**Week 5 Finalizing the Application - Project Completion Report**

**Project Overview**

Our project is a functional e-commerce web application built using PHP, MySQL, and MVC architecture. Users can browse products, manage their cart (add/remove/update items), view order summaries with tax and shipping, and interact through a secure, maintainable interface.

**Testing Performed**

**1. Add to Cart Functionality**

* **Tested:** Adding a product to the cart from the catalog page.
* **How:** Selected quantities and submitted the form to add an item.
* **Results:** Item correctly appears in the cart view with accurate quantity, subtotal, and updated totals including tax and shipping.

**2. Remove from Cart**

* **Tested:** Removing items from the cart.
* **How:** Clicked the ‘Remove’ link on a cart item.
* **Results:** Item removed successfully, cart totals adjusted, and page refreshed without errors.

**3. Cart Quantity Updates**

* **Tested:** Adjusting quantity of items in the cart.
* **How:** Updated quantity in the input field (if enabled) and submitted changes.
* **Results:** Cart updates quantity and recalculates subtotal and total correctly.

**4. User Interface & Styling**

* **Tested:** Visual layout on different browsers and screen sizes.
* **How:** Loaded catalog and cart pages on desktop and mobile.
* **Results:** Consistent, readable styles applied from style.css. Layout and buttons appropriately sized and spaced.

**5. Database Integration**

* **Tested:** Persistent storage and retrieval of cart items.
* **How:** Added/removed items and verified changes persisted in MySQL database via phpMyAdmin queries.
* **Results:** Database reflects user cart updates accurately; no data loss or corruption found.

**6. Page Navigation**

* **Tested:** Navigation between catalog, cart, and other pages.
* **How:** Using links and form actions to move across the app.
* **Results:** All pages load correctly without broken links or errors like “Not Found.”

**7. Form Security & Input Validation**

* **Tested:** Submitting form data with invalid or malicious inputs.
* **How:** Entered negative numbers, text in quantity fields, and HTML tags.
* **Results:** Input sanitized and rejected as needed. No cross-site scripting or injection vulnerabilities detected.

**GitHub Repository**

* **URL:** [Your GitHub Repo Link Here]
* Tags: Created and pushed release tag named Phase #4 for week 5 submission.
* Included all code files, SQL export of database schema and data, and README file with project summary and setup instructions.

**Project Summary (README)**

This project demonstrates a PHP-based e-commerce shopping cart application with a MySQL backend. It uses MVC architecture to separate business logic, data access, and presentation. Users can browse products, manage their shopping carts, and view detailed order totals including tax and shipping costs. The application uses secure form handling, proper input validation, and structured code for maintainability.

**Lessons Learned**

* Planning with MVC early could save time refactoring later.
* Matching front-end form methods with controller expectations is crucial to avoid errors.
* Testing each feature thoroughly before moving on helps catch bugs faster.
* Clear separation of concerns makes code more readable and debugging easier.
* Using GitHub tags and commits regularly helps keep track of versions clearly.