Advanced Web Technologies

Student ID: 23020438

Online Grocery Store: Description and Working Procedure Report

Task	**Comments**
1. Grocery company webpage	All completed
2. Submit SQL files to generate the tables in the grocery company	All completed
3. Registration page	All completed, included CAPTCHA too
4. Login page (mention whether with/without Captcha)	All completed with CAPTCHA
5. Security aspects (preventing attacks) of the Registration and Login page	Security measures like cross site scripting (XSS) and Captcha for both pages
6. Search engine optimization aspects in the grocery company webpage	I added SEO for the website.
7. User Registration validation using React framework	The website uses live validation using React.
8. Data management	I used MySQL database to store the product details, customer and managers information
9. RESTful webservice for the grocery company Manager	The website uses a RESTful API for the manager to access orders that have been placed.
10. Any other information (remarks)	I added Captcha validation to the registration page too.
11. Manager login details	Username: manager@yahoo.com, password: faith1

Introduction

The website lets customers browse products, register, log in, and place orders. It also allows managers to view order details by order ID or see a list of all orders. The project comprises of a user registration, product selection, order management, and security system. It uses simple tools like PHP, MySQL, JavaScript, React, and Bootstrap to create an easy-to-use and secure system.

Overview of Components

Front-End (User Interface): These are the web pages users see and interact with, like the home page, login page, and order page. They are styled with Bootstrap to look good on phones and computers.
Back-End (Server Logic): This is the behind-the-scenes code written in PHP to handle tasks like checking user logins, saving orders, and fetching product details from the database.
Database (Data Storage): A MySQL database called x9h24 stores information about customers, products, and orders.

Web Pages

index.php: The home page where anyone can browse products by category Vegetables and Meat.
orders.php: A page for logged-in customers to place orders.
login.php: A login page with a custom CAPTCHA to prevent bots from accessing the website.
register.php: A registration page built with React, also using CAPTCHA.
manager_orders.php: A page for managers to view order details or list all orders.
logout.php: A page to log users out.

Server Scripts:
login_process.php: Checks user login details and starts a session.
register_process.php: Saves new user details to the database.
submit_order.php: Saves orders placed by customers.
get_products.php: Gets a list of products for a chosen category.
get_product_details.php: Shows details (name, price, image) for a selected product.

Database Tables (in x9h24):

customers: Stores user info like name, phone, email, password, and role (customer or manager).
products: Stores product info like name, category, price, description, and image file name.
orders: Stores order info like customer email, product ID, and order date.

Other Files:

styles.css: Styles the website to make it look consistent and user-friendly.
images/captcha/: Holds CAPTCHA images for login and registration security.
images/produts: Holds product images like carrot.jpg shown on the website.

Website Architecture

User Interface Layer: This is what users see in their browser. Pages like index.php and orders.php were built with HTML, styled with Bootstrap for a clean look, and they use

JavaScript to make things interactive like loading products without refreshing the page. The register.php page uses React for smooth form handling.

Server Logic Layer: This runs on the university server. The PHP scripts handle tasks like checking login details, saving orders, and sending product data to the browser. The scripts talk to the database to store or fetch information.

Database Layer: The MySQL database x9h24 stores all the data. The customers table holds user details, the products table holds product details (including image paths), and orders holds order details. The tables are linked.

Implementation Details

Front-End:
Pages are styled with Bootstrap to look good on all devices. JavaScript uses fetch to talk to the server without reloading pages, making the site feel fast. register.php uses React to check form inputs like valid email before sending them to the server.
Product images are shown on index.php, orders.php, and manager_orders.php to make products more appealing.

Back-End:
PHP scripts use secure methods (prepared statements) to avoid hacking attempts like SQL injection.
User sessions track who is logged in. The role field in customers (either "customer" or "manager") controls access to pages like manager_orders.php.
The API (api_orders.php) lets managers view orders by ID (GET /api/orders/1) or see all orders (GET /api/orders). It returns data in JSON format.

Security:
A custom CAPTCHA (images like captcha1.jpg with codes like "x9h24") stops bots from logging in or registering.
Passwords are stored securely using a hashing function.
Inputs are checked to prevent attacks like cross-site scripting (XSS).

Database:

The orders table links each order to a customer (via email) and a product (via product_id).
Product images are stored as file names in the products table, with actual files in images/products.

Styling:
styles.css makes sure all pages look the same, with consistent fonts, colors, and layouts.
Images (CAPTCHA and products) are styled with borders and proper sizes for clarity.

Working Procedure

1. **Start at the Home Page (index.php)**

   Anyone can visit index.php. They pick a category between Vegetables and Meat from a dropdown menu. The page then shows a list of products like Carrot in another dropdown. Choosing a product shows its details, like name, price, description, and a picture of the product.
   Customers browse products. If they click "Place Order" without logging in, they're sent to login.php. If logged in, they go to orders.php.

2. **Register (register.php)**

   New users fill out a form with their name, phone, email, password, and a CAPTCHA code. The React form checks if the email is valid and the password is strong. If everything is correct, register_process.php saves the user to the customers table and sends them to orders.php. If there's an error, they see a message and get a new CAPTCHA.

3. **Log In (login.php)**

   Users enter their email, password, and CAPTCHA code. JavaScript checks the inputs, then login_process.php verifies them against the customers' table. If correct, it sets a session and sends customers to orders.php or managers to manager_orders.php. If wrong, it shows an error and refreshes the CAPTCHA.

4. **Place Orders (orders.php)**

   Only logged-in users can access this page. They pick a category and product, see details (name, price, description, picture), and click "Place Order." The submit_order.php script saves the order to the orders table with the user's email and product ID. A success message appears, or an error if something goes wrong.

5. **Log Out (logout.php)**

   Clicking "Logout" clears the session and sends the user to login.php.

For Managers

Log In (login.php)

Managers log in with their email (manager@yahoo.com) and password (faith1), plus a CAPTCHA code. The login_process.php script checks their role (manager) and sends them to manager_orders.php.

Things that could be improved.

The website could support multiple items in one order. Right now, each order can only have one product. Customers must place separate orders for multiple items. It could change the database to have an orders table for order details (like email and date) and an order_items table for products in each order with product ID and quantity. Update orders.php to let users add multiple products to a "cart" before checking out. Update submit_order.php to save all items in one order. This would make shopping easier and more like real online stores. Customers can buy everything at once, improving the user experience.

The website could also have pages and support searching for orders. If there are many orders, the orders table in manager_orders.php could get too long and slow to load. Adding pagination to get orders so it only shows 10-20 orders at a time, with "Next" and "Previous" buttons makes it easier. Also, adding search options by customer email or date to filter orders. Update manager_orders.php with a search form and pagination links. Managers can find orders faster and the page loads quicker, especially with lots of orders.

I would improve security with better CAPTCHA and login. The custom CAPTCHA is simple and might not stop advanced bots. The session-based login works but isn't ideal for the API in a real-world app. Using Google reCAPTCHA for stronger bot protection without annoying users. This would make the website safer from hackers and bots, making it more reliable.

Showing order history for customers would also be a significant improvement on the website. Customers can't see their past orders, which is a common feature in online stores. This could add an "Order History" section to orders.php that calls a new API endpoint to show all orders for the logged-in user. Display them in a table with order ID, date, and products. Customers would feel more in control and likely use the website more.

This project could add order status tracking. There's no way to track if an order is "Pending," "Shipped," or "Delivered," which customers and managers expect. One could add a status column to the orders table. It could update manager_orders.php to let managers change the status via a new API endpoint, to show the status in orders.php for customers. Customers know when their order will arrive, and managers can manage orders better.

Conclusion

The online grocery store website is a working system that lets customers browse products, sign up, log in, and order items securely. Managers can check order details by ID or see all orders in a table, with product images and customer info. The website uses a clear structure with a user-friendly front-end, a secure back-end, and a database to store data. Features like product images, a custom CAPTCHA, and a web service for managers make it complete. However, there are some limits, like only allowing one item per order and a basic

CAPTCHA. The improvements like multi-item orders, better security, and order history would make the website more like a real online store.

REFERENCES

GetBootstrap.com. (2023) Bootstrap v5.3 Documentation. [online] Available at: https://getbootstrap.com/docs/5.3/getting-started/introduction/ [Accessed 12 April 2025].

W3C (World Wide Web Consortium). (2015) CSS Snapshot 2015. [online] Available at: https://www.w3.org/TR/CSS/ [Accessed 12 May 2025].