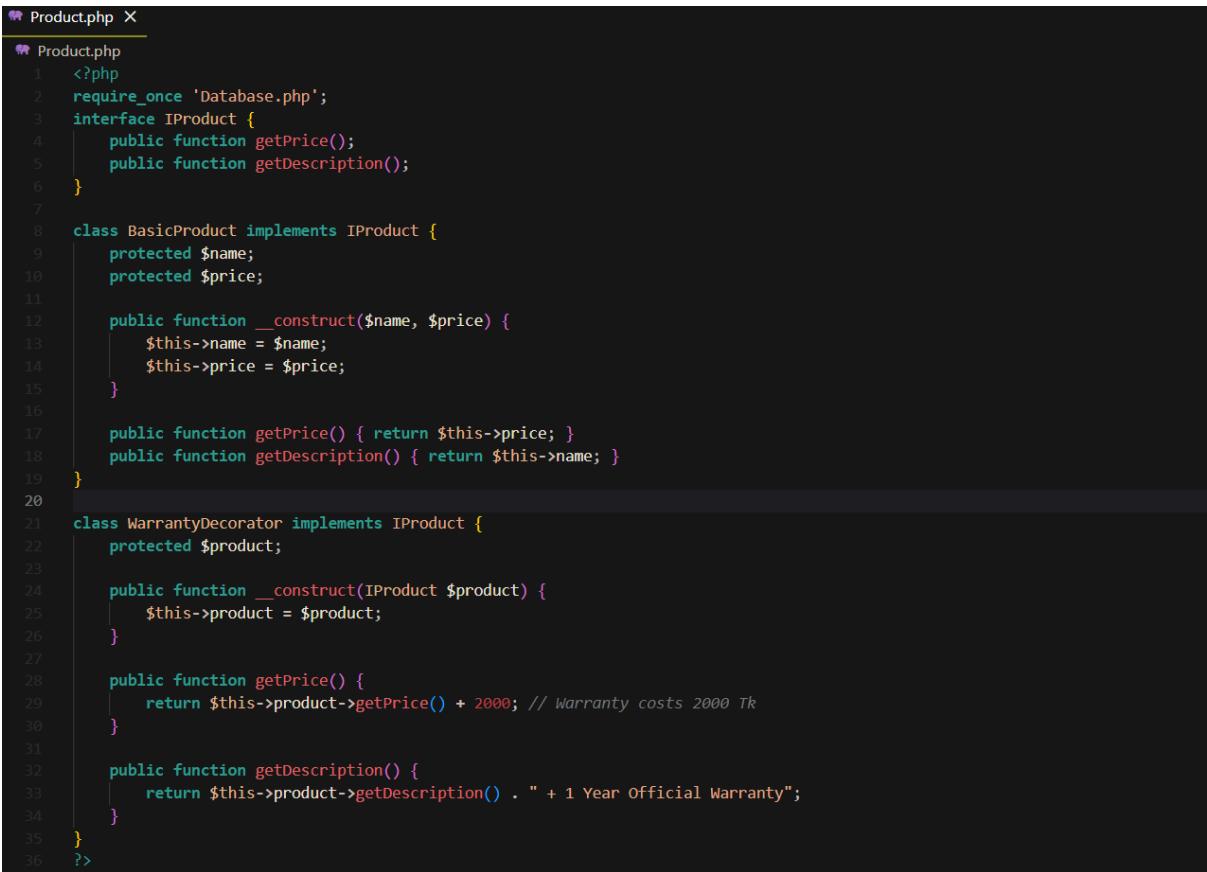
cart_view.php X
cart_view.php
1 <?php
2 session_start();
3 require_once 'DiscountStrategy.php';
4
5 $rawTotal = 0;
6 if (isset($_SESSION['cart'])) {
7     foreach ($_SESSION['cart'] as $item) {
8         $rawTotal += ($item['price'] * $item['qty']);
9     }
10 }
11
12 // Strategy Pattern Usage
13 $context = new DiscountContext();
14 $msg = "";
15
16 if (isset($_POST['apply_coupon'])) {
17     if ($_POST['code'] == 'HOLIDAY10') {
18         $context->setStrategy(new HolidayDiscount());
19         $msg = "Coupon Applied: 10% off";
20     }
21 }

```

3.6 Decorator Pattern (Product Warranty)

- **Problem:** Users can add an optional "1 Year Warranty" to a product. Creating a separate class for every product-warranty combination (e.g., iPhoneWithWarranty) would lead to class explosion.
- **Implementation:** We used the **Decorator Pattern**. We wrap the BasicProduct object inside a WarrantyDecorator. This dynamically adds the warranty cost (+2000 Tk) and updates the description without altering the original product class.

- Screenshot of Code:



```
Product.php X
Product.php
1  <?php
2  require_once 'Database.php';
3  interface IProduct {
4      public function getPrice();
5      public function getDescription();
6  }
7
8  class BasicProduct implements IProduct {
9      protected $name;
10     protected $price;
11
12     public function __construct($name, $price) {
13         $this->name = $name;
14         $this->price = $price;
15     }
16
17     public function getPrice() { return $this->price; }
18     public function getDescription() { return $this->name; }
19 }
20
21 class WarrantyDecorator implements IProduct {
22     protected $product;
23
24     public function __construct(IProduct $product) {
25         $this->product = $product;
26     }
27
28     public function getPrice() {
29         return $this->product->getPrice() + 2000; // Warranty costs 2000 Tk
30     }
31
32     public function getDescription() {
33         return $this->product->getDescription() . " + 1 Year Official Warranty";
34     }
35 }
36 ?>
```

3.7 Proxy Pattern (Admin Security)

- **Problem:** The Admin Dashboard handles sensitive operations. We need to ensure that no unauthorized user can even attempt to load the dashboard logic.
- **Implementation:** We used the **Proxy Pattern**. The AdminProxy sits in front of the RealAdminDashboard. It intercepts the request, checks if the user has the "admin" role, and only then loads the real dashboard.

- Screenshot of Code:

The screenshot shows a code editor window with the file 'AdminProxy.php' open. The code implements the Proxy pattern in PHP. It defines a 'RealAdminDashboard' class that implements the 'IAdminDashboard' interface. This class has a single method 'render()' which returns true, indicating access is granted. The 'AdminProxy' class also implements 'IAdminDashboard'. It contains logic to check if a session exists and if the user's role is 'admin'. If not, it outputs an access denied message and exits. It then initializes a 'realAdmin' object if it is null, creating a new instance of 'RealAdminDashboard'. Finally, it returns the result of calling 'render()' on the 'realAdmin' object.

```
1 // Real Subject
2 class RealAdminDashboard implements IAdminDashboard {
3     public function render() {
4         return true; // Access Granted signal
5     }
6 }
7
8 // Proxy
9 class AdminProxy implements IAdminDashboard {
10    private $realAdmin;
11
12    public function render() {
13        if (session_status() == PHP_SESSION_NONE) session_start();
14
15        // Check Authentication & Role
16        if (!isset($_SESSION['role']) || $_SESSION['role'] != 'admin') {
17            echo "<h2>Access Denied: You do not have Admin privileges.</h2>";
18            echo "<a href='login_register.php'>Login</a>";
19            exit();
20        }
21
22        // Lazy Initialization
23        if ($this->realAdmin == null) {
24            $this->realAdmin = new RealAdminDashboard();
25        }
26
27        return $this->realAdmin->render();
28    }
29 }
30
31
32
33
34
35
36
37
38 ?>
```

4. Diagram vs. Implementation Comparison

This section analyses the differences between the initial system design (Class and Sequence Diagrams) and the final implemented code. While the high-level architecture remains consistent, several refinements were made during development to incorporate **Design Patterns** for better scalability and maintainability.

4.1 Class Diagram Analysis

Original Design: The initial Class Diagram depicted a simplified object structure:

- A single **Product** class containing all attributes (Price, Description).
- A generic association between **Order** and **Payment**.
- A direct **Admin** class for managing the dashboard.

Implementation Changes (Refinements):

1. **Product Structure (Decorator Pattern):** In the implementation, we moved away from a rigid **Product** class. Instead, we implemented the **Decorator Pattern** using an **IProduct** interface and a **WarrantyDecorator** class. This allows us to dynamically add features (like "1 Year Warranty") and modify the price at runtime, which was not represented in the static class diagram.
2. **Payment Logic (Factory Pattern):** The diagram showed a simple **Payment** entity. In the code, we refactored this into a **Factory Pattern** structure. We created a **PaymentFactory** that generates specific payment objects (**BkashPayment**, **NagadPayment**) based on user selection.
3. **Admin Security (Proxy Pattern):** To enhance security (NFR-S1), we interposed an **AdminProxy** class between the client and the **RealAdminDashboard**. The original diagram showed direct access, whereas the implementation enforces a security check via this Proxy.

Screenshot of Code:

```

Product.php X
Product.php
1  <?php
2  require_once 'Database.php';
3  interface IProduct {
4      public function getPrice();
5      public function getDescription();
6  }
7
8  class BasicProduct implements IProduct {
9      protected $name;
10     protected $price;
11
12     public function __construct($name, $price) {
13         $this->name = $name;
14         $this->price = $price;
15     }
16
17     public function getPrice() { return $this->price; }
18     public function getDescription() { return $this->name; }
19 }
20
21 class WarrantyDecorator implements IProduct {
22     protected $product;
23
24     public function __construct(IProduct $product) {
25         $this->product = $product;
26     }
27
28     public function getPrice() {
29         return $this->product->getPrice() + 2000; // Warranty costs 2000 Tk
30     }
31
32     public function getDescription() {
33         return $this->product->getDescription() . " + 1 Year Official Warranty";
34     }
35 }
36 ?>

```

4.2 Sequence Diagram Analysis

Original Design: The initial Sequence Diagram for the "Buy Product" use case illustrated a synchronous flow:

1. User clicks "Buy".
2. Server checks price.
3. Server shows "Total Bill".

Implementation Changes (Refinements):

1. **Orchestration (Facade Pattern):** The actual checkout process is much more sophisticated than the simple "Check Price" step shown in the diagram. We implemented an **OrderFacade** that orchestrates multiple subsystems in a specific order:

Payment Processing → Database Insertion → Session Cleanup.

2. **Asynchronous-like Events (Observer Pattern):** The most significant addition is the **Notification System**. The original sequence diagram ended at "Show Total Bill". In the implementation, immediately after the order is saved, the **OrderSubject** fires a notification event. This triggers the **EmailNotifier** and **AdminLogObserver** to run background tasks

(logging and emailing) before the final success message is rendered to the user.

Screenshot of Code:

```
<?php
require_once 'Database.php';
require_once 'PaymentFactory.php';
require_once 'OrderObserver.php';

class OrderFacade {
    public function placeOrder($userId, $cartTotal, $paymentMethod, $cartItems) {
        $db = Database::getInstance()->getConnection();

        $paymentObj = PaymentFactory::create($paymentMethod);
        $paymentObj->pay($cartTotal);

        $stmt = $db->prepare("INSERT INTO orders (user_id, total_amount, payment_method, order_status) VALUES (?, ?, ?, 'Pending')");
        $stmt->bind_param("ids", $userId, $cartTotal, $paymentMethod);
        $stmt->execute();
        $orderId = $db->insert_id;

        foreach ($cartItems as $item) {
            $stmtItem = $db->prepare("INSERT INTO order_items (order_id, product_id, quantity, unit_price) VALUES (?, ?, ?, ?)");
            $stmtItem->bind_param("iiid", $orderId, $item['id'], $item['qty'], $item['price']);
            $stmtItem->execute();
        }

        $subject = new OrdersSubject();

        $subject->attach(new DatabaseNotifier($userId));
        $subject->attach(new EmailNotifier());
        $subject->attach(new AdminLogObserver());

        $subject->notify($orderId, 'Pending');

        return $orderId;
    }
}
?>
```