Student Dashboard Application Guide

July 21, 2025

Contents

1 Introduction

This document provides a comprehensive guide to creating a student dashboard application using HTML and CSS, and deploying it in a Docker container with a MySQL database. The dashboard allows administrators to manage student records and students to view their profiles. The final output is a web application accessible via a browser, containerized with Docker, and connected to a MySQL database for persistent storage.

2 Prerequisites

Before starting, ensure you have the following installed:

- **Docker**: Docker Desktop or Docker CLI for containerization.
- **MySQL Client**: MySQL Workbench or similar for database management.
- **Web Browser**: For testing the application.
- **Text Editor**: VS Code or any editor for coding.
- **PHP Environment**: PHP installed locally or within the Docker container.

3 Application Development

3.1 Directory Structure

Create the following directory structure for the project:

```
student-dashboard/
 index.html
 styles.css
 index.php
 Dockerfile
 docker-compose.yml
 scripts/
     db_init.sql
```

3.2 HTML: Creating the Dashboard

The `index.html` file defines the structure of the dashboard, including a navigation bar, a form for adding student records, and a table to display them.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
   =1.0">
    <title>Student Dashboard</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
```

```
        <nav>
            <h1>Student Dashboard</h1>
            <ul>
                <li><a href="#home">Home</a></li>
                <li><a href="#profile">Profile</a></li>
                <li><a href="#logout">Logout</a></li>
            </ul>
        </nav>
        <div class="main-content">
            <div class="form-section">
                <h2>Add Student</h2>
                <form id="student-form" action="index.php" method="POST
    ">
                    <input type="text" name="name" placeholder="Name"
    required>
                    <input type="number" name="age" placeholder="Age"
    required>
                    <input type="text" name="grade" placeholder="Grade"
     required>
                    <button type="submit">Add Student</button>
                </form>
            </div>
            <div class="table-section">
                <h2>Student Records</h2>
                <table>
                    <thead>
                        <tr>
                            <th>Name</th>
                            <th>Age</th>
                            <th>Grade</th>
                        </tr>
                    </thead>
                    <tbody id="student-table">
                        <!-- Populated by PHP -->
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</body>
</html>
```

3.3 CSS: Styling the Dashboard

The `styles.css` file provides styling to make the dashboard visually appealing and responsive.

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}
```

```
.container {
    width: 90%;
    max-width: 1200px;
    background: white;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    border-radius: 8px;
    overflow: hidden;
}
nav {
    background: #007bff;
    color: white;
    padding: 10px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
nav h1 {
    margin: 0;
    padding-left: 20px;
}
nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex;
}
nav ul li {
    margin: 0 15px;
}
nav ul li a {
    color: white;
    text-decoration: none;
    font-size: 16px;
}
.main-content {
    display: flex;
    padding: 20px;
}
.form-section, .table-section {
    flex: 1;
    padding: 20px;
}
.form-section input {
    display: block;
    width: 80%;
    margin: 10px 0;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 4px;
}
.form-section button {
    padding: 10px 20px;
    background: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

```
.form-section button:hover {
    background: #0056b3;
}
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    border: 1px solid #ccc;
    padding: 10px;
    text-align: left;
}
th {
    background: #f2f2f2;
}
```

3.4 PHP: Connecting to MySQL

The `index.php` file handles form submissions and displays student records from the MySQL database.

```
<?php
$servername = "db";
$username = "root";
$password = "my-secret-pw";
$dbname = "student_db";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST['name'];
    $age = $_POST['age'];
    $grade = $_POST['grade'];

    $sql = "INSERT INTO students (name, age, grade) VALUES ('$name', $age, '$grade')";
    $conn->query($sql);
}

$sql = "SELECT * FROM students";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>" . $row["name"] . "</td><td>" . $row["age"] . "</td><td>" . $row["grade"] . "</td></tr>";
    }
}

$conn->close();
?>
```

3.5 Database Initialization

The `db_init.sql` script creates the database and table.

```
CREATE DATABASE IF NOT EXISTS student_db;
USE student_db;
CREATE TABLE IF NOT EXISTS students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    age INT NOT NULL,
    grade VARCHAR(50) NOT NULL
);
```

4 Deployment with Docker

4.1 Dockerfile

Create a `Dockerfile` to set up the PHP and Apache environment.

```
FROM php:7.4-apache
RUN docker-php-ext-install mysqli
COPY . /var/www/html/
EXPOSE 80
CMD ["apache2-foreground"]
```

4.2 Docker Compose Configuration

The `docker-compose.yml` file defines services for the web application and MySQL database.

```
version: '3.8'
services:
  web:
    build: .
    container_name: php_web
    ports:
      - "8080:80"
    volumes:
      - .:/var/www/html
    depends_on:
      - db
    networks:
      - app-network
  db:
    image: mysql:8.0
    container_name: mysql_db
    environment:
      MYSQL_ROOT_PASSWORD: my-secret-pw
      MYSQL_DATABASE: student_db
    volumes:
      - dbdata:/var/lib/mysql
      - ./scripts/db_init.sql:/docker-entrypoint-initdb.d/db_init.sql
    ports:
      - "3307:3306"
    networks:
      - app-network
networks:
```

```
  app-network:
    driver: bridge
volumes:
  dbdata:
```

4.3 Deployment Steps

Follow these steps to deploy the application:

1. **Create Project Directory**: Set up the directory structure as shown above and place all files (`index.html`, `styles.css`, `index.php`, `Dockerfile`, `docker-compose.yml`, and `scripts/db_init.sql`).
2. **Build and Run Containers**: Navigate to the project directory in a terminal and run:

   ```
   docker-compose up -d
   ```

   This builds the PHP web image and starts both the web and database containers.
3. **Verify Database**: Access the MySQL container to ensure the database and table are created:

   ```
   docker exec -it mysql_db mysql -u root -pmy-secret-pw
   ```

   Run `SHOW DATABASES;` and `USE student_db; SHOW TABLES;` to confirm.
4. **Access the Application**: Open a browser and navigate to `http://localhost:8080` to view the dashboard.
5. **Test Functionality**: Add student records using the form and verify they appear in the table.
6. **Stop and Clean Up**: To stop the containers:

   ```
   docker-compose down
   ```

   To remove the volume and data:

   ```
   docker-compose down -v
   ```

5 Troubleshooting

- **Connection Errors**: Ensure the MySQL container is running and the `servername` in `index.php` matches the service name (`db`) in `docker-compose.yml`.
- **Port Conflicts**: If port 8080 or 3307 is in use, modify the ports in `docker-compose.yml`.
- **Database Not Initialized**: Check that `db_init.sql` is correctly placed in the `scripts` directory.

6 Conclusion

This guide provides a complete setup for a student dashboard application using HTML, CSS, PHP, and MySQL, deployed via Docker. The application is accessible at `http://localhost:8080`, with data persisted in a MySQL database. For further customization, consider adding features like user authentication or advanced styling.