

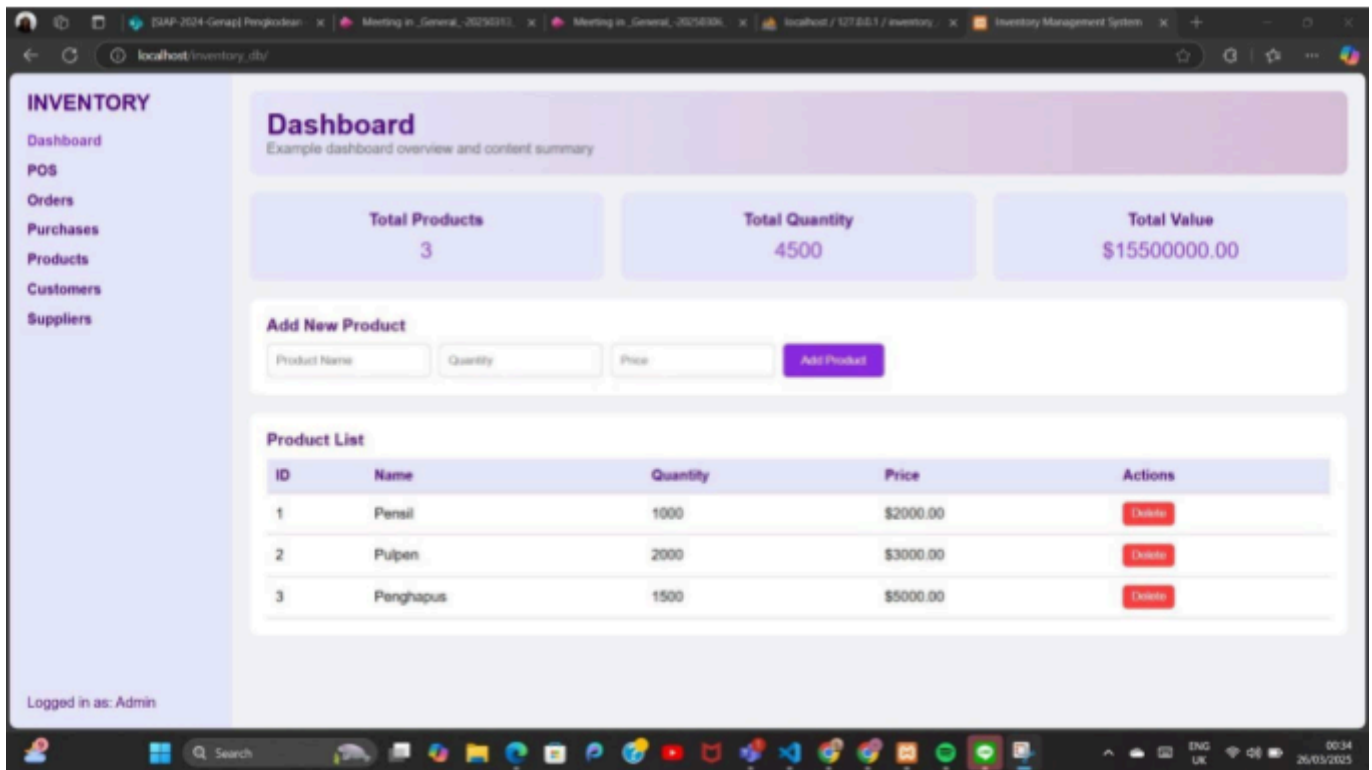
Nama : Adli Norzian Syahbudi

NIM : 12030123140334

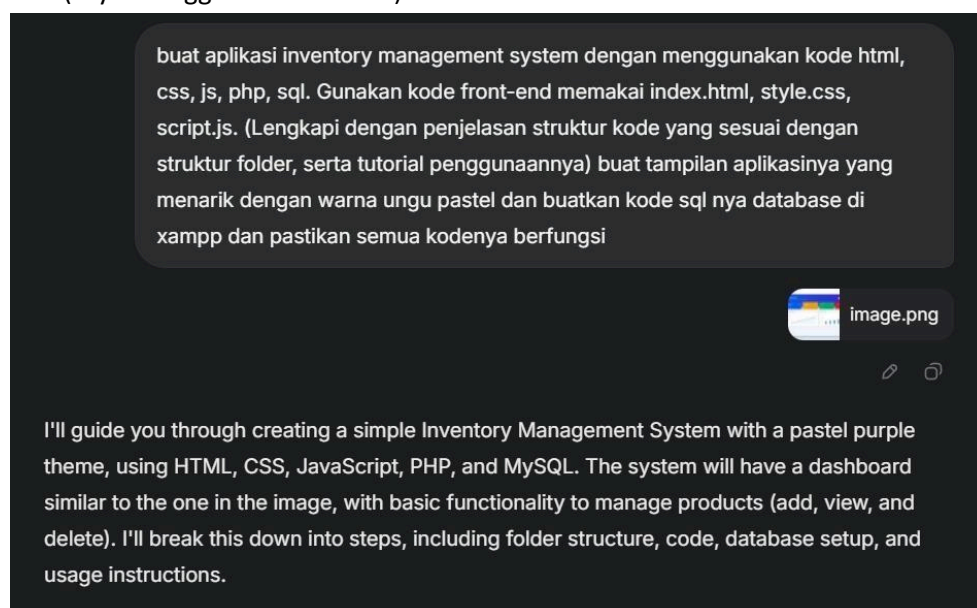
Mata Kuliah : Pengkodean dan Pemrograman (F)

Tugas 5 – Membuat Web Inventory Management System Tampilan

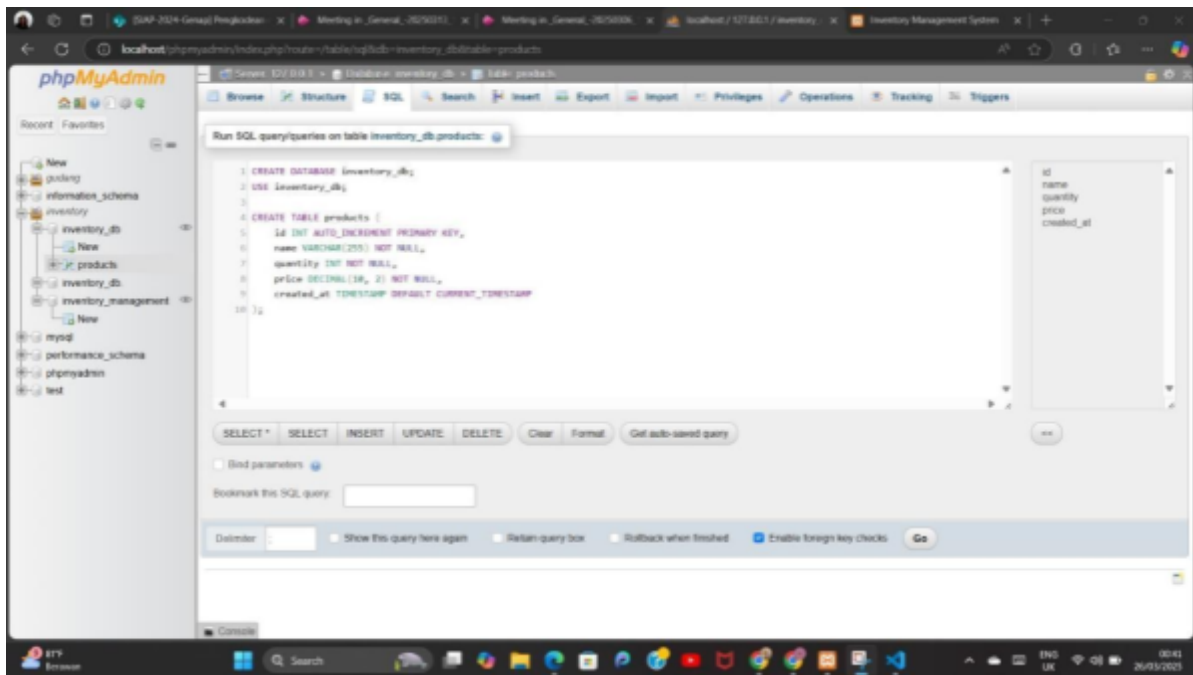
Web Inventory Management System :



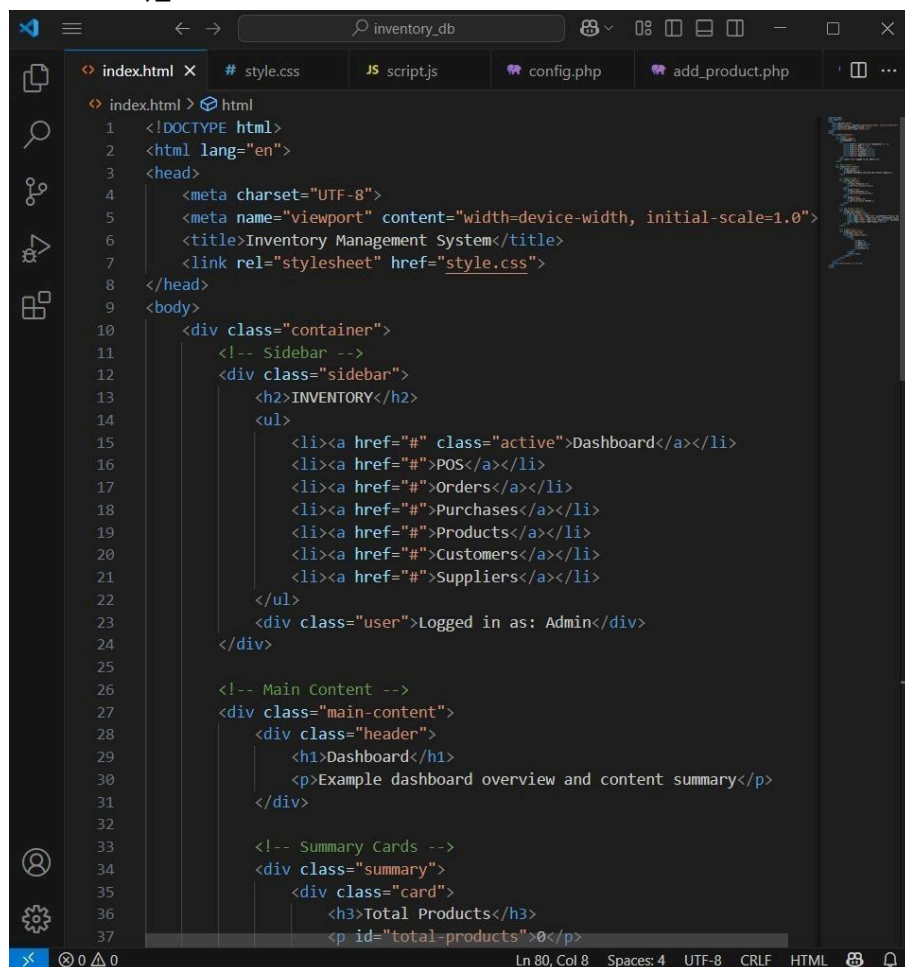
Step 1 : Buka AI lalu masukkan gambar yg sudah diambil dari Github, tuliskan perintah seperti yg saya buat dibawah (saya menggunakan Grok AI).



Step 2 : Setelah AI merespon, akan diberikan structure folder beserta kode-kode yg dibutuhkan untuk membuat Inventory Management System. Lalu buat database di phpMyAdmin dengan kode sql yg sudah diberikan.

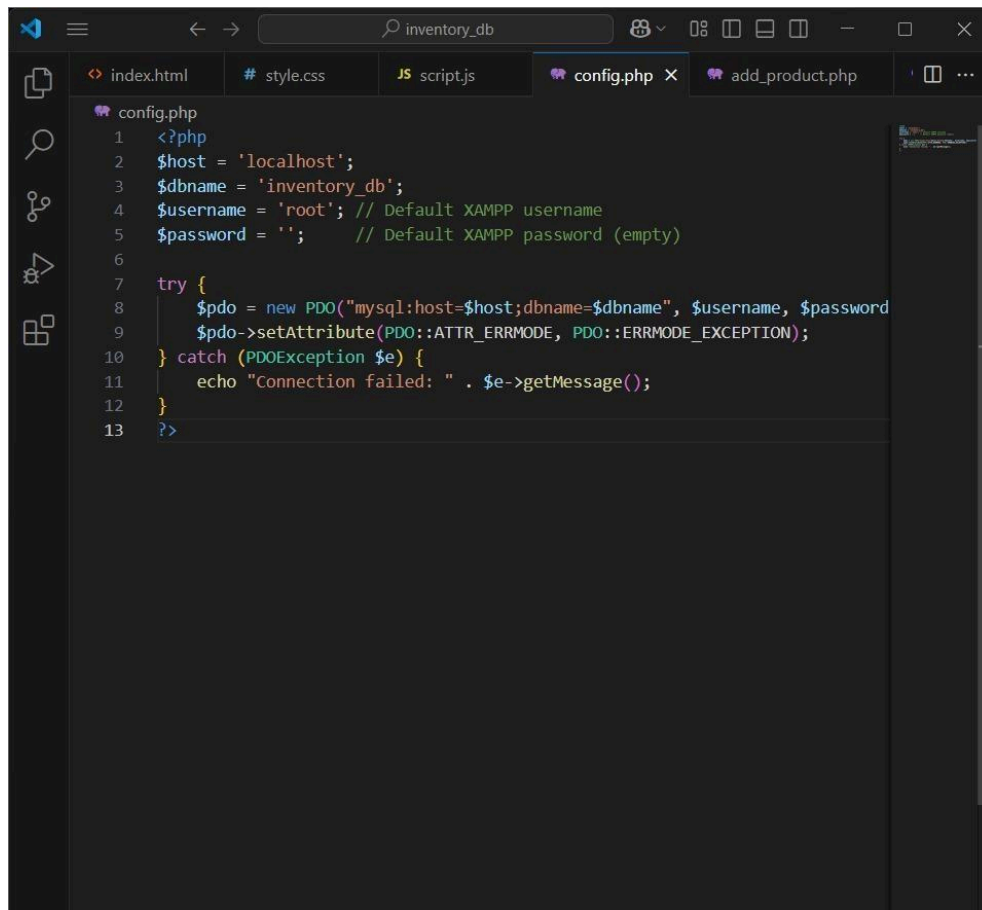


Step 3 : Setelah itu, masukkan kode-kode lainnya (html, style.css, javascript, dll) ke dalam Visual Code Studio, jangan lupa untuk save folder ke dalam Data C lalu klik xampp, klik httdocs, dan buat folder baru bernama “inventory_db”.



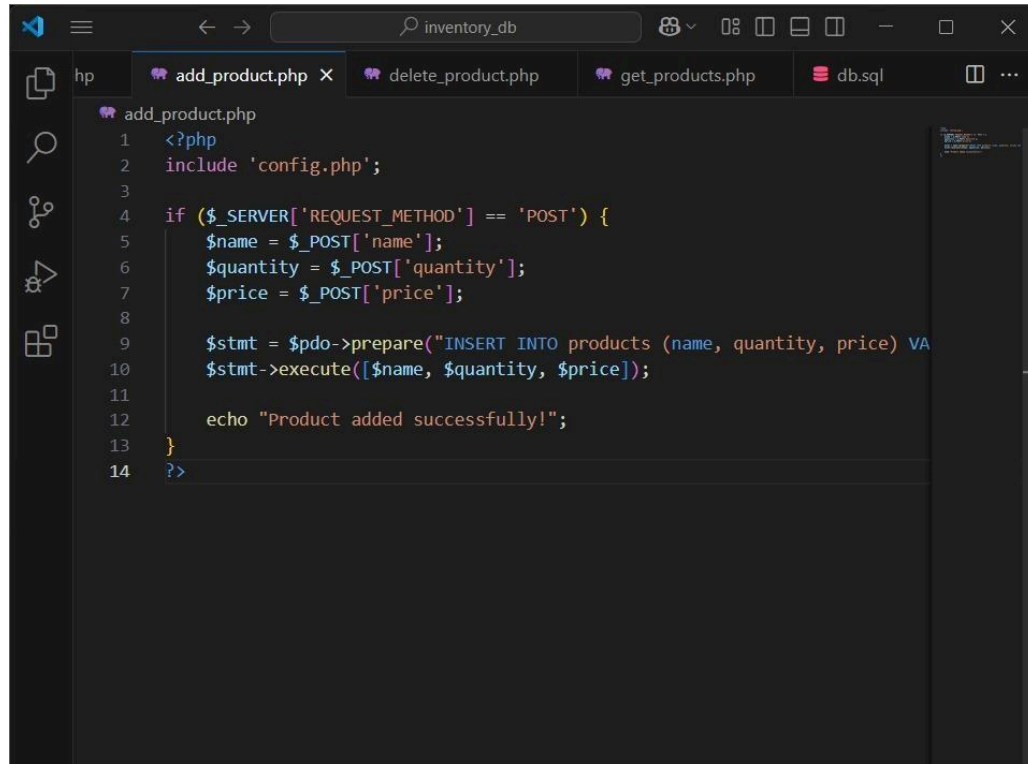
```
index.html # style.css x JS script.js config.php add_product.php
# style.css > table td button: hover
1 {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   font-family: Arial, sans-serif;
6 }
7
8 body {
9   background-color: #f0f2f5;
10 }
11
12 .container {
13   display: flex;
14 }
15
16 .sidebar {
17   width: 250px;
18   background-color: #e6e6fa; /* Pastel purple */
19   padding: 20px;
20   height: 100vh;
21   position: fixed;
22 }
23
24 .sidebar h2 {
25   color: #4b0082; /* Darker purple for contrast */
26   margin-bottom: 20px;
27 }
28
29 .sidebar ul {
30   list-style: none;
31 }
32
33 .sidebar ul li {
34   margin: 15px 0;
35 }
36
37 .sidebar ul li a {
```

```
index.html # style.css x JS script.js config.php add_product.php
JS script.js > deleteProduct
1 document.addEventListener('DOMContentLoaded', () => {
2   // Fetch and display products on page load
3   fetchProducts();
4
5   // Handle form submission to add a product
6   document.getElementById('add-product-form').addEventListener('submit',
7     e.preventDefault();
8     const formData = new FormData(e.target);
9
10    fetch('add_product.php', {
11      method: 'POST',
12      body: formData
13    })
14    .then(response => response.text())
15    .then(data => {
16      alert(data);
17      fetchProducts(); // Refresh the product list
18      e.target.reset(); // Reset the form
19    })
20    .catch(error => console.error('Error:', error));
21  });
22 });
23
24 function fetchProducts() {
25   fetch('get_products.php')
26   .then(response => response.json())
27   .then(products => {
28     // Update summary
29     const totalProducts = products.length;
30     const totalQuantity = products.reduce((sum, product) => sum +
31     const totalValue = products.reduce((sum, product) => sum + (pr
32
33     document.getElementById('total-products').textContent = totalP
34     document.getElementById('total-quantity').textContent = totalQ
35     document.getElementById('total-value').textContent = `$$${total
36
37     // Update product table
```



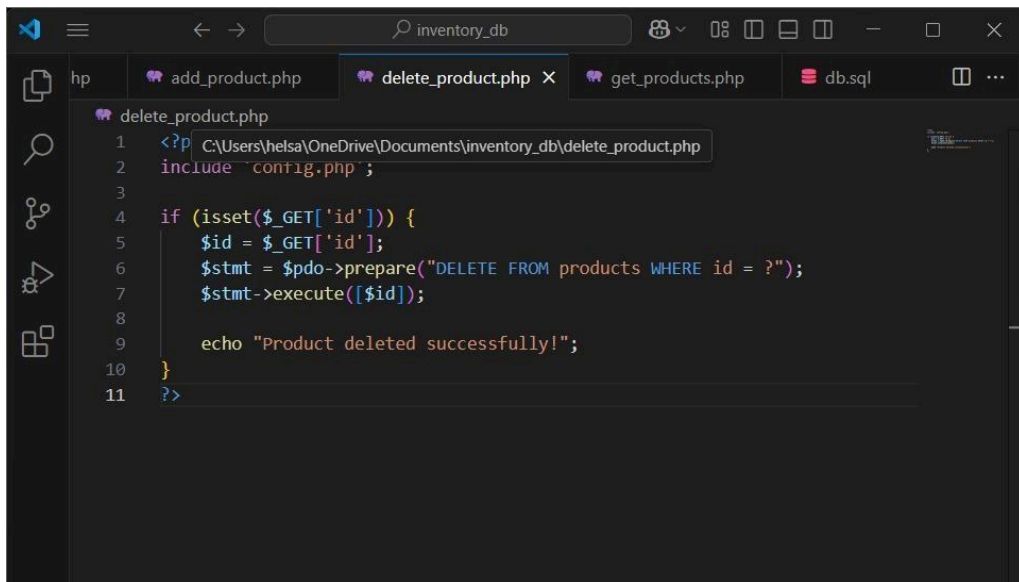
The screenshot shows the Visual Studio Code editor with a dark theme. The top toolbar includes icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The Explorer sidebar on the left shows a file tree with 'index.html', 'style.css', 'script.js', 'config.php', and 'add_product.php'. The 'config.php' file is open in the main editor. The code defines database connection variables and a PDO connection attempt.

```
1 <?php
2 $host = 'localhost';
3 $dbname = 'inventory_db';
4 $username = 'root'; // Default XAMPP username
5 $password = ''; // Default XAMPP password (empty)
6
7 try {
8     $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
9     $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
10 } catch (PDOException $e) {
11     echo "Connection failed: " . $e->getMessage();
12 }
13 ?>
```



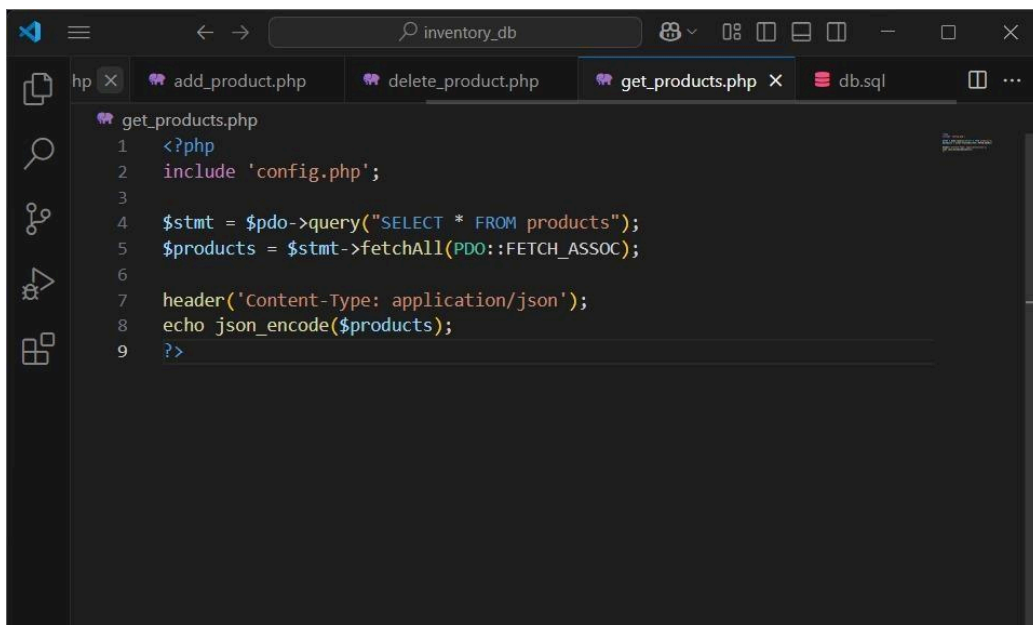
The screenshot shows the Visual Studio Code editor with a dark theme. The top toolbar is the same as the first image. The Explorer sidebar shows a file tree with 'hp', 'add_product.php', 'delete_product.php', 'get_products.php', and 'db.sql'. The 'add_product.php' file is open in the main editor. The code includes 'config.php' and handles a POST request to insert a product into the database.

```
1 <?php
2 include 'config.php';
3
4 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
5     $name = $_POST['name'];
6     $quantity = $_POST['quantity'];
7     $price = $_POST['price'];
8
9     $stmt = $pdo->prepare("INSERT INTO products (name, quantity, price) VA
10 $stmt->execute([$name, $quantity, $price]);
11
12     echo "Product added successfully!";
13 }
14 ?>
```



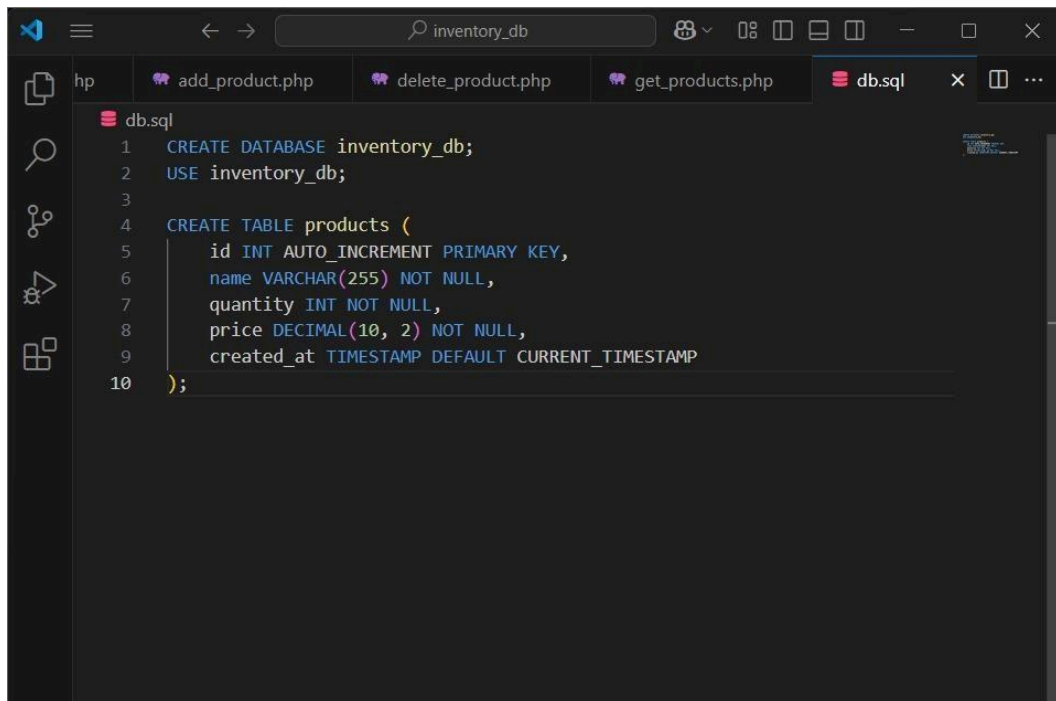
The screenshot shows the Visual Studio Code editor with the file `delete_product.php` open. The editor has a dark theme and a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The top toolbar includes icons for Explorer, Search, Source Control, Run and Debug, Extensions, and a file explorer. The file explorer shows a project named `inventory_db` with files `add_product.php`, `delete_product.php`, `get_products.php`, and `db.sql`. The `delete_product.php` file is selected and its content is displayed in the editor. The code is as follows:

```
1 <?php C:\Users\helsa\OneDrive\Documents\inventory_db\delete_product.php
2 include 'config.php';
3
4 if (isset($_GET['id'])) {
5     $id = $_GET['id'];
6     $stmt = $pdo->prepare("DELETE FROM products WHERE id = ?");
7     $stmt->execute([$id]);
8
9     echo "Product deleted successfully!";
10 }
11 ?>
```



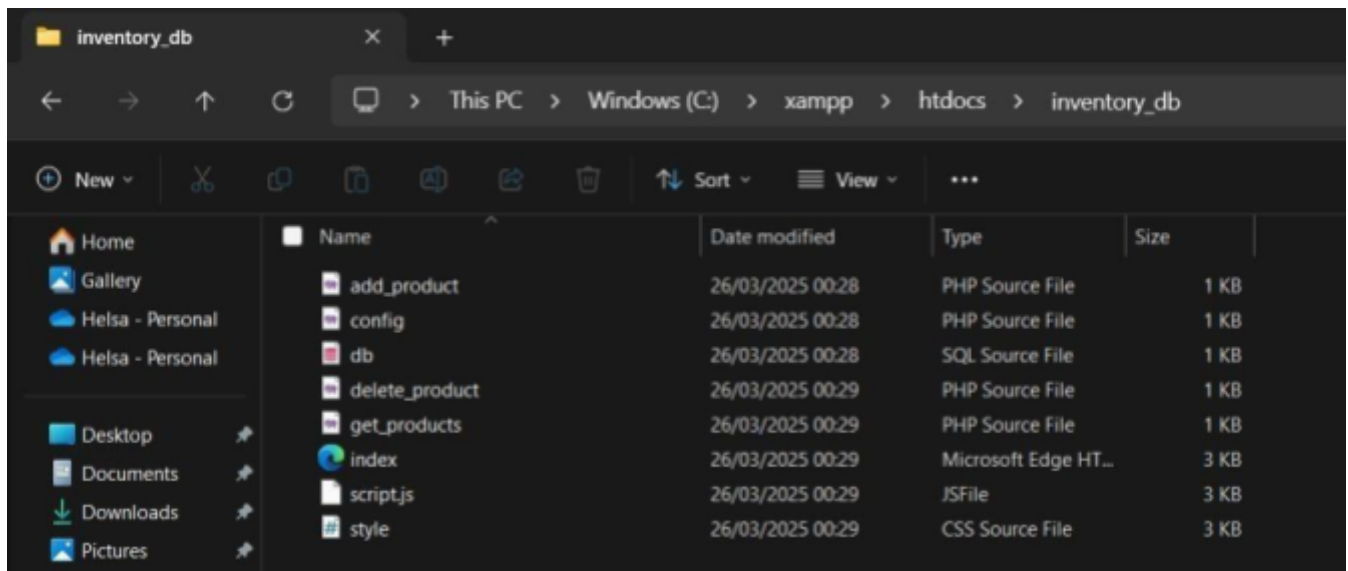
The screenshot shows the Visual Studio Code editor with the file `get_products.php` open. The editor has a dark theme and a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The top toolbar includes icons for Explorer, Search, Source Control, Run and Debug, Extensions, and a file explorer. The file explorer shows a project named `inventory_db` with files `add_product.php`, `delete_product.php`, `get_products.php`, and `db.sql`. The `get_products.php` file is selected and its content is displayed in the editor. The code is as follows:

```
1 <?php
2 include 'config.php';
3
4 $stmt = $pdo->query("SELECT * FROM products");
5 $products = $stmt->fetchAll(PDO::FETCH_ASSOC);
6
7 header('Content-Type: application/json');
8 echo json_encode($products);
9 ?>
```



The screenshot shows a code editor with a dark theme. The top bar has a search icon and the text 'inventory_db'. Below the top bar, there are tabs for 'hp', 'add_product.php', 'delete_product.php', 'get_products.php', and 'db.sql'. The 'db.sql' tab is active, showing the following SQL code:

```
1 CREATE DATABASE inventory_db;
2 USE inventory_db;
3
4 CREATE TABLE products (
5     id INT AUTO_INCREMENT PRIMARY KEY,
6     name VARCHAR(255) NOT NULL,
7     quantity INT NOT NULL,
8     price DECIMAL(10, 2) NOT NULL,
9     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
10 );
```



Step 4 : Buka web lalu cari *localhost/inventory_db/* setelah itu akan langsung muncul tampilan web Inventory Management System yg sudah dibuat, silakan dicoba dan pastikan datanya bertambah juga di phpMyAdmin.

