104346575@student.swin.edu.au

This readme file contains following contents:

- Files in the System

- File Explanation

- User Guide

104346575@student.swin.edu.au

List of Files in the System

HTML Files

- accessDenied.html
- login.html
- register.html

JavaScript Files

- bidding.js
- listItem.js
- login.js
- maintenance.js
- register.js

PHP FIles

- bidding.php
- buyItNow.php
- deleteItems.php
- fetchItems.php
- generateReport.php
- getCategories.php
- history.php
- listItem.php
- listing.php
- login.php
- logout.php
- maintenance.php
- placeBid.php
- processItems.php
- register.php

104346575@student.swin.edu.au

XML Files

- auction.xml
- customers.xml

CSS Files

- pagestyle.css
- style.css

XSL Files

- report.xsl

Images

- shoponline.png

104346575@student.swin.edu.au

File Explanation

HTML Files

- **accessDenied.html :** This HTML file creates a web page that displays an "Access Denied" message to users who attempt to access a restricted area without proper authorization. Additionally, the page provides a link to a login page (login.html), encouraging the user to log in to gain access. The overall purpose of this file is to prevent unauthorized access and guide users to the appropriate login page.
- **login.html :** This HTML file creates a login page for users to access a system. The document is titled "Login" and includes a script file (login.js) to handle the login process, along with an external CSS file for styling. The form includes input fields for the user's email and password, both of which are required. When the form is submitted, it triggers the processLogin() function from the JavaScript file to handle the login logic. Below the form, a message encourages users who are not registered to sign up by providing a link to the registration page (register.html). An empty span with the ID loginMessage is included to display any login-related messages dynamically.
- **register.html :** This HTML file creates a user registration page, titled "Register," which allows new users to sign up for an account. It includes input fields for first name, surname, email, password, and password confirmation, all of which are processed by the registerUser() function from an external JavaScript file (register.js).

JavaScript Files

- **bidding.js :** It immediately loads items from the server using the loadItems() function, which sends an asynchronous GET request to fetchItems.php and updates the content of an HTML element with the ID itemsList every 5 seconds. The placeBid() function allows users to place a bid by prompting them for a bid amount, then sending this data to placeBid.php via a POST request, displaying the server's response in an alert box. Similarly, the buyItNow() function enables users to instantly purchase an item by sending its ID to buyItNow.php through a POST request, also showing the response in an alert.
- **listItem.js :** It manages the interaction and dynamic content for listing an item on a webpage. The toggleOtherCategoryField() function controls the visibility of an additional input field, which appears when the user selects "Other" as the category from a dropdown

menu. The listItem() function processes the form submission, validating the number of days entered by the user to ensure it's a positive number. If valid, the function sends the form data to listItem.php using a POST request, updating a result div with the server's response. The populateCategories() function dynamically populates the category dropdown by fetching available categories from getCategories.php via a GET request and appending them to the dropdown, while always including an "Other" option.

- **login.js :** It handles the login process for a web application, ensuring cross-browser compatibility by creating an XMLHttpRequest object that works in both modern browsers and older versions using ActiveXObject. The processLogin() function is triggered when the login form is submitted. It retrieves the user's email and password from the input fields, then sends these credentials to login.php via a POST request, using the XMLHttpRequest object. The server's response is handled within the onreadystatechange event; if the login is successful (indicated by a "success" response), the user is redirected to the bidding.php page. If not, the error message returned by the server is displayed within an HTML element with the ID loginMessage. The function returns false to prevent the form from being submitted in the traditional way, ensuring the login process is handled entirely via AJAX.
- **maintenance.js :** It provides functionality to process items and generate reports on a web page through asynchronous requests. The processItems() function creates an XMLHttpRequest object to send a POST request to processItems.php. When the server responds and the request is complete (readyState == 4 and status == 200), the response text is displayed within an HTML element with the ID result. Similarly, the generateReport() function sends a POST request to generateReport.php, updating the same result element with the server's response once the request is successful. Both functions ensure that server interactions happen without reloading the page, allowing for dynamic content updates.
- **register.js :** It manages the user registration process in a web application. The registerUser() function collects user input from the registration form, including first name, surname, email, password, and password confirmation. Before proceeding, it checks that all fields are filled out and that the passwords match. If any validation fails, an error message is displayed in an HTML element with the ID message, and the form submission is halted. If the input passes validation, the function sends the data to register.php via a POST request, using the XMLHttpRequest object. The server's response is handled within the onreadystatechange event; if the registration is successful (indicated by a response containing "Registration successful"), the user is redirected to bidding.php. Otherwise, the

server's response, which may include error messages, is displayed in the message element.

PHP Files

- **bidding.php :** It has PHP and HTML code that manages the bidding page of a web application by first verifying user authentication through session variables. If the user is not logged in, they are redirected to the login page; otherwise, the page displays the content for bidding.
- **buyItNow.php :** This PHP script handles the "Buy It Now" functionality for an auction system. It starts by checking if the user is logged in by verifying the session; if not, it outputs an error message and stops execution. It then retrieves the item ID from the POST request and loads the auction data from an XML file. Using DOM and XPath, it searches for the item with the specified ID that is currently "in_progress." If such an item is found, it updates the item's bid price to the "Buy It Now" price, sets the bidder ID to the current user, and marks the item as "sold." The updated XML file is saved, and a confirmation message is sent back. If the item is not available or already sold, an error message is displayed instead.
- **deleteItems.php :** It provide an administrative interface for managing auction items. It begins by verifying if the user is logged in and has admin privileges; if not, it includes an access denial page, clears, and destroys the session. If the user is an admin, the script processes operations based on form submissions. It supports item deletion and searching through an XML file that stores auction data. If a delete request is received, the item specified is removed from the XML and the page reloads. For search requests, it filters items based on the provided ID or displays all items if no search term is entered. The resulting items are displayed in a table with an option to delete each item, and various statuses and messages are shown based on the results of the operations.
- **fetchItems.php :** It generates HTML content to display auction items based on their status and time remaining. It starts by setting the timezone to Sydney, Australia, and loading auction data from an XML file. It then iterates over each item, comparing the current date with the auction's end date. Depending on the status and timing, it classifies the item as "Sold," "Failed," "Processing," or "In progress," and calculates the remaining time if the auction is still active. The script formats the output for each item, including its details,

status, and time left. It also provides "Place Bid" and "Buy It Now" buttons if the auction is ongoing.

- **generateReport.php :** This page is intended for admin users, processes auction data to generate a report on sold and failed items. It first checks for administrative privileges and denies access if the user is not an admin. It then loads auction data from an XML file, categorizing items as either "sold" or "failed," calculating the revenue from each category (3% of bid price for sold items, 1% of reserve price for failed items), and updates the XML file accordingly. The script creates a new XML document to store detailed reports on sold and failed items, including their attributes, and calculates the total revenue. It applies an XSLT transformation to format the report using an XSL stylesheet and outputs the transformed XML as HTML.
- **getCategories.php :** It retrieves and returns a list of unique item categories from an XML file. It first checks if the XML file exists, then loads and parses it. It iterates through each item in the XML, collecting categories while excluding the "Other" category.
- **history.php:** It generates an HTML page displaying a user's bids. It first checks if the user is logged in by verifying the session. If not logged in, it displays an error message. It then determines if the user is an admin, adjusting the navigation menu accordingly. The script loads an XML file containing auction data and searches for bids associated with the logged-in user. If no bids are found, an error message is shown; otherwise, a table listing the user's bids is displayed.
- **listItem.php :** It handles the process of listing a new auction item. It starts by checking if a session is active and sets the default timezone. If the auction XML file does not exist, it creates a new XML structure. The script retrieves form data, validates the price settings, and calculates the auction end date. It generates a unique item ID and adds the new item to the XML file with details including the item name, category, description, and prices. The XML file is then formatted and saved. Finally, it outputs a success message with the item number, date, and time when the auction starts.
- **listing.php :** It generates an HTML form for listing an item for auction. It first ensures that a user is logged in by checking the session. If the user is not logged in, they are redirected to the login page. It then checks if the user has admin privileges. a form is provided for users to input details about the auction item, including the item name, category, description, and pricing information. There is also an option to specify the duration of the auction in days, hours, and minutes. JavaScript functions are used to handle dynamic elements like showing or hiding the category input field when "Other" is selected.

- **login.php :** It handles user login by starting a session and processing login credentials submitted through a POST request. It loads customer data from an XML file and searches for a customer with the provided email. If a matching customer is found, the script verifies the password against the stored value. Upon successful authentication, it sets session variables for the customer's ID and user type, distinguishing between 'admin' and 'customer' based on the email address. If authentication fails, an error message is displayed. The script ensures secure login handling and appropriate session management to control user access within the application.
- **logout.php :** It handles user logout by starting a session and then clearing all session variables to remove any stored user data. It proceeds by destroying the session entirely, which effectively logs the user out. After the session is terminated, the script redirects the user to the login.html page to prompt them to log in again.
- **maintenance.php :** It is designed to manage the access and display of a maintenance page for administrative functions. It first starts a session and checks if the user is logged in and has an admin type. If the user is not an admin, it includes an "access denied" page, clears the session variables, destroys the session, and halts further script execution. If the user is an admin, the script continues by displaying an HTML page with options for maintenance functions.
- **placeBid.php :** It handles the process of placing a bid on an auction item. It starts by initiating a session and loading the XML file that contains auction item details. The script retrieves the item ID and bid price from the POST request and the customer ID from the session. It then searches for the auction item with the given item ID in the XML file. If the item is found and the bid is valid (i.e., the auction is in progress and the bid price is higher than the current bid), it updates the item's bid price and bidder ID with the new values. The XML file is then saved with the updated information, and a success message is displayed. If the item is not found or the bid is invalid, an appropriate error message is shown.
- **processItems.php :** It manages the processing of auction items by updating their status based on the auction's end date and bid price. It starts by checking if the user is logged in as an admin; if not, it includes an access denied page, clears the session, and ends the script. If the user is an admin, the script sets the timezone to Sydney and loads the auction items from an XML file. It then iterates through the items, checking if each auction is in progress and if the current date has passed the auction's end date. Depending on whether the bid price meets or exceeds the reserve price, it updates the item's status to either 'sold'

or 'failed'. After processing, it saves the updated XML file and outputs a message indicating how many items were processed.

- **register.php :** It handles user registration by collecting and validating user information submitted via a POST request. It starts by initializing the session and preparing to output plain text. It checks for the existence of an XML file storing customer data, creating one if it does not exist. The script then verifies if the submitted email is already registered; if so, it returns an error message. If the email is unique, it generates a new customer ID, creates a new XML entry for the user, and saves the data to the XML file. The script also formats the XML for readability before saving. While it includes commented-out code for sending a registration confirmation email, this functionality is currently disabled to avoid performance issues. Finally, the script stores user details in the session and outputs a success message. Functions for generating unique IDs and random strings are also included to ensure the customer ID is unique and sufficiently randomized.

104346575@student.swin.edu.au

ShopOnline User Guide

Introduction

ShopOnline is a web-based selling and buying system that utilizes Ajax technologies to provide an interactive auction experience. This guide will walk you through how to use the system, including registration, listing items, bidding, and managing auctions.

Accessing the System

1. **Start XAMPP Server**: Ensure that the XAMPP server is running.
2. **Open Your Browser**: Navigate to http://localhost/Project2/login.html to access the login page.

Registration

1. **Access Registration Page**: On the login page, if you do not have an account, click on the "Register" link below the login button.
2. **Fill Out Registration Form**: Enter your details to create a new account.
3. **Submit Registration**: Click "Register" to complete the process. You will be redirected to the bidding page upon successful registration.

Login

1. **Enter Credentials**: On the login page, input your email and password.
2. **Submit Login**: Click "Login" to access the system. If your credentials are correct, you will be redirected to the bidding page.

Bidding

1. **View Auction Items**: On the bidding page, all available auction items will be listed.

2. **Place a Bid**: For any item you wish to bid on, click the "Place Bid" button at the end of the item detail. A popup will appear for you to enter your bid.
3. **Buy It Now**: Alternatively, click the "Buy It Now" button if you prefer to purchase the item immediately.
4. **View Auction History**: To view your previous purchases and current highest bids, select "History" from the top right corner dropdown menu.

Listing Items

1. **Access Listing Page**: From the top right corner dropdown menu, select "Listing" to go to the item registration page.
2. **Enter Item Details**: Fill out the form with the item details. Ensure the auction duration is at least 1 day.
    - **Category Selection**: Choose from the dropdown menu or select "Other" to enter a new category.
    - **Starting Price**: If not set, it will default to 0.
3. **Submit Listing**: Click "List" to list the item. A message with the auction start date, time, and unique item number will be displayed.

Admin Features

1. **Manage Auctions**:
    - **Access Manage Auction Page**: Select "Manage Auction" from the top right corner dropdown menu.
    - **Search and Delete Items**: Enter the item ID to view its details or delete the item if necessary.
2. **Maintenance**:
    - **Access Maintenance Page**: Select "Maintenance" from the top right corner dropdown menu.
    - **Process Items**: Handle items that are sold or expired. This feature is restricted to system administrators only.

104346575@student.swin.edu.au

Access Control

- **Admin Login**: Use the following credentials to access admin features:
  - **Email**: admin@shoponline.com
  - **Password**: admin

The above credentials are already in customer.xml file but when the program automatically creates customers.xml file it will not have these credentials so just register a new user with these credentials. The register page will redirect you to bidding.php lougout once and login again with those credentials to gain admin access as it will set user type in session as admin. This is only required if the customer.xml file is not copied from this project.

- **Access Denied**: Non-admin users attempting to access admin features will see an "Access Denied" message and be logged out.