Excellent. Let's document **Practical 11** for your project file.

This practical is focused on demonstrating the core database operations: **Create, Read, Update, and Delete (CRUD)**. Your project has strong examples for C, R, and U. The "Delete" operation is typically an admin function, which we'll address as part of the documentation.

Here is a detailed guide on what to write in your practical file for Practical 11.

Practical 11: Store and Retrieve Data from MySQL DB

Aim: To implement and demonstrate the storage (Create, Update) and retrieval (Read) of data from the MySQL database for the main entities of the ByteBazaar application, such as users and products.

1. Key Questions & Analysis

1.1. Are SQL queries, correct?

Yes. The SQL queries used in the project are syntactically correct and follow best practices for security and efficiency.

- **Data Retrieval (SELECT):** The api/get_products.php script uses a SELECT * FROM products query to fetch product data. The api/get_user_orders.php script uses a JOIN clause to correctly retrieve order details by linking the orders, products, and users tables based on foreign keys.
- **Data Creation (INSERT):** The api/register.php script uses a prepared INSERT INTO users (...) VALUES (...) statement to create new user records.
- **Data Modification (UPDATE):** The api/update_profile.php script uses a prepared UPDATE users SET ... WHERE u_id = ? statement to modify existing user records.

1.2. Are insert, select, and delete working?

The INSERT (Create), SELECT (Read), and UPDATE operations are fully functional.

- **INSERT (Create):** This is demonstrated by the user registration process. When a user fills out the form on register.html and submits it, the api/register.php script successfully inserts a new row into the users table.
- **SELECT (Read):** This is the most widely used operation in the project. It's used to dynamically load products on the homepage and products page (api/get_products.php),

fetch details for a single product (api/get_product.php), and retrieve a user's order history for their profile page (api/get_user_orders.php).
- **UPDATE (Not explicitly required, but implemented):** The profile update functionality (api/update_profile.php) allows users to change their personal information, demonstrating a working UPDATE query.
- **DELETE:** A user-facing DELETE operation is not implemented, as customers typically cannot delete their own accounts or products. This functionality would be part of an admin dashboard (as covered in Practicals 12 and 14).

1.3. Is the DB schema normalized?

Yes. The database schema is well-normalized, which reduces data redundancy and improves data integrity.

- **Separate Tables:** Data is logically separated into distinct tables: users, products, categories, and orders.
- **Foreign Keys:** Relationships between tables are correctly established using foreign keys. For example, the orders table contains a p_id to reference the products table, a seller_id to reference the users table, and a buyer_id to also reference the users table. This ensures that an order cannot exist without a valid product and valid users.

2. Add Search by Name (Supplementary Problem)

The supplementary problem requires adding a search functionality. This is implemented through the search bar in the navigation. The backend logic for this would be an extension of the existing get_products.php script.

Example Implementation:

A search term would be passed as a GET parameter (e.g., api/get_products.php?search=headphones). The PHP script would then use a WHERE clause with the LIKE operator to filter the results.

PHP

```
// Example modification for get_products.php
$search_term = $_GET['search'] ?? '';
```

```php
$sql = "SELECT * FROM products";
if (!empty($search_term)) {
    // Use a WHERE clause to filter by title
    $sql .= " WHERE p_title LIKE ?";
}
$sql .= " ORDER BY p_createdat DESC";

$stmt = $conn->prepare($sql);

if (!empty($search_term)) {
    $search_param = "%" . $search_term . "%";
    $stmt->bind_param("s", $search_param);
}

$stmt->execute();
$result = $stmt->get_result();
// ... rest of the code to fetch and return products
```

3. SQL Dump + UI Demo (Post-Laboratory Work)

To demonstrate the working system, provide screenshots that show data flowing from the database to the UI.

Test Case 1: Database Schema
[Insert the screenshot of your database_structure.png file, which shows the tables and their columns.]
Test Case 2: Data Retrieval in UI
[Insert the screenshot practical 2/Outputs/featured_products.png, which shows the products page successfully populated with data from the products table.]
Test Case 3: Data Insertion and Retrieval

- **Step 1:** [Insert the screenshot practical 2/Outputs/register.png showing the registration form.]
- **Step 2:** After submitting, show a screenshot from phpMyAdmin of the users table with the new user's row highlighted.
- **Step 3:** [Insert a screenshot of the profile.html page (like practical 1/profile.png), showing that the user is now logged in and their data is displayed.]

This documentation provides clear evidence of your ability to connect to a MySQL database and perform the fundamental CRUD operations required by Practical 11.