



Group Members:

Athanase Abayo

Mabinty Mambu

Olivier Kwizera

Victoria Ama Nyonator

Course Name: Web Technologies

Cohort B

Sprint Final Project Report

Lecturer: Dr. Osafo-Mafo Kwadwo

Teacher Assistant: Barbara-Marie Doh

November 30, 2025

Background

SwapIt is a secure and sustainable digital sharing-economy platform designed to help individuals, students, and small businesses access items without needing to buy or own them permanently. Built on the principles of trust, affordability, and community empowerment, the platform enables users to borrow, lend, rent, or freely share everyday items: maximizing the value of underused assets while reducing wasteful consumption. Many people struggle with purchasing items they only need occasionally, overspending on tools or equipment, or lacking a reliable network of people willing to lend what they need. SwapIt solves this problem by providing a single, flexible online marketplace where users can easily post items, browse available resources, request access, and manage their listings. Through this connected system, regular users and small businesses can save costs, declutter their spaces, generate value from unused items, and support a more circular, sustainable lifestyle while the administrator oversees platform operations and ensures smooth functionality.

Functionality

The website includes **4 major features**:

1. User Authentication

- Login & validation system (auth.php)
- Protects swapping features so only logged-in users can post or interact

2. Item Posting / Upload

- Users can post items they want to swap
- Items are stored in a database (db/ folder)
- Includes data validation and image upload (via public/)

3. Swap Item Viewing

- Users can browse available items
- Pulls item info from the database
- Allows search & filtering (based on your interface logic)

4. Item Update & Management

- Users can edit or remove posts
- System checks database updates (test-update.php)
- Admin can manage listings when necessary

Implementation

Technologies Used

- Backend: PHP, MySQL
- Frontend: JavaScript, HTML, CSS
- Security: Bcrypt, Prepared Statements, Input Sanitisation, Rate Limiting
- Storage: localStorage (client), MySQL (server)
- Version Control: GitHub
- Development: PHP Development Server

Frameworks/Libraries

- **PHP**
- Basic **AJAX/Fetch API** from JavaScript in your test HTML files
- **MySQL** database (via db/configuration)

APIs

Located in /api/:

- auth.php → handles login, signup, sessions
- test-db.php → checks database connectivity
- test-update.php → sample update endpoint
- php-info.php → server and PHP configuration details
- Google api authenticator

Database Structure

Found in /db/:

- contains SQL files defining tables for:

- users
- items
- swaps or exchange history

Flowchart

User → Login → View Dashboard → Upload Item → System Stores in DB → Other Users View → Swap Happens

Classes

JavaScript Classes (8) and how they function:

- **Translation Manager & Language Switcher – Handle** multilingual translations (English/French) and UI language switching.
- **DashboardManager**: loads and displays user-specific data like listings, orders, and activity.
- **ProfileManager**: manages user profile editing, avatar uploads, and form validation.
- **BrowseFilter**: enables item filtering, searching, and sorting on the browse page.
- **CartManager**: handles shopping cart and wishlist, stores data in localStorage, and updates counters/notifications.
- **LoginHandler & SwapItAuth** : manage authentication, sessions, and login/signup flows.

How It Works Together:

- **Authentication Flow**: LoginHandler → SwapItAuth → DashboardManager → personalized dashboard.
- **Translation Flow**: TranslationManager → LanguageSwitcher → automatic page translation.
- **Cart/Wishlist Flow**: CartManager handles adding/removing items with live UI updates.
- **Browse/Filter Flow**: BrowseFilter enables instant filtering and searching of items.

- **Profile Management Flow:** ProfileManager and SwapItAuth manage profile updates and avatar uploads.

Testing/Validation

1. Functional Testing

- We tested each feature of the website (login, signup, forms, navigation, item listings, etc.)
- Verified that each component worked according to the requirements.

2. User Interface (UI) Testing

- Checked spacing, alignment, colors, responsiveness, and usability across pages.
- Ensured buttons, links, and forms behaved correctly.

3. Real-Time Validation Testing

- Tested input fields to confirm that invalid data triggered immediate error messages (e.g., wrong email format, empty required fields).

4. Cross-Browser Testing

- Opened the website on different browsers (Chrome, Firefox, Edge) to ensure consistent performance.

5. Device/Responsiveness Testing

- Tested the site on a laptop and mobile view to confirm that the layout adjusts properly.

6. Database Validation

- Verified that data is being stored, retrieved, updated, and deleted correctly (CRUD operations).

- Checked whether invalid data was prevented from entering the database.

7. User Testing

- Invited classmates/group members to use the system and report issues.
- Observed them to see if navigation was intuitive.

8. Bug Fixing & Retesting

- Documented errors during testing.
- Fixed the bugs and retested until they were resolved.

Key Achievements for SwapIt

- Secure authentication with session management and rate limiting
- Comprehensive security implementation using 9 OWASP vulnerabilities
- Full-stack application with PHP backend and JavaScript frontend
- Responsive design across devices
- Complete user journey from signup to browsing, cart, and checkout
- Database architecture with automatic fallback for development
- Security logging system for audit and monitoring
- Professional documentation for maintenance and deployment
- Integrating BIA languages, French, and English to improve accessibility and user experience across diverse audiences.
- Implementation of animation on our website

Unaccomplished Feature

- **User-to-user interaction (messaging system):**
We were not able to implement a real-time messaging feature that would allow different users to communicate or interact directly on the platform.
- Implementing Security Misconfiguration

We attempted to deploy our website on **Vercel, Netlify, GitHub Pages, Render, and Firebase Hosting**, but the platforms consistently returned errors indicating that they **could not host our HTML, CSS, and JavaScript files together**. This was mainly due to configuration issues and missing build settings required for proper static site deployment, leading to an unsuccessful hosting process.

Links

Link to the project on GitHub

<https://github.com/athanase02/group5-swapit>

Link to WordPress or CMS

<https://swapit-68c9a5.webflow.io/>

Link to live server

<http://localhost:3000/home.html>