Of course. Let's document **Practical 8**. This practical is about connecting your application to the MySQL database and performing basic CRUD (Create, Read, Update, Delete) operations. You have already implemented the key parts of this in your project.

Here is a detailed guide on what to write in your practical file for Practical 8.

Practical 8: Connect with MySQL to Store and Retrieve Data

Aim: To securely connect the ByteBazaar web application to a MySQL database, enabling the storage (INSERT, UPDATE) and retrieval (SELECT) of user and product data using PHP.

1. Key Questions & Analysis

1.1. Is the connection established securely?

Yes. A secure and centralized connection to the MySQL database is established using a dedicated configuration file (db/db_config.php).

- **Centralized Credentials:** Database credentials (server, username, password, database name) are stored as constants in a single file. This prevents credential duplication and makes maintenance easy.
- **Connection Handling:** A new mysqli object is instantiated to create the connection. The script immediately checks for connection errors and terminates with a descriptive message if the connection fails, preventing further script execution and potential security risks.

Example Connection Script from db/db_config.php:

PHP

```
<?php
// From db/db_config.php
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
```

```
define('DB_NAME', 'bytebazaar_db');

$conn = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Check connection
if ($conn->connect_error) {
    die("ERROR: Could not connect. " . $conn->connect_error);
}
?>
```

1.2. Are insert, select, and update operations working?

Yes. The project successfully implements the core INSERT, SELECT, and UPDATE operations.

- **SELECT Operation:** The get_products.php script performs a SELECT query to retrieve all products from the products table and sends them to the frontend. Similarly, get_product.php retrieves a single product by its ID.
- **INSERT Operation:** The register.php script uses a prepared INSERT statement to add a new user to the users table after successful validation. The process_order.php script also uses INSERT to create new records in the orders table.
- **UPDATE Operation:** The update_profile.php script uses a prepared UPDATE statement to modify an existing user's information in the users table.

1.3. Is error handling in place?

Yes. Robust error handling is implemented in all database interaction scripts.

- **Connection Errors:** As mentioned, db_config.php checks for and handles initial connection failures.
- **Query Failures:** Each PHP script that interacts with the database (e.g., get_products.php, register.php) wraps its database logic in a try...catch block. If any part of the query preparation or execution fails, an exception is thrown, caught, and a clean JSON error message is sent to the frontend. This prevents broken PHP errors from being displayed to the user.
- **Transaction Rollback:** In process_order.php, database operations are wrapped in a transaction ($conn->begin_transaction()). If any of the multiple INSERT or UPDATE queries fail, $conn->rollback() is called to undo all changes, ensuring the database remains in a consistent state.

2. Database Dump Submission (Post-Laboratory Work)

For this section, you need to provide the SQL code that defines your database schema. You

can get this by going to phpMyAdmin, selecting your bytebazaar_db database, and clicking the "Export" tab.

SQL Schema for users, products, categories, and orders tables:
(Copy the content of your user_db.sql, products_db.sql, categories_db.sql, and orders_db.sql files here.)

SQL

```sql
-- From db/user_db.sql
CREATE TABLE `users` (
  `u_id` INT AUTO_INCREMENT PRIMARY KEY,
  `first_name` VARCHAR(50) NOT NULL,
  /* ...rest of the user table schema... */
);

-- From db/products_db.sql
CREATE TABLE `products` (
  `p_id` INT AUTO_INCREMENT PRIMARY KEY,
  `p_title` VARCHAR(255) NOT NULL,
  /* ...rest of the products table schema... */
);

-- From db/categories_db.sql
CREATE TABLE `categories` (
  `c_id` INT AUTO_INCREMENT PRIMARY KEY,
  `name` VARCHAR(100) NOT NULL UNIQUE,
  /* ...rest of the categories table schema... */
);

-- From db/orders_db.sql
CREATE TABLE `orders` (
  `o_id` INT AUTO_INCREMENT PRIMARY KEY,
  `o_createdat` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  /* ...rest of the orders table schema... */
);
```

3. Result Output Testing (Viva)

To demonstrate the working system, provide screenshots that show the results of your database operations.

Test Case 1: SELECT Operation Result
[Insert a screenshot of your products.html page, successfully displaying the grid of products fetched from the database.]
Test Case 2: INSERT Operation Result
[Insert a screenshot from phpMyAdmin showing a new user in the users table after they have successfully registered through the website.]
Test Case 3: UPDATE Operation Result
[Insert a screenshot of the profile.html page showing updated user information (e.g., a new first name). Then, provide a second screenshot from phpMyAdmin showing that the corresponding row in the users table has also been updated.]
This detailed report will effectively document your successful integration of a MySQL database into your PHP application, covering all the key requirements of Practical 8.