## Website Technology Stack (Full-Stack)

We will use a robust, Java-centric stack to honor the original project's language choice, but adapt it for the web.

| Layer | Component | Description & Role |
|---|---|---|
| **Frontend** (Client) | **HTML5, CSS3, JavaScript** (or React/Vue.js) | Builds the interactive, user-friendly interface (**UI/UX**) on the browser. Ensures responsiveness across devices. |
| **Backend** (Server) | **Java Spring Boot** & **RESTful APIs** | Handles all the business logic, security, session management, and communication with the database. **Secured Login**, **Cart Logic**, and **Invoice Generation** happen here. |
| **Database** | **PostgreSQL** or **MySQL** | Stores all persistent data: **User Credentials**, **Product Catalog**, **Order History**, and **Cart contents**. Replaces the file-based storage. |

Export to Sheets

---

## Website Architecture and Data Flow

The system will be structured around four main web modules, corresponding to the paper's modules.

### 1. Secure User Authentication Module (Spring Security)

| Feature | Implementation Detail |
|---|---|
| **User Registration** | Frontend sends user data to a Spring Boot API. Backend hashes the password using **BCrypt** before storing it in the PostgreSQL database. |
| **Login Validation** | User submits credentials. Spring Security verifies the hashed password. Upon success, a **secure session** is created, allowing the user to access protected pages. |
| **Security** | Uses **HTTPS** (SSL/TLS) for encrypted communication and **CSRF protection** to prevent unauthorized command execution. |

Export to Sheets

### 2. Product Display & Catalog Module

| Feature | Implementation Detail |
|---|---|
| **Product Data** | Product details (name, price, image URL, description) are stored in the database. |
| **Catalog Page** | A Backend API (`/api/products`) retrieves all product data. The Frontend uses **JavaScript** to dynamically render the product cards, making the display visually appealing and fast (as requested in the paper). |
| **Visuals** | Product images are stored in a cloud service (like Amazon S3) or a separate asset folder, with only their URLs saved in the database. |

Export to Sheets

## 3. Shopping Cart Module (Session & Database Management)

| Feature | Implementation Detail |
|---|---|
| **Add/Remove Item** | When a user clicks "Add to Cart," the Frontend sends an asynchronous request (**AJAX**) to the Backend API (`/api/cart/add/{productId}`). |
| **Cart Storage** | For logged-in users, the cart is permanently stored in a **Cart table** linked to their User ID. For guests, a temporary **session cookie** holds the cart ID. |
| **Total Calculation** | The Backend recalculates the running **Total Cost** every time an item is added or removed, ensuring accuracy before displaying it on the Cart Page. |

Export to Sheets

## 4. Checkout & Invoice Generation Module

| Feature | Implementation Detail |
|---|---|
| **Order Confirmation** | The user initiates checkout. The Backend locks the items (temporarily reducing inventory), calculates the final bill, and waits for payment. |
| **Payment Integration** | **(Future Scope)** Integrate a real gateway like **Stripe** or **PayPal**. The system redirects the user for secure payment processing. |
| **Order Processing** | After successful payment, the Backend finalizes the transaction, moving the cart contents to the **Orders table**. |
| **Invoice Generation** | A dedicated Java utility generates a professional **PDF Invoice** (using a library like iText) based on the order details. This invoice is saved to the file system (or cloud storage) and emailed to the user. |

Export to Sheets

## Key Advantages of this Web Implementation

| Advantage | Benefit Over Desktop App |
|---|---|
| **Accessibility** | Users can shop from **any device** (desktop, mobile, tablet) using just a browser. |
| **Scalability** | Spring Boot and a database allow the system to handle thousands of concurrent users and products easily. |
| **Real-World Security** | Leverages **Spring Security** and HTTPS, the industry standard for securing e-commerce transactions and user data. |
| **Data Integrity** | Replacing file storage with a **Database** ensures data is always consistent, backed up, and can be queried efficiently. |