Web Development - Lab 3

2nd Year LMD - Computer Science

Objective: Convert the Step 3 implementation of Lab 2 to an MVC structure

Objective

In this lab, students will refactor their BMI calculator implementation from Lab 2 (Step 3) to follow the Model-View-Controller (MVC) pattern. They will:

- Organize their code into an MVC structure.
- Implement controllers to handle requests and validation.
- Use models to interact with the database.
- Maintain clean separation between logic and presentation.

Database Schema

The following database schema will be used to store user BMI history:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('user', 'admin') NOT NULL DEFAULT 'user'
);

CREATE TABLE bmi_records (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    weight FLOAT NOT NULL,
    height FLOAT NOT NULL,
    bmi FLOAT NOT NULL,
    status VARCHAR(50) NOT NULL,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Project Structure

Students should organize their project using the following directory structure:

```
/bmi_project/
    |-- /app/
        |-- /controllers/
            |-- BmiController.php
        |-- /models/
            |-- BmiModel.php
        |-- /views/
            |-- bmi_form.php
            |-- bmi_result.php
    |-- /public/
        |-- index.php
    |-- /config/
        |-- database.php
    |-- .htaccess
```

Tasks

1. Implement the Model (BmiModel.php)

Create a ‘BmiModel‘ class to handle database interactions:

```
class BmiModel {
    private $db;
    public function __construct($database) {
        $this->db = $database;
    }
    public function saveBmiRecord($name, $weight, $height, $bmi, $status) {
        // SQL Insert Query
    }
    public function getBmiHistory() {
        // SQL Select Query
    }
}
```

2. Implement the Controller (BmiController.php)

Create a ‘BmiController‘ class to process form submissions and return results:

```
class BmiController {
    private $model;
    public function __construct($model) {
        $this->model = $model;
    }
    public function calculateBmi($name, $weight, $height) {
        // BMI Calculation Logic
        // Save Record
        // Return Data to View
    }
}
```

3. Implement the Views (bmi_form.php and bmi_result.php)

- ‘bmi_form.php‘: HTML form for user input. - ‘bmi_result.php‘: Displays results with Bootstrap styling.

4. Create the Entry Point (index.php)

Set up routing and handle requests:

```
require '../config/database.php';
require '../app/models/BmiModel.php';
require '../app/controllers/BmiController.php';
$model = new BmiModel($db);
$controller = new BmiController($model);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $controller->calculateBmi($_POST['name'], $_POST['weight'], $_POST['
        height']);
}
require '../app/views/bmi_form.php';
```

Deliverables

Students should submit their refactored BMI calculator project with:

- Complete MVC structure.
- Functional BMI calculation and history storage.
- Bootstrap-styled user interface.

Home Assignment

Extend the project by implementing:

- User authentication (Login/Logout system).
- Role-based access control (Admin/User roles).
- Displaying user BMI history using Chart.js.