iBay Project Report - Group 4

Sean Mason, Drew Mistry, Damian Laubscher, Atri Majumder

May 14, 2025

Website URL:
`http://295group4.sci-project.lboro.ac.uk/`

Test login credentials:
Username: `test`, Password: `test1`

The groups responsibilities:

Sean - `Team Leader, Back-end, DataBase and Server Management`
Drew - `Front-end, Web Design and Product Direction`
Damian - `Front-end, Report and Bug-Fixing`
Atri - `Back-end`

Project Overview

The iBay website is an online marketplace that allows users to browse, list, and manage products. It includes functionality such as user authentication, listing creation, filtering, and pagination. The project uses a client-server architecture, combining PHP for back-end logic and JavaScript for front-end interactivity. The visual theme of the site is purple and white for consistency and aesthetics.

Webpages and Features

Navbar and Footer

- **Navbar:** The navigation bar (navbar) is critical for the website, it provides navigation and search functionality. It is dynamically loaded onto each page of the website, including the main page, by using a navbar.php snippet and insert script. its style is handled by the mainstyle.css stylesheet. It is divided into three parts:
  - iBay logo
    - The iBay logo is positioned on the left side of the navbar, and when clicked, the user is redirected to the home page of the website using simple HTML.
  - Search bar
    - In the center of the navbar is the search bar. Consisting of a search field and search button, it allows the user to search for products no matter where on the site they are located. If there is a match with the user search, the user is redirected to the products with only that for which they have searched. If the user's search is not found, they are still relocated to the products page with all products in the database. This will also happen if the user does not enter anything.
    - This is done by using the search.php file which, when the search bar is submitted, receives a GET request. The input element with id="searchInput" is where the user types their search query. The name ="q" attribute makes it that the query is sent as a URL parameter when the form is submitted. The ¡button element with class="search-button" submits the form when clicked. A placeholder is used to ensure the search bar looks presentable when not in use.

```
1 <?php
2     require_once("connection.php");
3
4     // Check if the search query exists
5     if (isset($_GET['searchInput']) && !empty(trim($_GET['searchInput']))) {
6         $query = "%" . $_GET['searchInput'] . "%"; // Wildcard for LIKE
      query
7
8         // Prepare and execute the SQL statement
9         $stmt = $conn->prepare("SELECT itemId FROM iBayItems WHERE title
      LIKE ? OR description LIKE ?");
10        $stmt->bind_param("ss", $query, $query);
11        $stmt->execute();
12
13        $result = $stmt->get_result();
14    ...
15
```

* User Account section
  - On the right of the navbar is the search bar. If the user is not yet signed in, it displays sign-in button which, when pressed, redirects the user to the sign-in page.
  - If the user is signed in, the user's name is displayed instead. When pressed a dropdown menu opens up with the options to: redirect to the "My Account" page, redirect to the "My Listings" page, or logout of the current account. The logout is done by using logout.php which works by removing the session variables, destroying the session and redirected the user to the login page.
  - This is all done using a combination of php and JavaScript

```
div class="header-right">
    <div class="account-dropdown">
      <?php if (isset($_SESSION['userId'])): ?>
        <div class="dropdown">
          <button class="account-btn" onclick="toggleDropdown()">
            <i class="fas fa-user"></i> <?php echo htmlspecialchars(
    $_SESSION['userId']); ?>
          </button>
          <div class="dropdown-content" id="accountMenu">
            <a href="profile.php"><i class="fas fa-user-circle"></i> My
    Account</a>
            <a href="my_listings.php"><i class="fas fa-list"></i> My
    Listings</a>
            <a href="php/logout.php"><i class="fas fa-sign-out-alt"></i>
    Log out</a>
          </div>
        </div>
      <?php else: ?>
        <a href="login_page.html" class="account-btn">Sign In</a>
      <?php endif; ?>
    </div>
</div>

```

* **Footer:** The footer, like the navbar, is also dynamically loaded onto the webpage. It uses the footerLoader.js script to do this.
  - The footer contains a copyright notice which is displayed within a popup container. The footerLoader.js script ensures that the footer remains at the bottom of the page, even if the content is short, by using CSS flexbox techniques.

Main Webpage

* **Overview:** The home page of the iBay website serves as the entry point for the user, providing an interface and navigation features such as product browsing, user account management, and creating new listings. It follows the theme of the website being purple and white. It is divided into several key sections, each with specific functionality to enhance usability and engagement. The page is designed to be fully responsive, ensuring usability across devices of various screen sizes using mainstyle.css and home.css stylesheets. Font Awesome icons are used throughout the page and website to enhance visual appeal and improve navigation clarity. It is divided into three sections:

* Welcome to iBay
  - This section welcomes the users to the platform with a welcoming message and call-to-action buttons. The section is a lovely purple to standout at the users and holds two buttons: "Start Shopping" and "Start Selling". These buttons redirect the user to the Products or "New Listing" pages and are styled to make them pop when the user hovers over the button to increase interactivity.

```
<section class="hero-section">
      <div class="hero-content">
        <h1 class="hero-title">Welcome to iBay</h1>
        <p class="hero-subtitle">Your one-stop marketplace for buying and
    selling</p>
        <div class="hero-buttons">
          <a href="products.php" class="cta-btn primary">
            <i class="fas fa-shopping-cart"></i> Start Shopping
          </a>
          <a href="new_listing.php" class="cta-btn secondary">
            <i class="fas fa-tag"></i> Start Selling
          </a>
        </div>
      </div>
</section>


```

* Popular Categories
  - This section displays the four popular categories on the site: Electronics, Fashion, Home and Garden, and motors. Each of these categories is given an appropriate icon and a "Browse" button to relocate the user to the products page with the page already filtered to the users selected category.
  - This is achieved using the browse-category-btn class, which links to the page products.html?category=givenCategory where the givenCategory is the one selected by the user.
* Ready to get started?
  - This bottom section is styled very similarly to the "Welcome to iBay" section but serves a secondary call-to-action. It has the same buttons as the "Welcome to iBay" section which are renamed. The purpose for this section is to add volume to the main page while also promoting engagement of the user by exploring the products and creating listings.

Login Page

* **Overview:** The login page allows users a secure way to access their accounts on the iBay website. It is designed to ensure that the theme of the website is preserved while allowing the user to sign in using the mainstyle.css and login.css stylesheets. The page includes form validations, links to the sign-up page or return back to the home page, and integration with the back-end for authentication. The login page, like all other pages on the website, also has the navbar and footer at the top and bottom of the page.
* The login form
  - The login form is the main component of the page. The form uses the POST method to securely send user credentials to userLogin.php for authentication. The user has to input their username and password in the designated fields. Each field is given a placeholder and icon to make sure it is clear and easy for the user to understand. The user then submits the form by pressing the "Log In" button. They are also giving the option to sign up if they don't have an account or return to the home page.
* The userLogin.php script

- It first establishes a connection to the database by calling connection.php and then initializes a session to store user-specific data like the user's ID. It then retrieves the user's input (userID and password) from the POST request
- It then prepares and executes the SQL query to securely query the database for the hashed password associated with the provided userID.
- If the user exists in the database, their hashed password is fetched and the script verifies the given password.

```
<?php
require 'connection.php';
require_once 'popup.php';

session_start();

$userId = trim($_POST['userId']);
$password = $_POST['password'];

$stmt = $conn->prepare("SELECT password FROM iBayMembers WHERE userId = ?");
$stmt->bind_param("s", $userId);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows === 0) {
    redirectWithPopup("../login_page.html", "Username does not exist");
}

$stmt->bind_result($stored_hashed_password);
$stmt->fetch();

if (password_verify($password, $stored_hashed_password)) {
    $_SESSION['userId'] = $userId;
    redirectWithPopup("../main.php", "Log in successful!");
} else {
    redirectWithPopup("../login_page.html", "Incorrect password");
}

$stmt->close();
?>
```

Sign up Page

* **Overview:** The sign up page allows users a secure way to set up an account on the iBay website. It collects all the information: username, full name, email, address, postcode, and password. The form submission is handled by userSignup.php script, which processes the data, validates it, and stores it in the database. The page is styled with signup.css and mainstyle.css to keep the theme of the website consistent.
* The sign up form
  - The sign up form is the main component of the page. The form uses the POST method to securely send user data to the userSignup.php script. All the input fields are marked with an icon and placeholder to ensure clarity for the user. The form is then followed by the "Create Account" button to submit the form. At the bottom of the page the user is also given the option to log in if they already have an account or to return back to the home page.
* The userSignup.php script
  - It first establishes a connection to the database by calling connection.php and retrieves the user's input from the form. It then validates the users input and checks if the userID or email is already in use on the iBay website. The password is then hashed and the new user is inserted into the iBayMembers table. Finally, the user is then redirected to the home page.

My Account Page

* **Overview:** The my account page allows users access to their personal information: username, full name, email, address, and postcode. It dynamically fetches user data from the server and displays it in a clean and organized layout. The page also includes functionality to redirect the user to the login page if they are not authenticated, ensuring secure access to profile information. The page uses signup.css and mainstyle.css to ensure a consistent theme across the website.
* Profile container
  - The container displays all the personal information of the user which is fetched using JavaScript code.
  - The JavaScript checks the authenticity of the user by checking if there is a userID in the php session, and then passing that into a javascript SessionStorage. If none is found the user is redirected to the login page.
  - It then fetches the personal information by using a GET request to the user_data.php script. If the request is successful, the response is parsed as JSON, and the user's profile information is updated in the DOM. If the request fails, an error message is displayed for each field.
* The edit_user.js script
  - The javascript file fetches the user data from user_data.php. Once the data has been retreived, it imparts it into input fields as place holders, allowing the user to click save and re upload a form which updates their user data in the database.

My Listings Page

* **Overview:** The my listings page allows the users to see their current listings that they have posted on the iBay website. It dynamically loads the user's listings from the server and displays them in a grid format. The page also includes pagination for navigating through multiple listings and a button to create more listings. The style is maintained using myListings.css and items.css. There is three sections of the page as well as the use of both getItems.js and getItem.php scripts:

* Header and Filters section
  - Displays the title of the page “My Listings” and button to create a new listing using a simple HTML to link it to the new listing page.
  - This section also contains filters to filter the listings of the user. You can filter by category, price range or sort by price (high to low or low to high).
* Listings grid
  - This section dynamically displays the user's product listings in a grid format. Each listing is represented as a card, which is injected into the DOM using the get_items.js script.
* Pagination
  - This section allows users to navigate through multiple pages of their listings. The "Previous" button goes to the previous page, the numbered button is the current pages number and the next page button goes to the next page if that is available.
* The get_items.js script
  - This script starts retrieving all filter information and passing it to get_listings.php. This executes the relevant search query and returns all items to the js, allowing them to be put into cards and shown on the page. Due to our implementation, seeing your listings is a due to a filter in the search query. However, your cards will load with an 'edit' button so you cange the information on them and save them.

```
1  // Function to update URL with current filters
2    function updateURL(params) {
3      const url = new URL(window.location.href);
4      url.search = params.toString();
5      window.history.pushState({}, '', url);
6    }
7
8    // Function to fetch items
9    function fetchItems(page) {
10     const params = buildFilterParams();
11     params.append('page', page);
12
13     // Update URL with current filters
14     updateURL(params);
15
16     fetch(`php/get_listings.php?${params.toString()}`)
17       .then(res => res.json())
18       .then(data => {
19         const container = document.getElementById('listings-container');
20         const prevPageBtn = document.getElementById('prevPage');
21         const nextPageBtn = document.getElementById('nextPage');
22         const currentPageSpan = document.getElementById('currentPage');
23         const sessionUserId = data.sessionUserId;
24
25         if (!Array.isArray(data.items) || data.items.length === 0) {
26           container.innerHTML = '<div class="empty-state"><p>No items
       found matching your filters.</p></div>';
27           return;
28         }
29
30         container.style.display = "grid";
31
32         // Update total items and pagination
33         totalItems = data.total || data.items.length;
34         const totalPages = Math.ceil(totalItems / ITEMS_PER_PAGE);
35
36         // Update pagination controls
37         prevPageBtn.disabled = page <= 1;
```

```
38          nextPageBtn.disabled = page >= totalPages;
39          currentPageSpan.textContent = page;
40
41          // Clear existing items
42          container.innerHTML = '';
43
44          // Add new items
45          data.items.forEach(item => {
46            const div = document.createElement('div');
47            div.className = 'listing-item';
48
49            // Use the first image, fallback to placeholder
50            const imageUrl = item.images && item.images.length > 0
51              ? item.images[0]
52              : '../assets/placeholder.jpg'
53
54              div.innerHTML = `
55              <div class="item-card">
56                <img src="${imageUrl}" alt="${item.title}" class="item-
    thumbnail">
57                <h3>${item.title}</h3>
58                <p> $ {item.price}</p>
59                ${item.userId == sessionUserId
60                  ? `<button class="edit-button" onclick="editItem(${item.
    itemId})">Edit</button>`
61                  : `<button onclick="viewItem(${item.itemId})">View</
    button>`}
62              </div>
63            `;
64
65
66            container.appendChild(div);
67          });
68        })
69        .catch(err => {
70          console.error("Error fetching listings:", err);
71        });
72    }
```

New Listing Page

* **Overview:** The new listings page allows users to create and submit a new product to the database. The user is ensured to enter the item details: title, category, price, description and images. The page is designed to keep the theme of the website with the stylesheets of myListings.css and mainstyle.css. Like all other pages it has the navbar and footer appended at the top and bottom. The page is in two parts and incorporates the use of addFiles.js and newListing.php scripts:
* Header section
  - The header section consists of the header "Add New Listing" next to the "My Listings" button which directs the user to their existing listings HTML. The form allows the user to input the following data: item name, category, price, bid start date, bid end date, postage information, item description, and an option to upload an image. The process of dynamically previewing the uploaded images is handled by add_files.js.
* Form section
  - The form uses the add_listing.php script which processes the form submission and uses the POST method to ensure the data is sent securely.
* The add_files.js script

- When the image is selected the script generates a preview for each image. The script allows users to add or remove images dynamically before submitting the form and validates the selected image for file type and size before previewing it.

* add_listing.php script
  - This script starts a session and checks if the user is logged in. If they are not, the user is redirected to the login page. It then reads and validates the form data, checking if they are valid entries. For example with the price there is a price limit. Then the listing is inserted into the database. Then the user is redirected back to the my listings page. Here is just a little bit of the code:

```php
<?php
require 'connection.php';
require_once 'popup.php';

// Start the session if you need to check if a user is logged in
session_start();

// Redirect with a popup if user is not logged in
if (!isset($_SESSION['userId'])) {
    redirectWithPopup('../login_page.html', 'Please log in to list an item');
}

// Read POST data safely
$title = trim($_POST['title']);
$category = $_POST['category'];
$price = floatval($_POST['price']);
$postage = trim($_POST['postage']);
$description = trim($_POST['description']);
$start = $_POST['start'] ?: null; // Optional field
$finish = $_POST['finish'];

// Additional validation
if ($price < 0 || $price > 1000000) {
    redirectWithPopup('../new_listing.php', 'Invalid price entered');
}

if (!$finish) {
    redirectWithPopup('../new_listing.php', 'You must enter a finish date');
}

// Check that finish date is not before start date (if start date provided)
if ($start && strtotime($start) > strtotime($finish)) {
    redirectWithPopup('../new_listing.php', 'Finish date must be after start date');
}

$stmt = $conn->prepare("INSERT INTO iBayItems (userId, title, category, price, postage, description, start, finish) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
$stmt->bind_param("ssssssss", $_SESSION['userId'], $title, $category, $price, $postage, $description, $start, $finish);
$stmt->execute();
$stmt->close();
?>
```

Products Page

* **Overview:** The products page is the central hub for browsing and filtering the product listings. It allows users to explore the range of products based on categories, and the price range. Use of mainstyle.css, items.css, my_listings.css and products.css to maintain the theme of the website. The page also makes use of both get_listings.php and get_items.js. The page is split into three parts:
* Header and filters section
  - This is on the left side of the page and provides the user with filtering options based on category, price range of the products or sort by the price (high to low or low to high).
* Products grid
  - This is the main part of the page, which uses the scripts, get_items.js and get_listings.php, to fetch the items and display it. This is exactly the same way the my listings page does it, but with the filter to not see the users own items. No one wants to search through what they are selling.
  - The user is given the opportunity to view the product by pressing the view button that relocates them to the item template page:
  - This page shows the product and its details and allows the user to buy the product. This is handled using load_item.js.
* Pagination
  - At the bottom of the page, just like the my listings page, there is pagination controls to navigate the pages of products.

Client-Server Technologies

SQL Injection Avoidance

* **Concept:** SQL injection is a security vulnerability that allows attackers to manipulate SQL queries by injecting malicious input into user-provided fields. This can lead to unauthorized access, data breaches, or even complete control over the database. Preventing SQL injection is critical for maintaining the security and integrity of a web application.
* Technology used
  - In this project, prepared statements with parameterized queries are used to prevent SQL injection. Prepared statements ensure that user input is treated as data rather than executable SQL code, making it impossible for malicious input to alter the query structure.
* Example from the project
  - In the user_login.php script

```
1 $stmt = $conn->prepare("SELECT password FROM iBayMembers WHERE userId =
      ?");
2 $stmt->bind_param("s", $userId);
3 $stmt->execute();
4 $stmt->store_result();
5
```

  - The ? placeholder is used in the SQL query and the bindParam() method binds the user-provided userID to the query. This ensures that even if the userID contains malicious SQL code, it will be treated as a string and not executed.

- In the add_listing.php script

```
$stmt = $conn->prepare("INSERT INTO iBayItems (userId, title, category,
    price, postage, description, start, finish) VALUES (?, ?, ?, ?, ?, ?,
     ?, ?)");
$stmt->bind_param("ssssssss", $_SESSION['userId'], $title, $category,
    $price, $postage, $description, $start, $finish);
$stmt->execute();
$stmt->close();


```

- This approach ensures that all user inputs (e.g. title, category, price) are securely handled.

* Advantages of using prepared statements

- Prepared statements: prevents SQL injection by separating SQL code from user input, improves code readability and maintainability and enhances performance by reusing prepared queries for multiple executions.

Client-Side vs Server-Side implementation of Web Applications

* **Client-Side Implementation**

- **Advantages**
- **Improved user experience:**
- Client-side scripts (e.g., JavaScript) allow for real-time interactivity, such as dynamic content updates and form validation without reloading the page.
- Example: add_files.js dynamically previews uploaded images on the New Listings page.
- **Reduced server load:**
- By handling tasks like form validation and filtering on the client side, fewer requests are sent to the server.
- Example: Filters on the "Products" page dynamically update displayed items without requiring a full page reload.
- **Faster response times:**
- Since processing happens in the browser, users experience faster interactions.
- **Disadvantages**
- **Security risks:**
- Client-side code can be manipulated by users, making it unsuitable for sensitive operations like authentication or database queries.
- Example: Form validation on the client side can be bypassed, so server-side validation is still necessary.
- **Browser dependency:**
- Client-side functionality depends on the user's browser and its support for JavaScript.

* **Server-Side Implementation**

- **Advantages**
- **Security:**
- Server-side code is executed on the server, making it inaccessible to users and more secure for sensitive operations.

- Example: user_login.php securely verifies user credentials against the database.
- **Centralized control:**
- All logic and data processing occur on the server, ensuring consistency across users.
- Example: add_listing.php validates and processes form data before storing it in the database.
- **Cross-browser compatibility:**
- Server-side code is independent of the user's browser, ensuring consistent behavior.
- **Disadvantages**
- **Increased server load:**
- Handling all processing on the server can lead to higher resource usage, especially for large-scale applications.
- **Slower response times:**
- Server-side operations require communication between the client and server, which can introduce latency.

* **Examples of Client-Server Techniques in the Project**
  - **Client-side example:**
  - The get_items.js script dynamically fetches and displays product listings on the Products and My Listings pages. It uses the Fetch API to send requests to the server and updates the DOM without reloading the page.
  - **Server-side example:**
  - The get_listings.js script retrieves detailed information about a specific product from the database and returns it as a JSON response.
  - **Combined example:**
  - On the products page, filters are applied using a combination of client-side and server-side techniques:
  - The client-side script get_items.js sends filter parameters (e.g., category, price range) to the server.
  - The server-side script processes the parameters, retrieves the filtered data from the database, and returns it to the client.
* **How to choose between Client-Side and Server-Side Technologies**
  - **Security Requirements:**
  - Use server-side implementation for sensitive operations like authentication, database queries, and payment processing.
  - **Server-side example:**
  - The get_listings.js script retrieves detailed information about a specific product from the database and returns it as a JSON response.
  - Example: The user_login.php script securely verifies user credentials on the server.
  - **Performance Needs:**
  - Use client-side implementation for tasks that require real-time interactivity or reduced server load.:
  - Example: The get_items.js script previews uploaded images on the client side, reducing server requests.

- **Scalability:**
- For large-scale applications, offload non-sensitive tasks to the client side to reduce server load.
- **User Experience:**
- Use client-side techniques for dynamic and interactive features, such as filtering and pagination.
- **Cross-Browser Compatibility:**
- Ensure critical functionality is implemented on the server side to avoid browser dependency issues.

Evaluation

* What improvements we would have loved to implement
  - AJAX-based search bar
  - An AJAX-based search bar allows real time suggestions for the user when they are typing for their item to search. We, as a team, attempted to do this many times. However, due to technical restraints it was hard to test and implement.
  - Enable bidding system for auction-style listings
  - Right now there is only the option for the user to buy an item, it would have been nice for us to experiment on a bidding system instead.
  - Expand filtering options (e.g., location-based search)
  - On the site you can filter by categories or price. For user convenience it would have been useful to implement a location based filter as well.
  - Improve location and user-specific regional listings
  - The postcode information is pretty basic on the website. A map or other advanced settings would definitely have been achievable for us to implement.

Final thoughts

* Developing the iBay platform has deepened our understanding of secure web practices, user experience design, and client-server communication development, requiring close collaboration across both front-end and back-end responsibilities. We hope that the project demonstrated our technical ability to build a functional and responsive e-commerce website.
* Through the use of technologies like PHP, JavaScript, and prepared SQL statements, we built a secure and interactive site that successfully handles authentication, product listings, dynamic filtering, and more.