KATHFORD
INTERNATIONAL COLLEGE OF ENGINEERING & MANAGEMENT | AFFILIATED TO TRIBHUVAN UNIVERSITY

Lab Report 4

Form handling validation and Server side Product Entry

Course: CSC381 - E-Commerce

Submitted by:

Parakram Kharel
Roll No: 24

Kathford International College of Engineering and Management
Affiliated to Tribhuvan University

July 27, 2025

1. Objective

Implement client-side validation with JavaScript and secure server-side form processing with PHP and MySQL, enabling reliable product and customer data entry.

2. Tools and Technologies Used

Technology	Purpose
HTML5	Semantic markup and form structure
CSS3	Styling and responsive layouts
JavaScript (jQuery)	Client-side validation and interaction
PHP	Server-side logic and database operations
MySQL	Data storage and management
Bootstrap 5	UI design and responsiveness
VS Code	Development environment
Browser DevTools	Debugging and testing

3. Theory / Background

Client-side JavaScript validation improves user experience by catching errors early. Server-side PHP with prepared statements secures data integrity and prevents SQL injection. Together with MySQL for data storage and the Post/Redirect/Get pattern for UX, this creates robust, scalable e-commerce entry forms.

4. Page Layout Design

4.1 Product Entry Form Structure

The product entry form is designed with a clean, responsive Bootstrap 5 layout featuring:

- Input fields for product title, description, keywords, and price with placeholder text and proper HTML5 validation
- Select dropdowns for product categories and brands, dynamically populated from the database
- Multiple image upload inputs with client-side file type and size validation
- A prominent submit button that triggers client-side validation before submitting to the server

- Consistent styling and spacing for a user-friendly experience across devices

4.2 Customer Entry Form Structure

The customer entry form follows similar design principles for consistency:

- Required fields including first name, last name, and email with validation and placeholder hints
- Optional inputs for phone number, address, city, state, postal code, and country
- Responsive grid layout for address fields for better usability on different screen sizes
- Submit button enhanced with gradient styling and hover animations for visual appeal
- Integrated client-side validation using jQuery to ensure form completeness and format correctness

4.3 Validation and Security Features

- JavaScript (jQuery) client-side validation prevents incomplete or incorrectly formatted submissions, reducing server load and improving user experience
- Server-side PHP validation, sanitization, and prepared statements with `bind_param()` ensure data integrity and prevent SQL injection attacks
- Implementation of the Post/Redirect/Get (PRG) pattern to prevent duplicate submissions on page refresh
- Secure handling of image uploads, including MIME type checks and unique file naming to avoid conflicts or security issues
- User feedback through alert popups on errors and confirmations after successful data submission

5. Code Snippets

5.1 HTML Code: Product Entry Form

```
<form id="productForm" method="POST" enctype="multipart/form-data"
novalidate>
    <div>
        <label for="product_title">Product Title *</label>
        <input type="text" id="product_title" name="product_title"
        required>
    </div>
    <div>
        <label for="description">Description *</label>
        <textarea id="description" name="description" required></textarea>
    </div>
    <div>
        <label for="product_keywords">Keywords *</label>
        <input type="text" id="product_keywords" name="product_keywords"
        required>
    </div>
    <div>
        <label for="product_category">Category *</label>
        <select id="product_category" name="product_category" required>
            <!-- Options dynamically populated -->
        </select>
    </div>
    <div>
        <label for="product_brand">Brand *</label>
        <select id="product_brand" name="product_brand" required>
            <!-- Options dynamically populated -->
        </select>
    </div>
    <div>
        <label for="product_price">Price *</label>
        <input type="number" id="product_price" name="product_price"
        step="0.01" required>
    </div>
    <div>
        <label for="product_image1">Image 1 *</label>
        <input type="file" id="product_image1" name="product_image1"
        accept="image/*" required>
    </div>
    <div>
        <label for="product_image2">Image 2 *</label>
        <input type="file" id="product_image2" name="product_image2"
        accept="image/*" required>
    </div>
```

```
    <div>
        <label for="product_image3">Image 3 *</label>
        <input type="file" id="product_image3" name="product_image3"
        accept="image/*" required>
    </div>
    <button type="submit" name="insert_product">Add Product</button>
</form>
```

5.2 PHP Code: Server-Side Product Insert

```
<?php
if (isset($_POST['insert_product'])) {
    $stmt = $conn->prepare("INSERT INTO products (product_title,
    product_description, product_keywords,
        category_id, brand_id, product_image1, product_image2,
        product_image3, product_price, date, status)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, NOW(), ?)");

    $stmt->bind_param("sssissssds",
        $product_Title, $product_Desc, $product_Key, $product_Cat,
        $product_Brand,
        $product_Image1, $product_Image2, $product_Image3, $product_Price,
        $product_Status
    );

    if ($stmt->execute()) {
      echo "<script>alert('Product inserted successfully');</script>";
  } else {
      error_log('Execute failed: ' . $stmt->error);
      echo "<script>alert('DB error. Please try again later.');</script>";
  }
}
?>
```

5.3 HTML Code: Customer Entry Form

```
<form id="customerForm" method="POST" novalidate>
    <div>
        <label for="first_name">First Name *</label>
        <input type="text" id="first_name" name="first_name" required>
    </div>
    <div>
```

```
        <label for="last_name">Last Name *</label>
        <input type="text" id="last_name" name="last_name" required>
    </div>
    <div>
        <label for="email">Email *</label>
        <input type="email" id="email" name="email" required>
    </div>
    <div>
        <label for="phone">Phone</label>
        <input type="tel" id="phone" name="phone">
    </div>
    <div>
        <label for="address">Address</label>
        <textarea id="address" name="address"></textarea>
    </div>
    <button type="submit" name="add_customer">Add Customer</button>
</form>
```

5.4 PHP Code: Server-Side Customer Insert

```
<?php
if (isset($_POST['add_customer'])) {
    $stmt = $conn->prepare("INSERT INTO customers
        (first_name, last_name, email, phone, address, city, state,
        postal_code, country, created_at)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, NOW())");

    $stmt->bind_param("sssssssss",
        $first_name, $last_name, $email,  $phone, $address,
        $city, $state, $postal_code, $country
    );

    if ($stmt->execute()) {
      $stmt->close();
      header("Location: add_customer.php?success=1");
      exit();
    }
    $stmt->close();
    error_log('Execute failed: ' . $stmt->error);
    echo "<script>alert('DB error. Please try again later.');</script>";

}
?>
```

5.5 JavaScript: jQuery Client-Side Validation

```
$(function() {
    $("#customerForm, #productForm").on("submit", function(e) {
        let errors = [];
        $(this).find("[required]").each(function() {
            if (!$.trim($(this).val())) {
                errors.push($(this).prev("label").text() + " is
                required.");
            }
        });

        let emailField = $(this).find("input[type='email']");
        if (emailField.length) {
            let email = $.trim(emailField.val());
            let emailReg = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            if (!emailReg.test(email)) {
                errors.push("Please enter a valid email.");
            }
        }
        if (errors.length > 0) {
            e.preventDefault();
            alert(errors.join("\n"));
        }
    });
});
```

6. Output / Screenshots

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	auto_increment
product_title	varchar(100)	NO		NULL	
product_description	varchar(255)	NO		NULL	
product_keywords	varchar(255)	NO		NULL	
category_id	int	NO		NULL	
brand_id	int	NO		NULL	
product_image1	varchar(255)	NO		NULL	
product_image2	varchar(255)	NO		NULL	
product_image3	varchar(255)	NO		NULL	
product_price	varchar(100)	NO		NULL	
date	timestamp	NO		NULL	
status	varchar(100)	NO		NULL	

Figure 1: Product Table.

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	auto_increment
first_name	varchar(50)	NO		NULL	
last_name	varchar(50)	NO		NULL	
email	varchar(100)	NO	UNI	NULL	
phone	varchar(15)	YES		NULL	
address	varchar(255)	YES		NULL	
city	varchar(100)	YES		NULL	
state	varchar(100)	YES		NULL	
postal_code	varchar(20)	YES		NULL	
country	varchar(100)	YES		NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Figure 2: Customer Table.

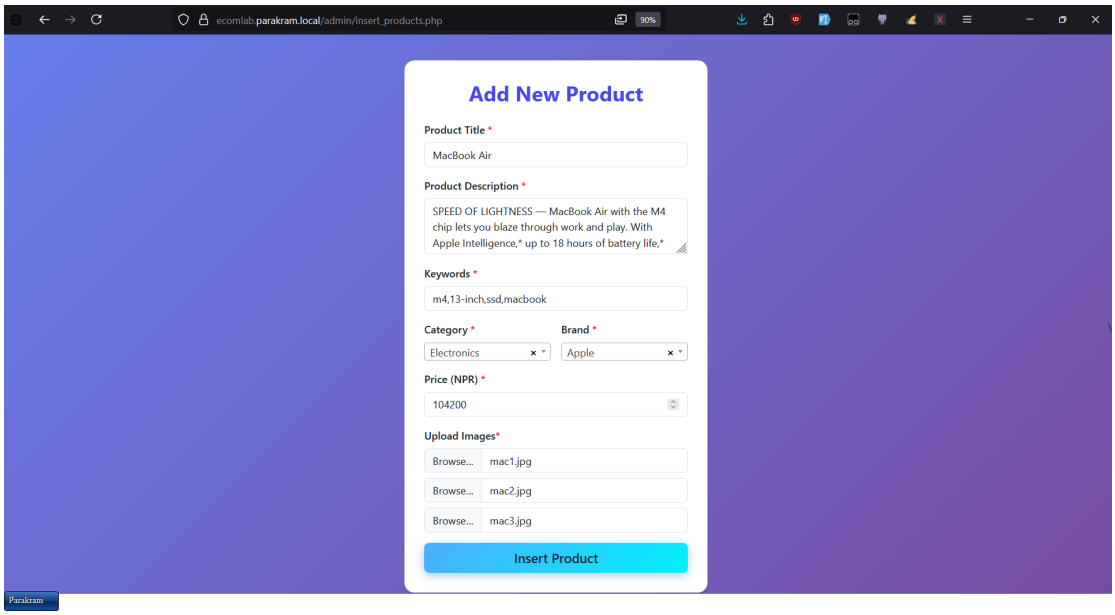
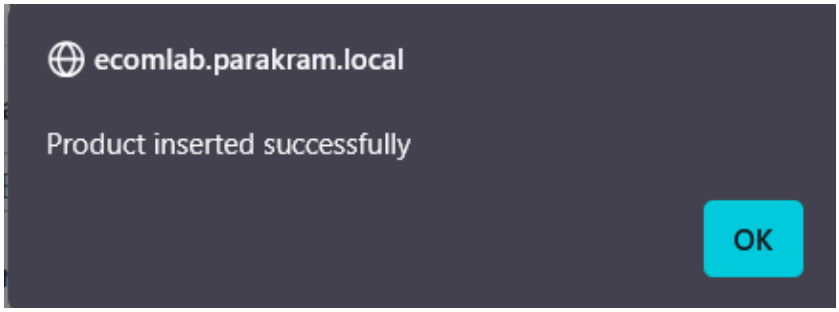


Figure 3: Insert Product Page.



product_id	product_title	product_description	product_keywords	category_id	brand_id	product_image1	product_image2	product_image3	product_price	date	status
6	MacBook Air	SPEED OF LIGHTNESS — MacBook Air with the M4 chip ...	m4,13-inch,ssd,macbook	4	2	68e4e9306ed26_mac1.jpg	68e4e9306ed26_mac2.jpg	68e4e93072cf0_mac3.jpg	104200	2025-10-07 16:04:28	true

Figure 4: Insert Product Success.

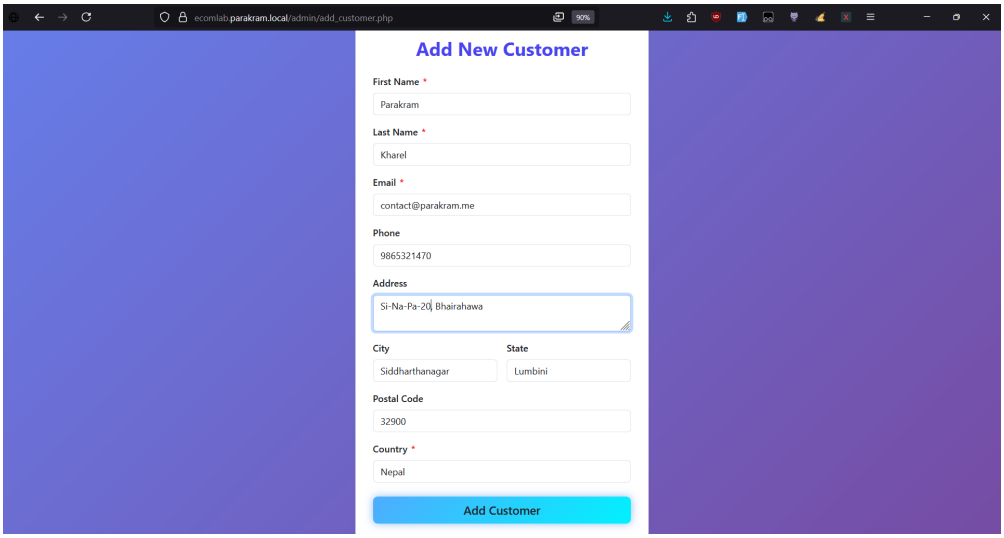
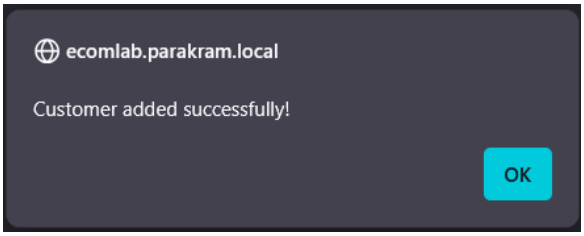


Figure 5: Insert Customer Page.



customer_id	first_name	last_name	email	phone	address	city	state	postal_code	country	created_at
1	Parakram	Kharel	contact@parakram.me	9865321470	Si-Na-Pa-20, Bhairahawa	Siddharthanagar	Lumbini	32900	Nepal	2025-10-07 16:27:52

Figure 6: Insert Customer Success.

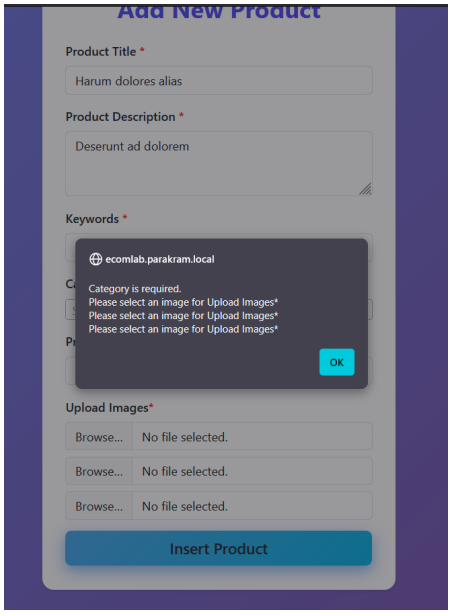


Figure 7: Client Side Validation.

7. Result

The e-commerce application integrates dynamic cart management using JavaScript arrays and objects. Users can add, update, and remove items in real-time with array methods like `push()`, `splice()`, and `reduce()`. Cart data persists via `localStorage`, while DOM updates ensure a seamless interface across devices. Server-side PHP securely processes form data using prepared statements, ensuring data integrity and preventing SQL injection. Image uploads are validated and uniquely named. The Post/Redirect/Get pattern effectively prevents duplicate submissions.

8. Conclusion

This lab covered key full-stack concepts by combining client-side validation and interactive UI with secure server-side handling and database management. Students practiced JavaScript array manipulation, DOM control, and PHP prepared statements. The project demonstrates how to build secure, user-friendly e-commerce forms enabling real-time cart operations and persistent data storage.

9. References

- MDN Array Methods Documentation
- MDN DOM Manipulation Guide
- MDN Web Storage API
- MDN JavaScript Promises Guide
- PHP mysqli Prepared Statements
- Bootstrap 5 Documentation
- jQuery Official Website