**KHULNA UNIVERSITY**



**Database Project Report**

**Course Title : Database Systems Project/Fieldwork**

**Course Code : 0714 02 CSE 2206**

**Submitted To**

**Aminul Islam  
Assistant Professor,  
Computer Science and Engineering Discipline  
Khulna University,  
Khulna-9208**

**Submitted By**

**Saiful Islam Reyad (Student ID: 230214)  
Md. Sabbir Khan (Student ID: 230204)  
Rownak Jahan (Student ID: 210214)**

**Submission Date: 12/04/25**

# **System Implementation Report**

### Project Title: CliniCode (Digital Prescription & Patient Record System for Clinics)

## **User Interface Design**

### ****Overview****

The interface supports all major functions like **patient record management**, **prescription generation**, and **appointment tracking**. It follows a consistent layout and color scheme suitable for medical software, aiming for a clean, user-friendly experience for both doctors and patients.

### Landing page (index.html)

This landing page is the **entry point** for the Prescription Generator system. It has a clean, simple layout with four main options presented as **cards**:

**Doctor's Login**

* + A login option for registered doctors.
  + Clicking the button takes the doctor to login/login.html to access their dashboard.

**Patient's Login**

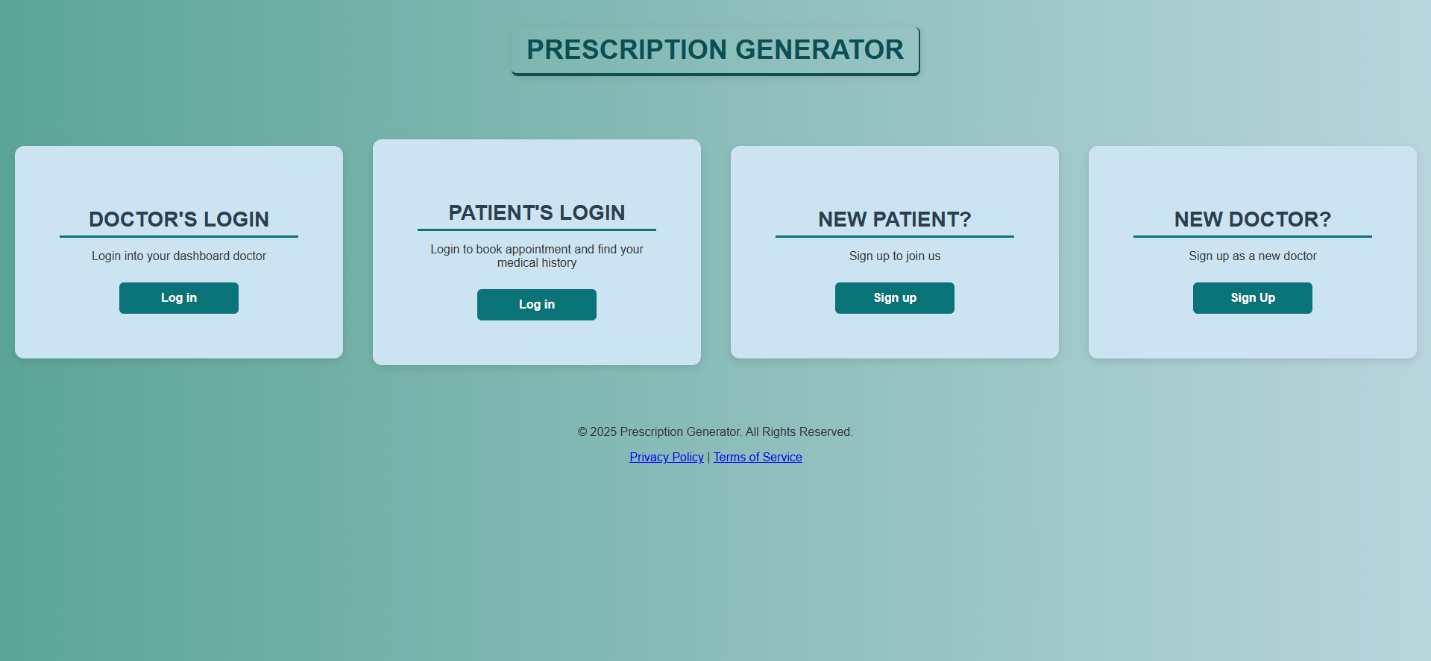
* + For existing patients to log in.
  + Redirects to patient/login.html to book appointments or check their medical history.

**New Patient? Sign Up**

* + Allows new patients to create an account.
  + Takes them to patient/signup.html to register.

**New Doctor? Sign Up**

* + Enables doctors to register and join the platform.
  + Leads to signup/doctor\_signup\_form.html.



**Functionality Overview**

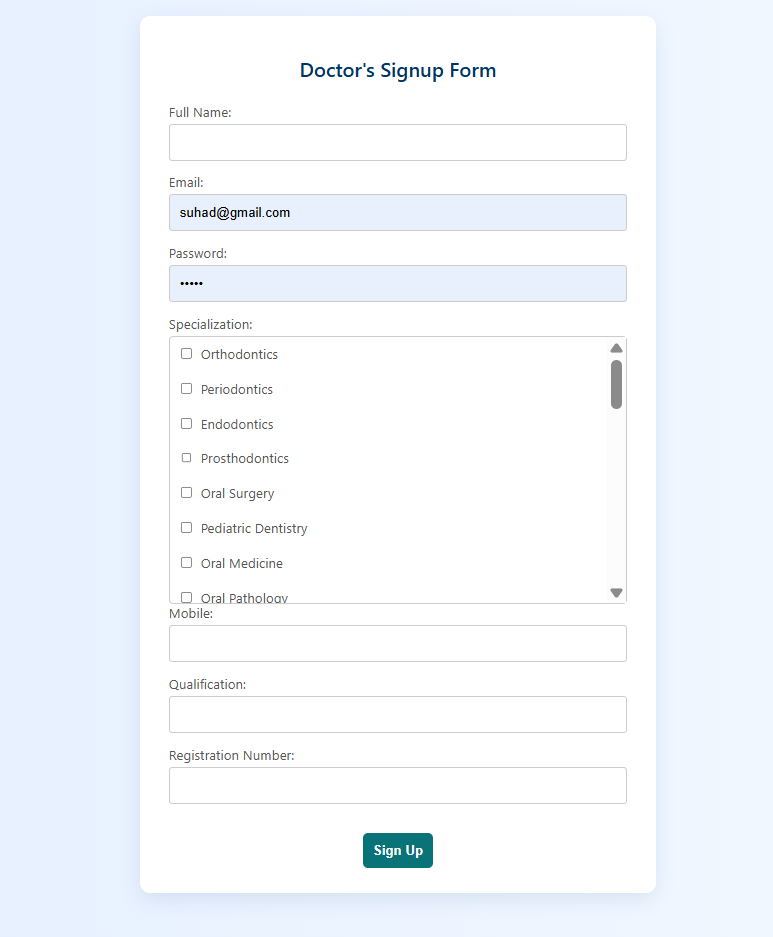
* The page does not interact with the database directly.
* It provides **navigation to login and signup pages**, where database operations (like authentication and registration) will occur.
* This UI separates user roles clearly — **Doctor** and **Patient** — which is essential for managing access and records in the system.

### Doctor Signup Page

This page allows new doctors to **register on the platform**. The layout includes a clean, labeled form that asks for:

* Full Name
* Email
* Password
* Mobile Number
* Qualification
* Registration Number
* **Specialization** (fetched dynamically)

A **"Sign Up"** button is at the bottom to submit the form.



**Functionality Overview**

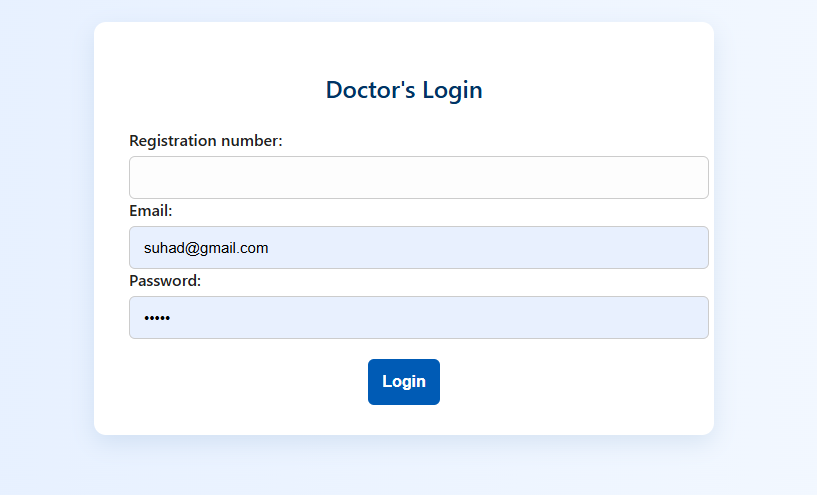
* When the page loads, it runs a script to **fetch specializations** from get\_specialization.php.
* The script displays these as **checkboxes**, allowing doctors to select one or more specializations.
* On submission, the form data is sent to doctor\_signup.php, which stores the information in the database.
* This interaction involves a **dynamic database fetch (GET)** and a **form submission (POST)** to register the doctor.

### Doctor Login Page

This page allows existing doctors to **log in to their dashboard**. It contains a clean form with the following fields:

* **Registration Number**
* **Email**
* **Password**

There is also a **Login** button to submit the form.

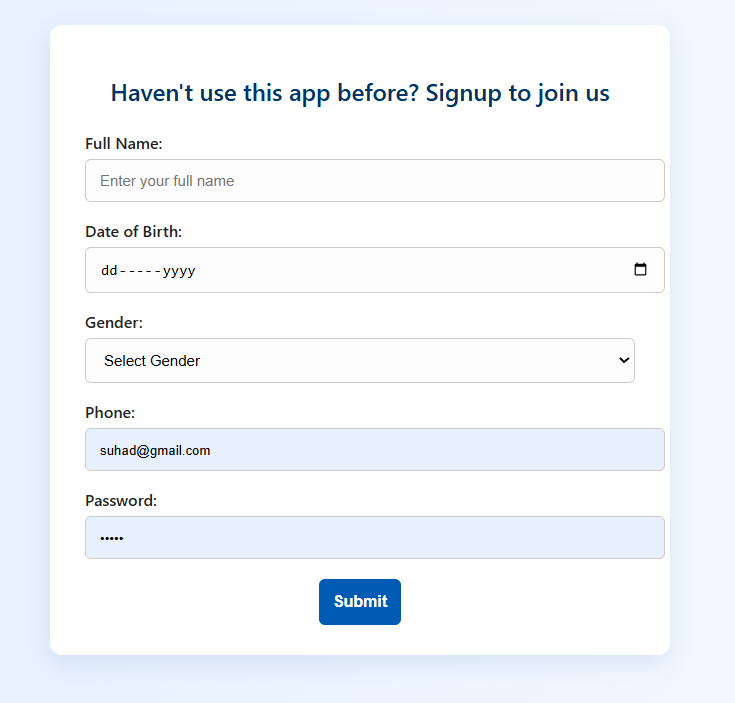
  
**Functionality Overview**

* The doctor enters their credentials (registration number, email, password).
* When submitted, the form sends the data via **POST** to doctor\_login.php.
* That script will **check the credentials in the database**, and if correct, will **grant access** to the doctor’s dashboard.

### Patient Signup Page

This page allows **new patients** to sign up. It contains a user-friendly form with the following fields:

* Full Name
* Date of Birth
* Gender (dropdown)
* Phone Number
* Password
* A **Submit** button to complete the signup



**Functionality Overview**

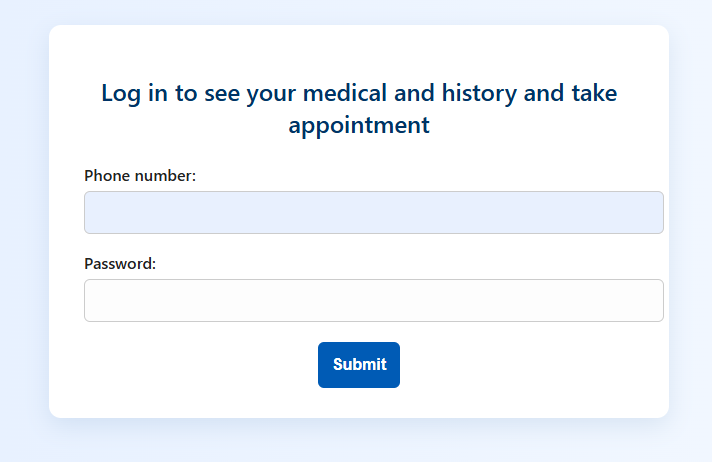
* Patients fill in their details and press the **Submit** button.
* The form sends the data to patient\_signup.php via **POST**.
* The backend script stores this data into the **database**, registering the new patient.
* It ensures that patients can later **log in and access services**, like booking appointments and viewing medical history.

### Patient Login Page

This page allows **registered patients** to log in. The interface includes:

* A **phone number** input
* A **password** input
* A **Submit** button to log in

The form is styled within a centered container, keeping it clean and easy to use.



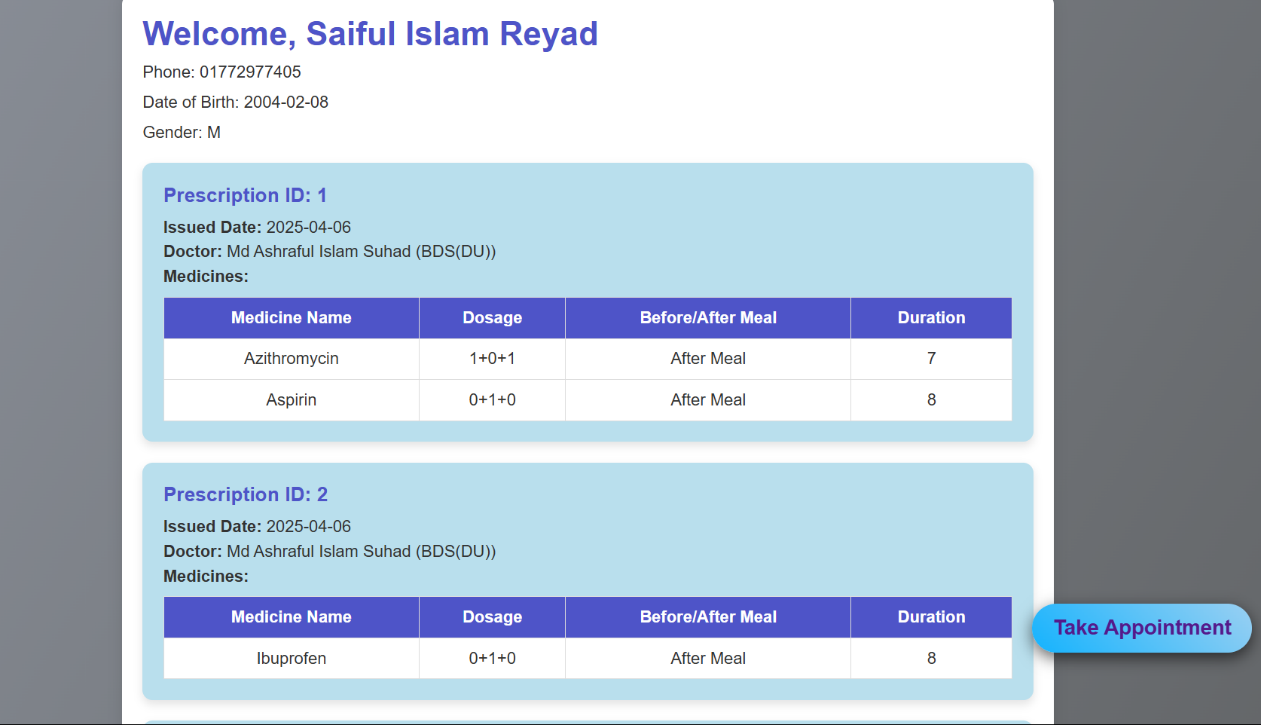
**Functionality Overview**

* The patient enters their **phone number** and **password**.
* On submission, the form sends the data to login.php using the **POST** method.
* The backend script verifies the credentials with the **database**.
* If valid, the patient is redirected to their dashboard where they can:
  + View **medical history**
  + **Book appointments** with doctors

### Patient Dashboard Page

This **Patient Dashboard** page is the core page where registered patients can:

1. **View their personal details**:
   * Full name
   * Phone number
   * Date of birth
   * Gender
2. **See their previous prescriptions**:
   * Prescription ID
   * Issued date
   * Doctor's name and qualification
   * Medicines prescribed, including:
     + Medicine name
     + Dosage
     + Before/After meal instructions
     + Duration
3. **Check past appointments**:
   * Appointment number
   * Problem description
   * Date and time of appointment
   * Appointment status
   * Doctor's name and qualifications
4. **Request a new appointment**:
   * Link to a form for scheduling new appointments.



…………



**Functionality Overview**

* The page starts by checking if the patient is logged in through **session data** ($\_SESSION['patient\_id']).
* If the patient is logged in:
  + **Personal information** is displayed, which is fetched from the session data.
  + **Prescriptions**:
    - All prescriptions for the logged-in patient are fetched from the database by joining the prescriptions, appointments, and doctors tables.
    - A table is generated to show the prescribed medicines, including dosage, before/after meal instructions, and duration.
  + **Appointments**:
    - All appointments for the patient are fetched from the appointments table, displaying details like appointment number, status, and doctor information.
  + **New Appointment**:
    - A link is provided to the appointment\_form.php, allowing the patient to book a new appointment.
* If the patient is **not logged in**, the page redirects them to the **login page** (../index.html).

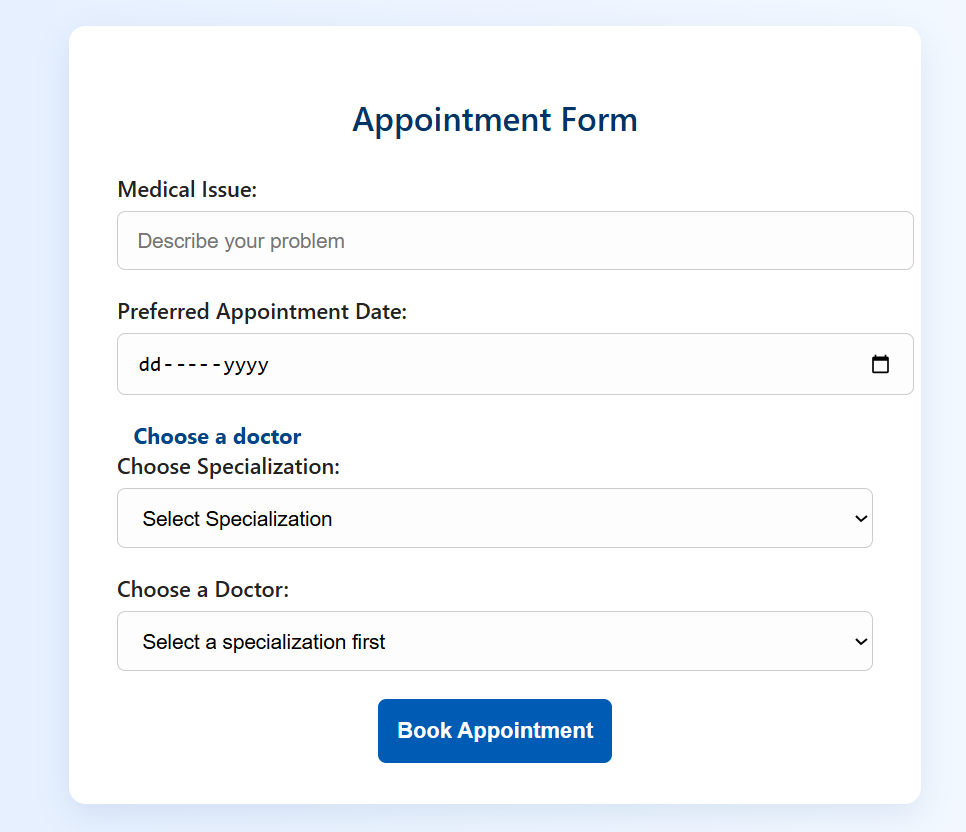
**Important Considerations:**

* **Security**:
  + Ensures only logged-in patients can access the dashboard.
  + All sensitive data is sanitized using htmlspecialchars() to prevent XSS (Cross-Site Scripting) attacks.
* **Error Handling**:
  + If no prescriptions or appointments exist, the page displays relevant messages: "No previous prescriptions found" or "No appointments found."

### Appointment Form Page

This **Appointment Form** page allows patients to:

1. **Fill in their medical issue**.
2. **Choose a preferred appointment date**.
3. **Select a specialization** from a list of available specializations.
4. **Select a doctor** based on the chosen specialization.

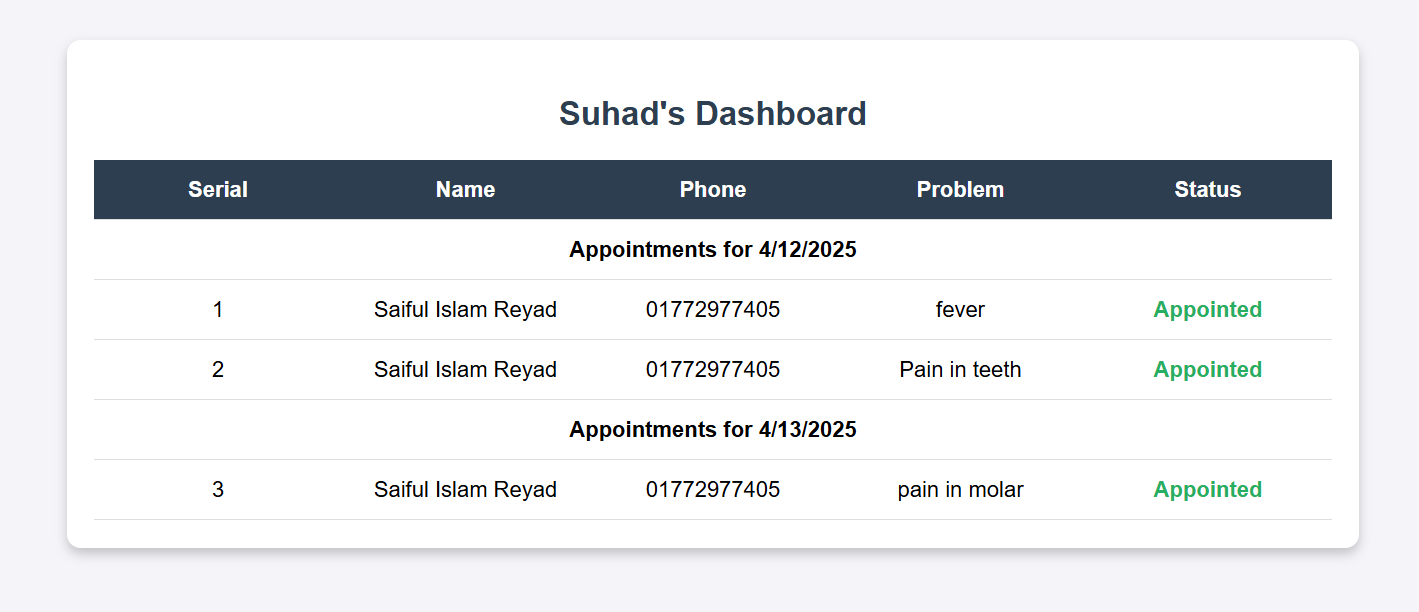


**Functionality Overview**

1. **Retrieve Patient ID**:
   * The patient\_id is retrieved from the URL via $\_REQUEST['patient\_id'] and is used to identify the patient who is booking the appointment. If the patient\_id is missing from the request, an error message is displayed.
2. **Form Setup**:
   * The form sends a POST request to submit\_appointment.php with the patient\_id and appointment details when submitted.
   * Hidden input is used to include the patient\_id in the form submission.
3. **Input Fields**:
   * **Medical Issue**: A text input field (<input type="text">) where the patient can describe the medical issue they are facing.
   * **Preferred Appointment Date**: A date input field (<input type="date">) that allows the patient to select their preferred appointment date.
4. **Dynamic Doctor and Specialization Selection**:
   * The form dynamically populates two dropdown menus:
     + **Specialization**: Initially shows a "Loading specializations..." message. Once the page is loaded, specializations are fetched from get\_specializations.php and displayed in the dropdown.
     + **Doctor**: Initially shows "Select a specialization first." Once a specialization is chosen, doctors associated with that specialization are fetched from get\_doctors.php and displayed in the doctor dropdown.
5. **Form Submission**:
   * When the user selects a doctor and submits the form, it sends the data (including the patient\_id) to submit\_appointment.php.

### Doctor's Dashboard Page

1. **View Appointments**:
   * A table displays the appointments that the doctor has for the day or upcoming appointments.
   * Each appointment includes:
     + Serial number (appointment identifier)
     + Patient's name
     + Patient's phone number
     + The problem or issue reported by the patient
     + The status of the appointment (e.g., pending, confirmed, completed)
2. **Dynamic Content**:
   * The appointments are dynamically populated into the table, ensuring that the doctor can see the latest updates without needing to reload the page manually.

  
**Functionality Overview**

* **Appointments Table**:
  + The table contains headers for the **Serial**, **Name**, **Phone**, **Problem**, and **Status**.
  + Under the <tbody> tag (with the ID appointmentsList), **appointment data will be added dynamically** using JavaScript (dash.js).
  + The JavaScript file (dash.js) is responsible for fetching the appointment data and populating the table rows with the corresponding details.

**Important Considerations:**

* **Dynamic Data Handling**:
  + This page depends on **JavaScript** (dash.js) to populate the appointment details dynamically, most likely from a database or API.
  + The page does not show the actual logic of data retrieval but assumes that dash.js handles fetching and displaying appointments properly.
* **User Interface**:
  + The table provides a structured and clear view of the appointments.
  + This simple layout ensures that the doctor can easily view and manage appointments.
* **No Direct User Interaction**:
  + The doctor can only see the appointments but may not be able to interact with them directly (such as confirming, rejecting, or editing appointments) based on the current code.

### Prescription Generator Page

**Connecting to the Database:**

The page begins by including a database connection file (db\_connect.php) to interact with the MySQL database. The appointment number (appointment\_no) is fetched from the URL using $\_GET.

**Fetching Appointment Information:**

The query retrieves detailed information about the appointment, including the patient's name, phone number, date of birth, gender, problem, appointment date, time, status, and associated doctor. It uses the appointment number to fetch these details from the database. If no data is found for the provided appointment number, an error message is displayed.

**Calculating Patient's Age:**

The function calculateAge() computes the patient's age based on their date of birth, which is used later in the form to display the patient’s age.

**Fetching Patient and Prescription Information:**

Another query is executed to get the patient's ID from the appointments table, which is then used to fetch the prescription history for that specific patient. For each prescription, the system retrieves:

* Prescription ID
* Date it was issued
* Chief Complaint (CC)
* Doctor’s name and qualification

Then, for each prescription, another query fetches the prescribed medicines, including dosage, when the medicine should be taken (before/after meals), and its duration.

**Displaying Patient and Appointment Info:**

The page then displays:

* **Patient Information:** This includes phone number, problem description, and date of birth.
* **Appointment Information:** This includes the date, time, and status of the appointment.

**Displaying Prescription History:**

* If the patient has previous prescriptions, the page will list them in a structured format:
  + Prescription ID
  + Issued Date
  + Doctor's Name and Qualification
  + Medicines prescribed (with details such as dosage, before/after meals, and duration)
* If no prescriptions are found, it displays a message indicating that no previous prescriptions exist.

**Patient Information Form:**

A form is provided to edit or display the patient's basic information, including:

* Name
* Age (calculated based on the date of birth)
* Gender
* Appointment Date

**Prescription Entry:**

* The page allows adding new prescriptions, including:
  + **C/C (Chief Complaint):** A text input.
  + **Investigation and Treatment Plan:** Input fields for entering quadrants (Q1, Q2, Q3, Q4) and extra investigation and treatment details.

**Dynamic Medicine Entry:**

* A section allows adding medicines dynamically. When the user clicks the "Add Medicine" button, new fields for medicine entry are added. Each medicine entry has:
  + A text input for the medicine name.
  + Checkbox options for when the medicine should be taken (morning, noon, night, or based on pain).
  + Checkbox for whether the medicine should be taken before or after eating.

**JavaScript for Dynamic Fields:**

The script at the end adds interactivity to the medicine entry:

* When the user clicks the "Add Medicine" button, new fields are dynamically added to the form.
* Each medicine entry has checkboxes for scheduling (morning, noon, night) and whether the medicine should be taken before or after meals.

**User Interface:**

The layout is split into two main sections:

* **Left Section:** Displays patient and appointment information.
* **Right Section:** Displays prescription information and includes fields for adding new medicines.

**Prescription Generation:**

A button (print) triggers the prescription generation functionality .

### 

### Filling up prescription and Result :

### …………

### Result: Filling up the medicines and treatment details, and printing this prescription will result in a structured, professional document containing the patient's information, prescribed medicines with dosages, timings (morning, noon, night), food instructions (before/after eating), and the treatment plan. This printed prescription ensures clarity for both the patient and pharmacist, improves communication, and maintains a record for future reference.

### 

**Functionality Overview**

The system is designed to interact with the database through standard CRUD operations:

* **Create:** When a new patient’s information and prescription details are entered, the system stores this data in the database for future reference.
* **Read:** Previously stored prescriptions and patient records can be retrieved and reviewed at any time.
* **Update:** Existing records can be edited if there are changes in treatment plans, medications, or patient details.
* **Delete:** Outdated or incorrect entries can be removed from the system to maintain clean and relevant data.

Additionally, the system can generate printable **prescription reports** based on the stored data. These reports are useful for documentation, patient handouts, and follow-up treatments.

### ****Project Demonstration****

This project is a **Digital Prescription Generator** designed for dental clinics. The main goal is to simplify the process of writing, managing, and printing prescriptions.

#### **How It Works:**

1. **Patient Information Entry:**
   * The user fills in fields like **Name**, **Age**, **Sex**, **Date**, and **Chief Complaint (C/C)**.
2. **Adding Medicines:**
   * Medicines can be added using a simple interface.
   * Each medicine includes:
     + Name
     + Timing (morning, noon, night)
     + Food instructions (before/after meals)
     + Number of days
3. **Preview & Print:**
   * Once all data is entered, clicking **"Generate Prescription"** opens a new page.
   * This page shows the formatted prescription aligned to a standard **A4 sheet** layout.
   * The user can then **print** the prescription directly for the patient.