|  |
| --- |
| Report (LaxStore) |
| Development Process |

|  |
| --- |
| 2024 | By: Vhugala Mutshembele |

****

### **LaxStore Project Report**

**Project Name**: LaxStore  
**Developer**: Vhugala Mutshembele (Solo Developer)  
**Technologies Used**: Python, Flask, HTML5, CSS, JavaScript, SQL (MySQL for database management), APIs, bcrypt for security, and Flask-Mail for automated emailing  
**Current Status**: Ongoing

**1. Project Overview**

**LaxStore** is a vibrant and dynamic online platform designed for streetwear enthusiasts, offering a centralized marketplace where users can browse, select, and purchase a wide range of streetwear brands. LaxStore is tailored to cater to the needs of both buyers and sellers, providing an intuitive, secure, and engaging shopping experience.

The platform integrates features such as user registration, product filtering, shopping cart functionality, checkout and payment processing, automated email receipts, and the ability to store user purchase history.

LaxStore seeks to redefine the shopping experience for streetwear lovers, creating an online space that celebrates the essence of streetwear culture through a sleek and functional website.

**2. Project Objectives**

* Create a dynamic and secure e-commerce website for streetwear enthusiasts.
* Allow users to browse and filter products from various brands.
* Implement a fully functional shopping cart system.
* Enable secure user registration, authentication, and login functionality.
* Process purchases, calculate totals, apply taxes, and send automated email receipts.
* Provide sellers with a platform to list their products and manage their inventory.
* Ensure a seamless user experience with a responsive design and effective product management.

**3. Key Features**

**User Features**

* **User Registration and Authentication**: Users can register, log in, and manage their accounts. Passwords are securely hashed using bcrypt for enhanced security.
* **Product Browsing and Filtering**: Users can browse through various streetwear brands and filter products based on categories.
* **Shopping Cart Functionality**: Users can add products to their cart and proceed to checkout. Cart data is stored locally using browser storage to manage shopping sessions.
* **Checkout Process**: Users can view the cart summary, calculate totals (including VAT), and make payments.
* **Automated Email Receipts**: Once a purchase is made, users automatically receive a detailed receipt via email.
* **Purchase History**: Users can view their purchase history, which is stored in the system’s database.

**Seller Features**

* **Product Management**: Sellers can list their products with specific details like price, size, and availability.
* **Inventory Management**: Sellers can manage their brand and product listings through an admin panel.
* **Brand Management**: Sellers can register their brand and store relevant details (e.g., contact information, social links, banking details).

**Admin Features**

* **Database Management**: LaxStore's data is stored in a MySQL database, including user information, product details, purchase history, and more.
* **Security Features**: User passwords are hashed using bcrypt, and automated messages and purchase receipts are handled securely using Flask-Mail.

**4. System Architecture**

The LaxStore project is built using the following system architecture:

* **Front-End**: HTML5, CSS, and JavaScript are used for the user interface, ensuring the platform is responsive and user-friendly. Pages such as the index (home page), access (login and registration), and checkout are designed with attention to aesthetics and functionality.
* **Back-End**:
  + **Python (Flask)**: Flask is used to manage the back-end logic, handling routes, and server-side processing. It handles requests for user registration, login, product management, and purchase processing.
  + **SQL (MySQL)**: MySQL is used for persistent storage, with tables such as Registration, Sign\_In, Financials, History, and Products\_Available managing critical user and product data.
* **Security**: bcrypt is used to hash user passwords securely, while Flask’s session management ensures users remain authenticated during their sessions. Sensitive user data (such as passwords and email addresses) is protected.
* **Email Automation**: Flask-Mail is configured to send automated email receipts to users after purchases. Emails are generated with full purchase details, including product breakdown, tax calculations, and final total amounts.

**5. Database Structure**

The database is composed of several key tables:

* **Registration Table**: Stores user details such as username, email, phone number, and hashed passwords.
* **Sign\_In Table**: Holds login credentials for user authentication.
* **Financials Table**: Tracks purchase information, including item details, prices, taxes, and payment amounts.
* **History Table**: Logs all purchases, recording receipt details and storing them for future reference.
* **Brands Table**: Stores brand information and links products to sellers.
* **Products\_Available Table**: Lists available products, categorized by brand, item type, and other attributes.

**6. Challenges Encountered**

* **User Data Security**: One of the main challenges was ensuring secure password storage and authentication. Using bcrypt for hashing passwords was a crucial step in enhancing security.
* **Automated Email Sending**: Integrating Flask-Mail to send automated purchase receipts required careful configuration to ensure emails were reliably sent upon successful purchase.
* **Database Management**: Properly managing the structure of the MySQL database, including ensuring efficient relationships between tables (such as users, purchases, and products), was essential for smooth data processing.

**7. Installation & Setup**

**Prerequisites**

* Python (3.7 or higher)
* MySQL database
* Required Python packages (Flask, Flask-Mail, Flask-SQLAlchemy, bcrypt, etc.)

**Packages Required**

To install the required packages, use:

To install the required packages, use:

pip install Flask Flask-Mail Flask-SQLAlchemy bcrypt mysql-connector-python python-dotenv Flask-WTF

**Setting up the Database**

1. Install MySQL on your local machine.
2. Create a database named LaxStore.
3. Run the SQL script lax\_tab.sql to create all necessary tables.

**Running the Application**

1. Clone the repository and navigate to the project directory.
2. Run the main application file using:

python lax\_access.py

python lax\_man.py

1. Open your web browser and navigate to http://laxstore.netlify.app/

### **8. Testing & Future Enhancements**

LaxStore is currently in the testing phase. Several test cases are being implemented to ensure the integrity of the user registration, login, and purchase flow. Testing is also focusing on database performance, email functionality, and security.

**Future Enhancements**:

* Implement payment gateways to handle real-time transactions.
* Expand product filtering options to enhance the user shopping experience.
* Develop a mobile application version to expand accessibility.

### **9. Conclusion**

LaxStore represents a robust and scalable online platform for streetwear enthusiasts. The project, though in its early stages, demonstrates a comprehensive approach to e-commerce development with an emphasis on security, functionality, and user experience. As the project progresses, LaxStore will continue to evolve into a fully realized public-facing platform for streetwear brands and customers.

For any questions or feedback, feel free to contact the developer, Vhugala Mutshembele, at vhugalagabriel@gmail.com.