Of course. Let's document **Practical 14**. This practical focuses on the Admin Dashboard and the management of users. Given that you've already implemented a full Admin Dashboard, this practical will be about detailing its user management capabilities.

Here is a detailed guide on what to write in your practical file for Practical 14.

Practical 14: Develop an Admin Dashboard to View/Manage Users

Aim: To develop a secure administrator dashboard for the ByteBazaar platform, enabling authorized users (admins) to view, manage, and modify user accounts and roles.

1. Key Questions & Analysis

1.1. Is the dashboard restricted to admins?

Yes. Access to the Admin Dashboard (admin-dashboard.html and related admin-*.html pages) is strictly restricted to users with the 'admin' role.

- **Server-Side Authorization:** Each PHP API endpoint (api/admin_get_users.php, api/admin_update_user.php, etc.) that serves the admin dashboard first checks the $_SESSION['user_role'] variable. If the logged-in user's role is not 'admin', the script immediately terminates and returns an unauthorized response, preventing data access even if the frontend checks are bypassed.
- **Client-Side Redirection:** The js/admin-dashboard.js and other js/admin-*.js scripts verify the user's role upon page load (by calling api/check_login.php or a dedicated admin role check). If the user is not an admin, they are immediately redirected to the homepage or login page, providing an immediate visual access control.

1.2. Does it display all relevant user information?

Yes. The "Users" section of the Admin Dashboard (admin-users.html) provides a comprehensive view of all registered users on the platform.

- **User List:** The dashboard displays a paginated list of users, including their:
  - **ID:** Unique user identifier.
  - **Name:** First and Last name.
  - **Email:** Contact email.
  - **Role:** Current role (customer, seller, admin).

    - **Registration Date:** When the account was created.
    - **Activity Status:** (e.g., is_active - if implemented) to indicate if the account is active.
- **Data Fetching:** The js/admin-users.js script fetches this information from the api/admin_get_users.php endpoint, which queries the users table in the database.

1.3. Are there options to manage users (e.g., change role, delete)?

Yes. The Admin Dashboard provides crucial management functionalities for user accounts.

- **Role Management:** Admins can easily change a user's role (e.g., promote a 'customer' to 'seller' or vice versa). This is typically done via a dropdown or button associated with each user in the list, which triggers a call to api/admin_update_user.php.
- **Deletion:** Admins have the ability to delete user accounts. A "Delete" button for each user sends a request to api/admin_delete_user.php, which executes a prepared DELETE statement on the users table. This often includes a confirmation step to prevent accidental deletions.
- **Activation/Deactivation (Optional but good to mention if you have it):** If you've implemented an is_active field in your users table, admins can toggle a user's account status.

2. CRUD on Users (Supplementary Problem)

The Admin Dashboard inherently provides full CRUD capabilities for user management, extending beyond just viewing.

- **Create (User):** While not typically creating users directly, an admin could have an option to manually add a new user. Your register.php script serves as the 'Create User' endpoint that any user (or admin) can access.
- **Read (User):** Fully implemented via admin_get_users.php displaying the list on admin-users.html.
- **Update (User):** Implemented via admin_update_user.php to change roles, names, etc.
- **Delete (User):** Implemented via admin_delete_user.php to remove user accounts.

3. User Management Demonstration (Post-Laboratory Work)

To comprehensively demonstrate the user management capabilities of your Admin Dashboard, provide the following screenshots:

Test Case 1: Admin Dashboard - Users List

[Insert a screenshot of the admin-users.html page, showing the list of all users with their details (Name, Email, Role, etc.). Highlight the options to edit/delete/change role.]

Test Case 2: Changing a User's Role (UI)

[Insert a screenshot showing the process of changing a user's role (e.g., a dropdown being used to select 'seller' for a 'customer'). This could be before and after the change on the UI.]

Test Case 3: Changing a User's Role (Backend Verification)

[After performing the role change in the UI, insert a screenshot from phpMyAdmin, specifically showing the users table with the updated user_role for that particular user.]

Test Case 4: Deleting a User (UI)

[Insert a screenshot of the admin-users.html page, showing the confirmation prompt (if implemented) before deleting a user, or showing the user removed from the list with a "User deleted successfully!" notification.]

Test Case 5: Deleting a User (Backend Verification)

[After deleting the user in the UI, insert a screenshot from phpMyAdmin, showing that the deleted user's record is no longer present in the users table.]

This detailed documentation, supported by your code's functionality and relevant screenshots, will provide robust evidence of your successful implementation of the Admin Dashboard's user management, fulfilling all requirements for Practical 14.