LTW Project

Project description for the 2024 edition of the Web Languages and Technologies course.

Objective

Develop a website that facilitates the **buying and selling** of pre-loved items. The platform should provide a seamless experience for users to easily **list**, **browse**, and **transact**. Moreover, the website should offer robust search and filter capabilities, detailed item descriptions, and user feedback systems.

To create this website, students should:

- Implement an SQLite **database** to maintain records of users, items, transactions, messages, and product categories.
- Design responsive web pages using **HTML** and **CSS** that reflect the marketplace's interface.
- Utilize **PHP** to dynamically generate web pages by interacting with the database.
- Enhance interactivity and responsiveness on the client side using **Javascript**, potentially incorporating features like live search results and asynchronous data updates with Ajax.

Workgroups

- Students will complete this project in **groups of three**.

  In classes where the number of students is not a multiple of three, one or two groups of **two students** will be created.
- Students should contact their practical class teachers during the **weekly class** (or using **Slack**) to establish these workgroups.

Requirements

The requirements are **deliberately broad** to allow each group the **freedom** to innovate and create a unique website. Let your creativity flow!

This **second-hand marketplace** platform features three types of users: **sellers** who wish to list and manage their items, **buyers** looking to purchase items, and **admins** who manage the platform's overall operations. Users should have the flexibility to act as sellers and buyers **without needing** separate accounts.

The **minimum** expected set of requirements is as follows:

All users should be able to:

- Register a new account.
- Log in and out.
- Edit their profile, including their name, username, password, and email.

Sellers should be able to:

- List new items, providing details such as category, brand, model, size, and condition, along with images.
- Track and manage their listed items.
- Respond to inquiries from buyers regarding their items and add further information if needed.
- Print shipping forms for items that have been sold.

Buyers should be able to:

- Browse items using filters like category, price, and condition.
- Engage with sellers to ask questions or negotiate prices.
- Add items to a wishlist or shopping cart.
- Proceed to checkout with their shopping cart (simulate payment process).

Admins should be able to:

- Elevate a user to admin status.
- Introduce new item categories, sizes, conditions, and other pertinent entities.
- Oversee and ensure the smooth operation of the entire system.

Required Technologies

Ensure the application incorporates the following technologies:

- **HTML**: For structuring the web content.
- **CSS**: For styling the web pages. Aim for a mobile-first, responsive design.
- **PHP**: For server-side scripting.
- **JavaScript**: For client-side scripting.
- **Ajax/JSON**: For asynchronous web page updates.
- **PDO/SQL**: For database interactions using SQLite.

Security Measures

Prioritize security to safeguard user data and interactions:

- Protect against **SQL injection** by using prepared statements with PDO.
- Mitigate **Cross-Site Scripting (XSS)** attacks by sanitizing user input and output.

- Prevent **Cross-Site Request Forgery (CSRF)** by implementing anti-CSRF tokens.
- Implement secure password storage with proper hashing and salting techniques.

Code Quality

Maintain high standards of code quality:

- Write **clean**, and **readable** code.
- Follow a consistent naming convention and coding style.
- Organize files and directories logically.

Design Consistency

Design a user-friendly interface:

- Adopt a **clean** and **intuitive** design that is consistent across the website.
- Use generic CSS rules where possible to ensure uniformity.
- Ensure compatibility with various devices, especially mobile phones.

Restrictions

Adhere to the following restrictions:

- You **cannot use** frameworks such as Laravel, jQuery, or Bootstrap.
- Consult with your instructor before using any small helper libraries.

Additional Requirements

Some suggested **additional** requirements. These requirements are a way of making sure each project is unique. You **do not have** to implement all of these:

- **Rating and Review System**: A feedback mechanism for users to rate products and sellers.
- **Promotional Features**: Options for sellers to feature their products on the main page through paid promotions.
- **Analytics Dashboard**: A tool for sellers to track performance metrics and sales data.
- **Multi-Currency Support**: A system to handle transactions in various currencies.
- **Item Swapping**: A feature allowing users to exchange items instead of monetary transactions.
- **API Integration**: Develop a robust API to allow third-party applications to interact with the platform.
- **Dynamic Promotions**: Implement a system for time-limited discounts, bundle deals (e.g., buy two, get one free), and quantity-based discounts (e.g., 10% off on purchases over a certain amount).

- **User Preferences**: Allow users to set and filter searches based on personal preferences such as size, condition, gender-specific items, etc.
- **Shipping Costs**: Calculate shipping costs depending on location (crude estimate).
- **Real-Time Messaging System**: A secure and efficient communication channel for buyers and sellers.

Remember, these are just suggestions to inspire further development and innovation. If you prefer, you can create your own additional requirements.

Work Plan

This is a proposed plan to guide your work. **No deliverables are expected** or will be evaluated on these dates:

- Week 7: Mockups and navigation diagrams for the main pages. First draft of the database design.
- Week 8: Finalize database script and create the database; partial implementation of some main pages and first CSS version.
- Week 9: Most main pages implemented.
- Week 10: All main pages implemented. Start working on secondary features.
- Week 11: Continue working on secondary features. Start working on Javascript and Ajax.
- Week 12: REST API or other secondary features. Testing and code cleanup.
- Week 13: Security concerns. Testing and code cleanup.

We recommend that students adopt an **agile** methodology. Don't start by planning every little detail right from the start, as you risk ending up with a great plan but a poor implementation. Be aware of code organization and quality from the beginning.

Evaluation

Evaluation will be done on the following **topics**:

- Complexity (*e.g.*, implemented features).
- Security (*e.g.*, XSS, CSRF, injection, password storage).
- Technology (*e.g.*, correct usage of HTML, CSS, Javascript, Ajax, No frameworks).
- Quality (*e.g.*, code quality, file organization, consistency).
- User Interface (*e.g.*, usability, consistency).

The goal is to implement the requirements in a **unique way** that meets the project objectives. Using code from other sources **without proper attribution** will lead to the student failing the class (possibly with an RFR mark).

Delivery

- Delivery until the 19th of May at 23:59 (WEST).
- Demo in last week's practical class (using the delivered version).

You must create a tag named "final-delivery-v1" on the commit you wish to deliver.

```
git tag final-delivery-v1
git push origin final-delivery-v1
```

To test if everything is correct, you should be able to clone the project into an empty directory:

```
git clone <your_repo_url>
git checkout final-delivery-v1
php -S 9000
```

And view your website at http://localhost:9000/.