Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Hassiba Benbouali de Chlef

Faculté des Sciences Exactes et Informatique

Département de L'informatique

Rapport du TP final

Module : *Développement web*

Thème :

Application web de gestion des rendez-vous des patients .

Réalisé par :

- *Loula Kawther.*
- *Nadour Mustafa.*

Sous L'encadrement du :

M. Mohamed Hadjhenni .

Année Universitaire : 2023/2024 .

Table des matières:

Introduction

Objectives and goals of the website :

Objectives :

1- Efficient Appointment Scheduling:

- Enable patients to easily schedule, reschedule, and cancel appointments online.
- Streamline the process for both patients and clinic staff, reducing manual workload.

2- Improved Patient Experience:

- Provide a user-friendly interface for patients to book appointments at their convenience.
- Enhance patient satisfaction by reducing wait times and improving accessibility to appointments.

3- Optimized Clinic Workflow:

- Assist clinic staff in managing appointments, reducing scheduling conflicts
- Improve the efficiency of clinic operations by automating appointment reminders and notifications.

4 - Enhanced Communication:

- Facilitate communication between patients and clinic staff regarding appointments.
- Ensure patients receive timely reminders and updates about their appointments.

Goals :

1. **Increase Appointment Accessibility:**
    - Aim to have at least X% of appointments booked through the website within the first year.
    - Reduce the average time it takes for patients to schedule an appointment by X%.
2. **Reduce No-Shows:**
    - Decrease the number of missed appointments by X% through improved reminders and confirmations.
    - Implement a system to track reasons for missed appointments to address common issues.
3. **Improve Staff Efficiency:**
    - Reduce the time spent on manual appointment scheduling tasks by X%.
    - Aim to eliminate scheduling conflicts by X% within the first six months of implementation.
4. **Enhance Patient Satisfaction:**
    - Conduct a patient satisfaction survey after six months to gauge improvements.
    - Aim for a satisfaction rating of at least X out of 10 based on feedback related to appointment scheduling.
5. **Ensure Data Security:**
    - Implement robust security measures to protect patient data.
    - Conduct regular audits to ensure compliance with data protection regulations (such as GDPR, HIPAA, etc.).
6. **Scalability and Adaptability:**
    - Design the system to be scalable, accommodating a growing number of patients and services.
    - Ensure the system is adaptable to changes in clinic procedures and requirements.

7. **Training and Support:**
    - Provide training for clinic staff on how to use the appointment management system.
    - Offer ongoing support to address any technical issues and gather feedback for improvements.

Importance and relevance of the project :

Importance :

1. **Efficiency and Functionality with PHP:**

    - **Server-Side Processing**: PHP is crucial for server-side processing, handling tasks such as appointment scheduling, database interactions, and user authentication.
    - **Data Management:** Through PHP, the website can efficiently manage patient data, appointment schedules, and staff information in the backend database.
    - **Dynamic Content Generation:** PHP allows for dynamic generation of HTML content based on user input and database queries, ensuring that the website provides up-to-date and relevant information to users.
    - **Secure Data Handling:** Utilizing PHP best practices ensures data security, protecting sensitive patient information in compliance with healthcare regulations.

.

2. **Structured and Accessible Content with HTML and CSS:**

    - **Semantic HTML:** Using HTML, the website's structure is clean and semantic, making it easier for search engines to index and rank the site. This improves its visibility to potential patients searching for clinic services.
    - **Styling and Layout with CSS:** CSS ensures consistent branding and user experience by styling the website's layout, colors, fonts, and overall design. This professional appearance builds trust with patients.

- **Accessibility Compliance:** HTML and CSS contribute to creating an accessible website, adhering to WCAG (Web Content Accessibility Guidelines) standards. This ensures that patients with disabilities can easily navigate and use the site.

Relevance :

1. **Digital Transformation in Healthcare:**
    - The use of PHP, HTML, and CSS aligns with the industry-wide shift towards digitalization in healthcare.
    - It positions the clinic as forward-thinking and responsive to patient needs in an increasingly tech-driven healthcare landscape.
2. **Enhanced Patient Engagement:**
    - The website's development technologies enable features such as online appointment booking, patient portals for medical history access, and interactive FAQs.
    - Patients appreciate the convenience and control over their healthcare experience, leading to higher engagement and satisfaction.
3. **Cost-Efficient Solution:**
    - Developing a web-based appointment management system is cost-efficient compared to traditional methods.
    - It reduces the need for manual appointment scheduling, paper-based records, and phone call confirmations, saving both time and resources for the clinic.
4. **Scalability and Future App Development:**
    - The use of PHP provides a scalable foundation for future app development, such as a mobile app for appointment management.
    - As patient needs evolve and mobile usage increases, having the ability to extend the system to mobile platforms ensures continued relevance and usability.

5. **Competitive Advantage and Brand Image:**
    - A well-developed website using PHP, HTML, and CSS gives the clinic a competitive edge over others with outdated systems or manual processes.

- It reflects positively on the clinic's brand image, showing a commitment to efficiency, patient-centered care, and technological innovation.

6. **Compliance and Security:**
    - The choice of development technologies ensures compliance with data protection regulations like HIPAA, safeguarding patient privacy.
    - As healthcare data security becomes more critical, patients are more likely to trust a clinic with a secure and modern online platform.

Design

Website Style Description

Modern and Approachable Design:

The clinic appointment management website is designed with a modern and approachable style, reflecting the clinic's commitment to providing contemporary healthcare services with a personal touch.

Clean and Professional Appearance:

The website's design exudes professionalism and cleanliness, mirroring the clinic's commitment to maintaining high standards of care and hygiene. The use of ample white space and a light color palette contributes to a welcoming and orderly feel.

Harmonious Color Scheme:

The color scheme is carefully chosen to create a harmonious and soothing atmosphere for visitors:

- **Light Bleu (#e6f0f1):** The dominant color conveys a sense of calmness and cleanliness, resembling the pristine environment of a healthcare facility.
- **Eastern Bleu (#16a4b7):** Used sparingly for buttons and links, this color adds a refreshing pop, inviting users to take action.
- **Scooter (#25bfcf):** An accent color used in special sections, it represents vitality and energy, aligning with the clinic's focus on patient wellness.

Brand Identity Integration:

The website prominently features the clinic's logo in the header, establishing brand identity and recognition. The logo's colors may align with the overall color scheme, reinforcing brand consistency.

Responsive and User-Centric Layout:

The layout is designed with user experience in mind, ensuring ease of navigation and accessibility across devices:

- **Header Navigation:** Clear and concise navigation links enable users to easily find information about the clinic's services and facilities.
- **Hero Image:** A captivating hero image in the showcase section immediately captures visitors' attention, providing a visual representation of the clinic's ambiance.
- **Call-to-Action:** Strategically placed call-to-action buttons encourage users to book appointments or explore services further, promoting engagement.
- Service Cards: The newsletter and services sections feature stylish cards with concise information, making it easy for users to browse and select services of interest.

Engaging Testimonials:

The "Point of View" section includes user testimonials presented in an engaging and trustworthy manner:

- **Background Image:** A background image adds a personal touch, depicting real-life interactions within the clinic.
- **Comment Box:** Testimonials are displayed in a comment-style box, creating a sense of authenticity and trust.
- **Quote and Author:** Each testimonial includes a quote with a stylized quotation mark and the author's name, adding credibility.

Humanized Team Introduction:

In the "Our Team" section, the clinic's medical professionals are introduced in a friendly and humanized manner:

- **Doctor Cards:** Each doctor is represented with an image and a brief description, showcasing their expertise and approachability.
- **Description:** Detailed descriptions highlight the team's qualifications and commitment to patient care, fostering trust and rapport.

Professional Details in Footer:

The footer section contains essential clinic information and contact details, ensuring users have easy access to relevant resources:

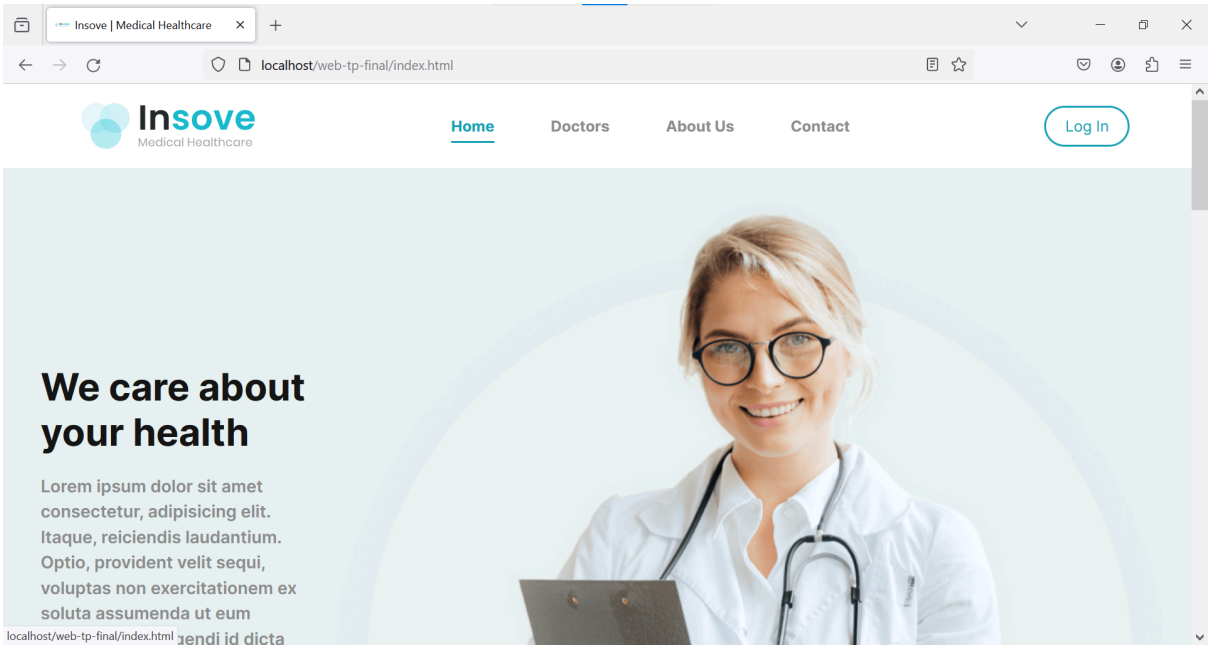
- **Contact Information:** Clinic address, phone number, and email are prominently displayed for convenience.
- **Copyright Notice:** A copyright notice at the bottom reflects the clinic's adherence to legal requirements and professionalism.
- **Back to Top Button:** A back-to-top button allows users to navigate back to the top of the page smoothly, enhancing user experience.

Interactive Elements and Hover Effects:

Throughout the website, subtle hover effects and interactive elements engage users:

- **Button Styles:** Buttons are designed with rounded corners and bold colors, inviting users to click for more information.
- **Hover Effects:** Elements such as buttons and links have gentle hover effects, providing visual feedback and enhancing interactivity.
- **Transitions:** Smooth transitions on hover and button clicks create a polished and seamless user experience, reflecting the clinic's attention to detail.

An image (mock-up) of the home page:



My home page uses a blue color scheme as the primary background color is a light blue (#e6f0f1) and text color is a dark blue (#171717). The font family used is Inter.

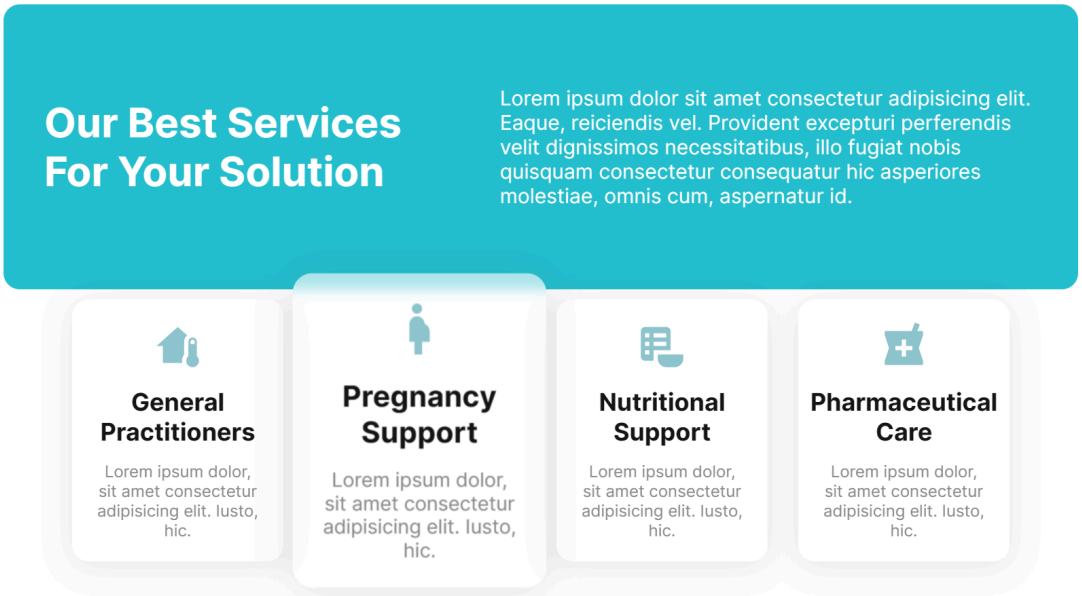
The layout is a single-page design with a header, navigation bar, hero image, and content area.

- **Header:** The header is located at the top of the webpage and spans the entire width of the page. It contains the clinic logo on the left and a navigation bar on the right. The navigation bar contains links to the home page, doctors, about us, and contact pages. There is also a login button.
- **Hero Image:** Below the header is a hero image on the right side of the content area. There is no text description for the image in the code you provided.

- **Content Area:** The content area is located in the middle of the webpage and consists of a title, a description of the clinic, and a sign-up button. The title text is large and bold, and the description text is smaller and uses a lighter weight font.

Overall, the design is clean and professional, with a focus on easy navigation and clear calls to action. The blue color scheme and Inter font family give the website a modern and trustworthy feel .

The page of services :

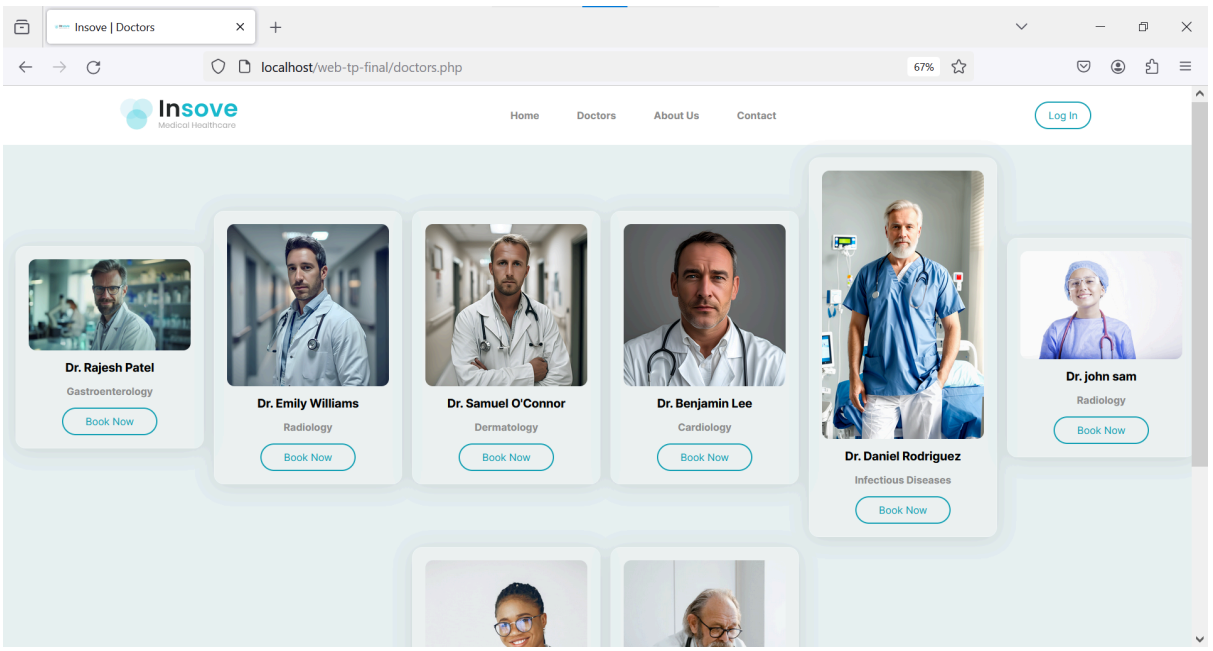


- **Background:** The background color is a light blue with a faint blur effect(rgba(255, 255, 255, 0.62) and backdrop-filter: blur(5px)).
- **Title:** The title is "Our Best Services For Your Solution" and the text color is white with a font size of 40px.
- **Description:** There is a description below the title written in white text with a font size of 20px. The description text reads: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, reiciendis vel. Provident excepturi perferendis velit dignissimos necessitatibus, illo fugiat nobis quisquam consectetur consequatur hic asperiores molestiae, omnis cum, aspernatur id."

- **Services:** Below the description, there are four service boxes arranged in a grid layout. Each service box has an icon, a title, and a short description. From left to right, the services are:
    - General Practitioners
    - Pregnancy Support
    - Nutritional Support
    - Pharmaceutical Care

Overall, the design is clean and professional with a focus on the four services offered. The blue color scheme and white text create a sense of trust and clarity.

An image of the Doctors page :



The page is a list of doctors working at My clinic Insove. Here's a more detailed description of the doctor's page:

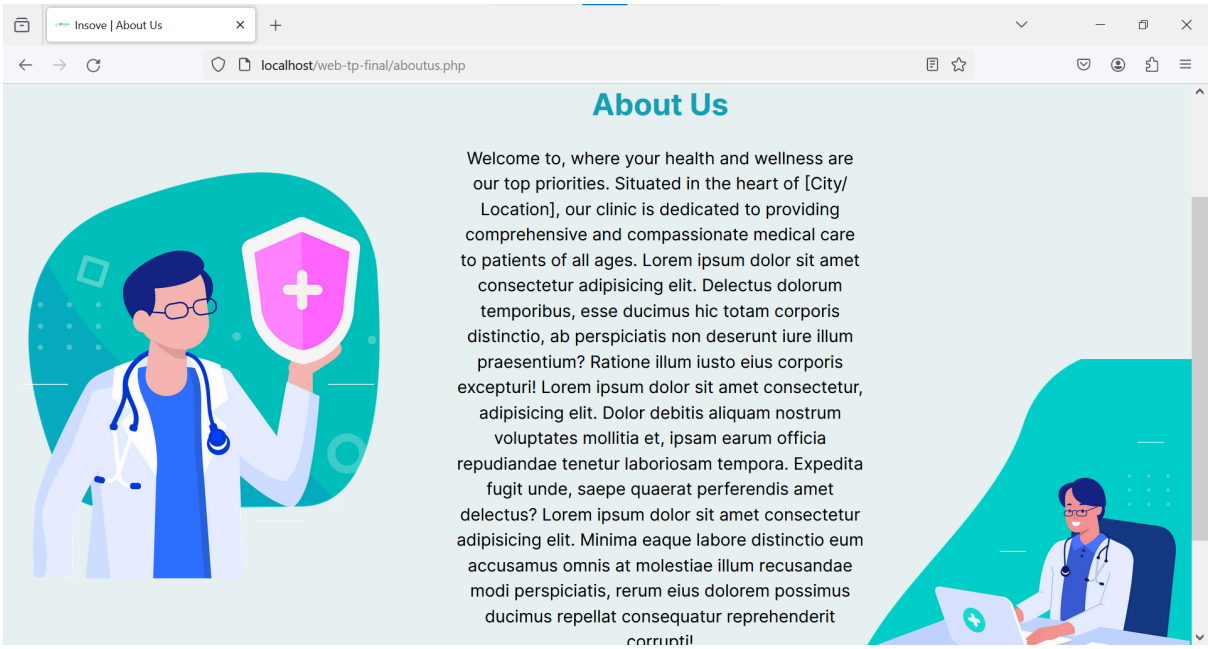
- **Header:** The header section contains the clinic logo on the left and navigation links on the right. The navigation links are for Home, Doctors (current page), About Us, and Contact. There is also a login button on the far right.
- **Main Content:** The main content area displays a list of doctor cards. Each doctor card contains an image of the doctor, their name, their specialty (e.g.,

Gastroenterology, Radiology), and a “Book Now” button. Clicking the button likely allows users to book an appointment with the doctor.

- **Footer:** The footer contains the copyright information for Insove Clinic.

Overall, the design of this page is clean and professional. The use of a light blue background and doctor cards with a glass effect creates a modern and trustworthy feel. The content is well organized, making it easy for users to find the information they are looking for.

An image of the About Us page :

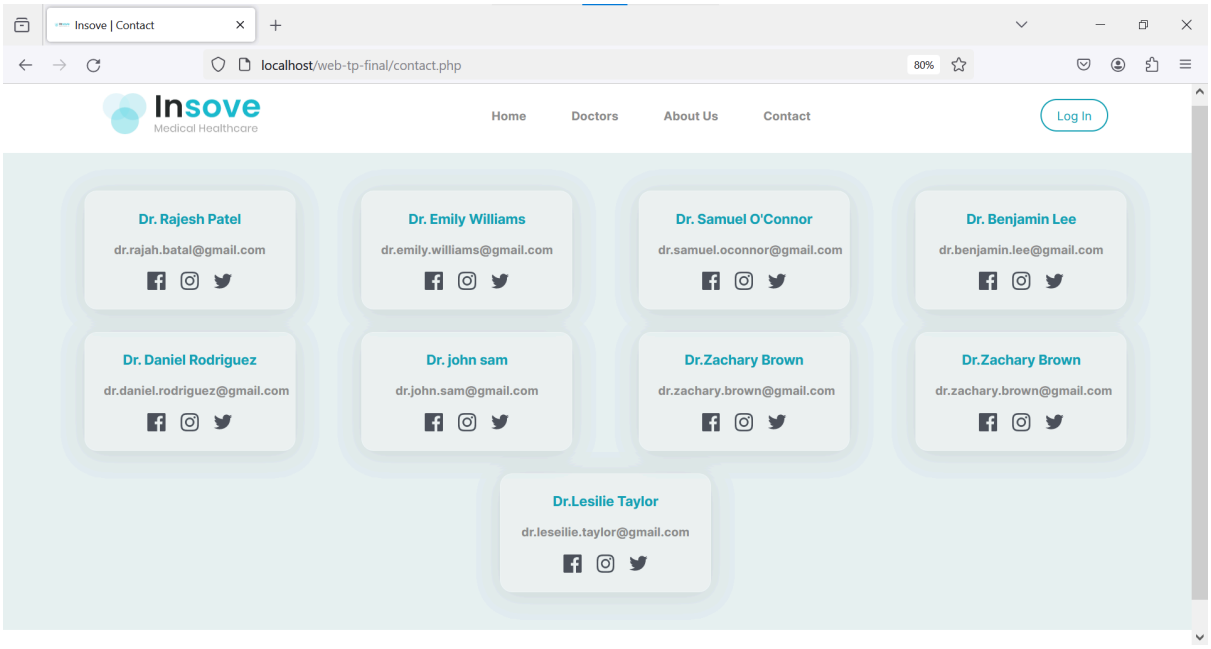


The Insove About Us page is a standard clinic bio page. Here’s a more detailed description:

- **Header:** The header section contains the clinic logo on the left and navigation links on the right. The navigation links are for Home, Doctors, About Us (current page), and Contact. There is also a login button on the far right.
- **Main Content:** The main content area is split into two sections. The first section contains a large logo for the clinic. The second section contains a welcome message from the clinic and a short description of the services they provide.
- **Footer:** The footer contains the copyright information for Insove Clinic.

Overall, the design of this page is clean and professional. The use of a light blue background and white text creates a modern and trustworthy feel. The content is well organized, making it easy for users to find the information they are looking for.

An image of the About Us page :

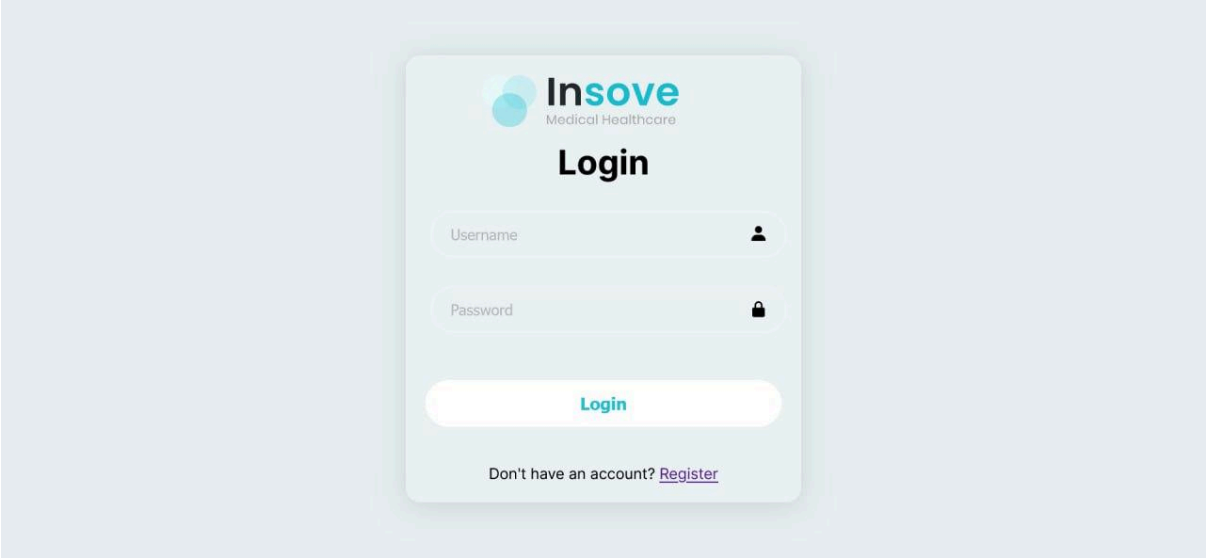


The Insove Contact page is a standard clinic contact information page. Here's a more detailed description of the page:

- **Header:** The header section contains the clinic logo on the left and navigation links on the right. The navigation links are for Home, Doctors, About Us, and Contact (current page). There is also a login button on the far right.
- **Main Content:** The main content area contains a list of the clinic's doctors. Each doctor's name, email address, and social media links are displayed.
- **Footer:** The footer contains the copyright information for Insove Clinic.

Overall, the design of this page is clean and professional. The use of a light blue background and white text creates a modern and trustworthy feel. The contact information for the clinic's doctors is well organized, making it easy for users to find the information they are looking for. However, it would be nice if the contact information for the clinic itself, such as phone number or address, was also included on the page.

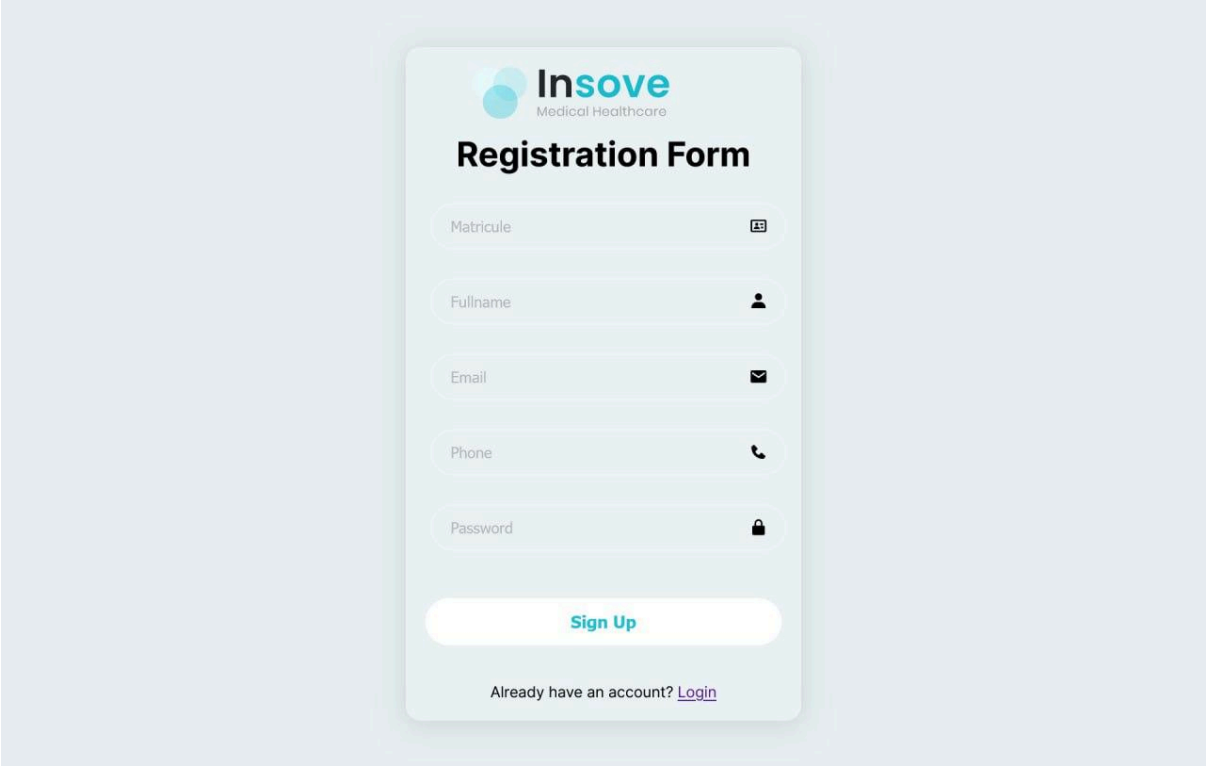
An image of the Login page :



The login page appears to be for a medical healthcare service provider called Insove. Here's a description of the login page:

- **Insove logo:** The login page features a logo for Insove at the top left corner.
- **Login title:** Below the logo, there's a heading that says "Login".
- **Username and password fields:** There are two input fields where users can enter their username and password credentials. Placeholder text within the fields reminds users to enter their username and password.
- **Login button:** Below the username and password fields, there's a blue button labeled "Login". Clicking this button likely submits the login form.
- **Register link:** Below the login button, there's a text link that says "Don't have an account? Register". Clicking this link would likely redirect users to a registration page where they can create a new account.

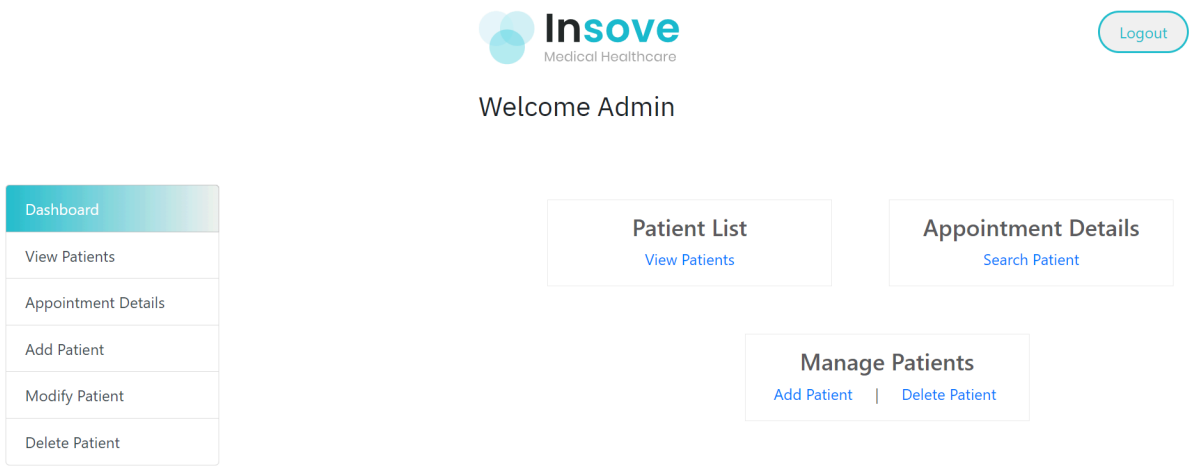
An image of the registration page :



The image is a registration form for a medical healthcare company called Insove. Here's a description of the design elements on the registration page:

- **Insove logo:** The registration form features a logo for Insove at the top left corner.
- **Registration form title:** Below the logo, there's a heading that says "Registration Form".
- **Input fields:** There are five input fields where users can enter their matricule, full name, email address, phone number, and password for registration. Placeholder text within the fields reminds users what information to enter in each field.
- **Password visibility toggle:** It appears that there is no icon to toggle password visibility on the registration form.
- **Sign up button:** Below the password input field, there's a blue button labeled "Sign Up". Clicking this button likely submits the registration form.
- **Login link:** Below the sign up button, there's a text link that says "Already have an account? Login". Clicking this link would likely redirect users to a login page where they can sign in to their existing account.

An image of the User Dashboard page :



The image is a dashboard admin panel of a medical healthcare company called Insove. Here's a description of the design elements on dashboard admin page:

Layout

- The dashboard is divided into two main sections:
    - **Navigation bar:** This section located at the top includes a logout button and potentially other navigation options (although only logout is implemented in the code you provided).
    - **Content area:** This section occupies most of the page and includes several tabs. Each tab displays different functionalities.

Tabs

The content area contains multiple tabs that can be switched between using the navigation bar on the left. Here are the functionalities available based on the code:

- **Dashboard:** This tab appears to be the default landing page and displays a summary of the clinic's activity, potentially including patient numbers and appointments.
- **View Patients:** This tab allows admins to search for patients and view a list of patients, likely including details such as fullname, gender, contact information, and email.

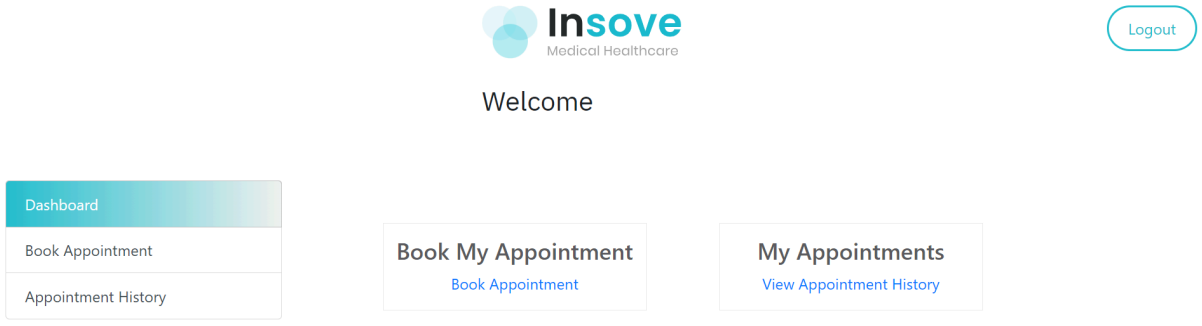
- **Appointment Details:** This tab likely allows admins to search for appointments based on patient details and view details about appointments.
- **Add Patient:** This tab provides a form for admins to add new patients to the system. The form likely captures details such as matricule, phone number, fullname, email, and password.
- **Modify Patient:** This tab provides a form for admins to modify existing patient information. The form likely captures similar details as adding a patient.
- **Delete Patient:** This tab allows admins to delete patients from the system. It prompts for confirmation before deleting a patient.

Styling:

The code uses Bootstrap for layout and styling. The styling includes:

- A blue color scheme as the primary accent color.
- A responsive layout that adapts to different screen sizes.
- Use of FontAwesome icons for visual elements.

An image of the User Dashboard page :



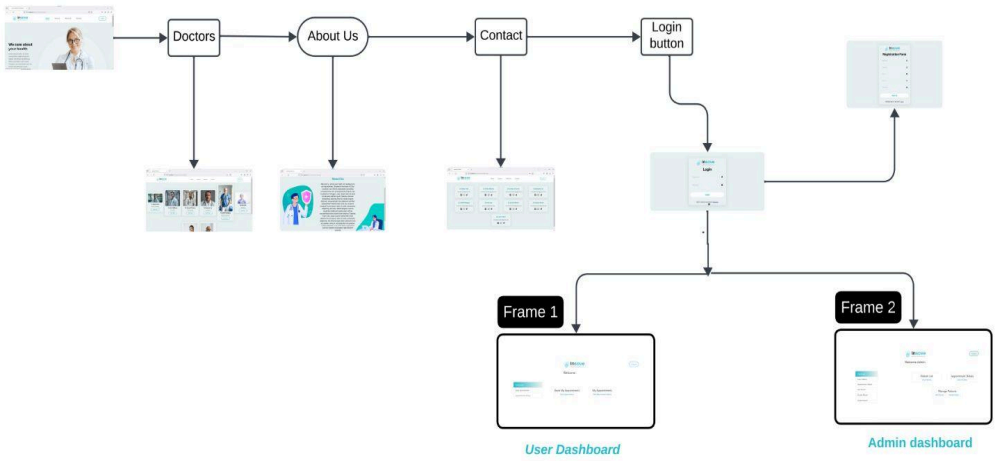
The Insove user dashboard has a light blue background with a dark blue gradient on the left hand side. The text at the top includes the clinic logo and says "Insove | User".

Below this is a navigation bar with three tabs: "Dashboard", "Book Appointment", and "Appointment History". The "Dashboard" tab is currently selected.

The center of the screen shows the contents of the selected tab. In this case, the "Dashboard" tab is selected and it shows two cards. The first card is titled "Book My Appointment" and it has a button that says "Book Appointment" when clicked it will take you to the book appointment tab. The second card is titled "My Appointments" and it has a button that says "View Appointment History" that when clicked will take you to the appointment history tab.

design of the website to be developed :

This diagram shows all the pages and the links between these pages:



Development

Clinic Database Overview

The Clinic database is designed to manage the administrative tasks of a clinic, including patient registration, appointment scheduling, and user authentication. Below is an overview of the database schema and its components:

Tables:

1. **Admin:**
    - **Fields:**
        - `admin_id` (Primary Key): Unique identifier for each admin.
        - `username`: Username of the admin.
        - `email`: Email address of the admin.
        - `admin_password`: Password of the admin.
    - Description: Stores information about the clinic administrators.
2. **Patient:**
    - **Fields:**
        - `patient_id` (Primary Key): Unique identifier for each patient.
        - `matricule`: Matricule or unique identifier for the patient.
        - `fullname`: Full name of the patient.
        - `email`: Email address of the patient.
        - `phone`: Phone number of the patient.
        - `user_password`: Password of the patient.
        - `registration_date`: Timestamp indicating the date of registration.
        - `admin_id` (Foreign Key): Reference to the admin who registered the patient.
    - Description: Stores details of patients registered at the clinic.
3. **Appointments:**
    - **Fields:**

        - `appointment_id` (Primary Key): Unique identifier for each appointment.
        - `service_name`: Name of the service for the appointment.
        - `doctor_name`: Name of the doctor for the appointment.
        - `appointment_time`: Date and time of the appointment.
        - `admin_id` (Foreign Key): Reference to the admin responsible for the appointment.
        - `patient_id` (Foreign Key): Reference to the patient associated with the appointment.
    - Description: Tracks appointments scheduled at the clinic, including service, doctor, and timing details.

Relationships:

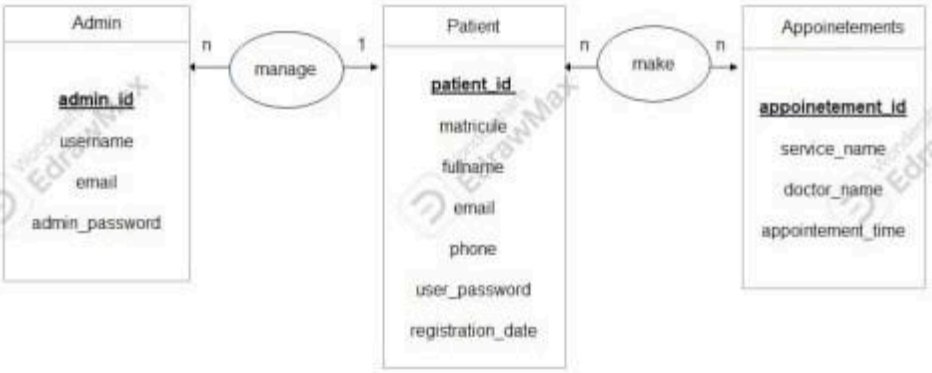
- The `Admin` table has a one-to-many relationship with the `Patient` table, as one admin can register multiple patients.
- The `Admin` table also has a one-to-many relationship with the `Appointments` table, as one admin can schedule multiple appointments.
- The `Patient` table has a one-to-many relationship with the `Appointments` table, as one patient can have multiple appointments.

Database Usage:

- **Registration:** Admins can register new patients by adding their details to the `Patient` table.
- **Authentication:** Admins can log in using their username and password stored in the `Admin` table.
- **Appointment Scheduling:** Admins can schedule appointments by adding them to the `Appointments` table, specifying the service, doctor, timing, and associated patient.

This database schema provides a structured approach to manage clinic-related data, facilitating efficient administration and patient care.

MCD:



MLD:

Admin(**admin_id**, username, email, admin_password);

Patient(**patient_id**, matricule, fullname, email, phone, user_password, registration_date, **#admin_id**);

Appointements(**appoinetement_id**, service_name, doctor_name, appointement_time);

make(**#patient_id , #appoinetement_id**);

Explanation of the code for each page PHP :

Login Authentication :

```php
<?php
include("bdd.php");
session_start();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $query = "SELECT * FROM Admin WHERE username = ? AND admin_password = ?";
    $stmt = $connection->prepare($query);
    $stmt->bind_param("ss", $username, $password);
    $stmt->execute();
    $result = $stmt->get_result();
    $row = mysqli_fetch_assoc($result);
    if ($result->num_rows == 1) {

        if($row['username'] == $username && $row['admin_password'] == $password){
          $_SESSION['admin_password'] = $password;
          $_SESSION['username'] = $username;
          header('Location: ./adminBoard.php');
        }else{
          echo "<div class='alert alert-danger'> Login failed </div>";
        }

    } else {
        header('Location: ./login.php?error=Invalid credentials');
    }
}
?>
```

This code snippet handles user login with session management for an admin panel, likely using

a prepared statement for security. Here's a breakdown line by line:

1. **<?php** - This line marks the beginning of a PHP code block.
2. **include("bdd.php");** - This line includes a file named "bdd.php" which likely contains your database connection details.
3. **session_start();** - This line starts a PHP session. Sessions are used to store user information across multiple pages without having to pass it in every request.
4. **if ($_SERVER["REQUEST_METHOD"] == "POST") {** - This line starts an if statement that

checks if the request method is POST. This means the user submitted a login form using the POST method, likely containing username and password.

Inside the POST check (assuming the form was submitted):

5-6. Variable assignments:

● **$username = $_POST['username'];** : This line retrieves the value from the "username" form field using $_POST and assigns it to the $username variable.
● **$password = $_POST['password'];** : Similar to the previous line, this retrieves the value from the "password" form field and assigns it to the $password variable.

7. **$query = "SELECT * FROM Admin WHERE username = ? AND admin_password = ?";** - This line constructs an SQL SELECT query to retrieve all columns (*) from the "Admin" table, but only for records where the "username" matches the provided $username and the "admin_password" matches the provided $password. The question marks (?) represent placeholders for the actual username and password values, which will be filled later using prepared statements for better security.
8. **$stmt = $connection->prepare($query);** : This line prepares the SQL statement using mysqli_prepare() on the database connection ($connection). This creates a statement object ($stmt) that can be used for secure execution.
9. **$stmt->bind_param("ss", $username, $password);** : This line binds the actual values for the placeholders in the prepared statement ($stmt). It uses "ss" format specifiers indicating two strings, and binds the $username and $password variables to the corresponding placeholders.
10. **$stmt->execute();** : This line executes the prepared statement with the bound values.
11. **$result = $stmt->get_result();** : This line retrieves the result set from the executed prepared statement and stores it in the $result variable.

12. **$row = mysqli_fetch_assoc($result);** :This line fetches the first row from the result set ($result) as an associative array and stores it in the $row variable.

13. **if ($result->num_rows == 1) {** : This line starts an if statement that checks if the number of rows returned by the query ($result->num_rows) is equal to 1. This indicates a single matching record was found for the username and password.

Inside the successful login check (one matching record found):

14. **if($row['username'] == $username && $row['admin_password'] == $password){** - This line performs an additional security check by comparing the retrieved username ($row['username']) and password ($row['admin_password']) from the database with the original submitted values ($username and $password). This might be an extra layer of validation in case something went wrong with the prepared statement execution.

15-16. Session variables and redirection:

● **$_SESSION['admin_password'] = $password;** : This line stores the password (not recommended for security reasons) in a session variable named admin_password. It's generally better to avoid storing passwords in sessions. Consider using a more secure method like a hash.

● **$_SESSION['username'] = $username;** : This line stores the username in a session variable named username.

● **header('Location: ./adminBoard.php');** : This line sends a header to redirect the user to the "adminBoard.php" page, likely the admin panel after successful login.

Login failed (no matching record or additional security check failed):

17. **}else{** - This else block executes if the additional security check inside the successful login check fails (point 14).

18. **echo " Login failed ";** : This line displays an HTML snippet with a bootstrap class likely indicating a failure.

Book an appointment :

```php
<?php
    include("bdd.php");
    if($_SERVER["REQUEST_METHOD"] == "POST"){
        if(isset($_POST['bookNow'])){

            $service = $_POST['Service'];
            $doctor = $_POST['Doctor'];
            $appointement_date = $_POST['date'];

            if (!is_string($service)) {
                echo "<script>alert('Please enter a valid data')</script>";
            }else {
                $query = "INSERT INTO Appoinetements (service_name, doctor_name, appointement_time) VALUES ('$service', '$doctor', '$appointement_date')";
                try{
                    $result = mysqli_query($connection, $query);
                    if($result){
                        echo"<script>alert('Appointement booked successful')</script>";
                    }else{
                        echo "<script>alert('Appointement booking failed')</script>";
                    }
                }catch(mysqli_sql_exception){
                    echo "<script>alert('Appointement booking failed')</script>";
                }
            }
        }else {
            echo "<script>alert('Please submit your data')</script>";
        }
    }
    mysqli_close($connection);
?>
```

This code snippet handles booking appointments in a clinic system.

Here's a breakdown line by line:

1. **alert('Please enter valid data')";** : If the validation in the previous line fails (service is not a string), this line displays an alert message using JavaScript indicating "Please enter valid data". Inside the service validation (assuming service data is valid): 10. }else { - This else block executes if the service validation passes (it's a string).

11. **$query = "INSERT INTO Appointments (service_name, doctor_name, appointement_time) VALUES ('$service', '$doctor', '$appointement_date')";**

: This line constructs an SQL INSERT query to add a new record to the "Appointments" table. It defines the columns (service_name, doctor_name, appointement_time) and sets their values to the corresponding variables ($service, $doctor, $appointement_date).

12**. try{** : This line marks the beginning of a try block. The code within this block will attempt to execute the database insertion.

13. **$result = mysqli_query($connection, $query);** : This line attempts to execute the INSERT query using mysqli_query() and the connection object ($connection). It stores the result (success or failure) in the $result variable. Inside the query execution (handling success or failure):

14. **if($result){** : This line starts an if statement that checks if the $result is not false (indicating successful query execution).

15. **echo"";** **:** If the query execution is successful, this line displays a JavaScript alert message indicating "Appointment booked successful".
16. }**else{** : This else block executes if the query execution fails ($result is false).
17. **echo "";** : If the query fails, this line displays a JavaScript alert message indicating "Appointment booking failed".
18. } - This line closes the if block for handling query execution success or failure.
19. **}catch(mysqli_sql_exception){** : This line starts a catch block within the try block. This catch block will only execute if an exception (error) occurs during the database interaction within the try block.
20. **echo "";** : If a database exception occurs, this line displays a JavaScript alert message indicating "Appointment booking failed". This might be a more generic message compared to the specific failure reason in the previous else block.

Connection to bdd :

```php
<?php
    $bd_host = "localhost";
    $bd_user = "root";
    $bd_password = "";
    $bd_name = "clinic";
    try{
        $connection = mysqli_connect($bd_host, $bd_user, $bd_password, $bd_name);
     }catch(mysqli_sql_exception ){
        echo "connection failed";
     }
?>
```

Your code defines database connection details and attempts to connect to the database. Here's
a breakdown line by line:
1. **<?php** - This line marks the beginning of a PHP code block.
2-4. Variable assignments:
● **$bd_host = "localhost";** - This line creates a variable named $bd_host and assigns the value "localhost" to it. This likely represents the hostname or IP address of the database server. **"localhost"** usually refers to the machine where the PHP code itself is running.

● **$bd_user = "root"**; - This line creates a variable named $bd_user and assigns the value "root" to it. This is likely the username used to connect to the database. Security Note: Using "root" is powerful but not recommended for most cases due to its high level of access. It's generally better to create a dedicated user with specific permissions for your application.
● **$bd_password = "";** - This line creates a variable named $bd_password and assigns an empty string "" to it. This represents the password for the database user. Important security note: It's strongly discouraged to store database credentials directly in code. This information can be leaked if your code is not secured properly. Consider using environment variables or a
separate configuration file to store these details.
5. **$bd_name = "clinic";** - This line creates a variable named $bd_name and assigns the value "clinic" to it. This represents the name of the database you want to connect to.
6. **try{** - This line marks the beginning of a try block. The code within this block will be attempted to be executed.
7. **$connection = mysqli_connect($bd_host, $bd_user, $bd_password, $bd_name); -** This line attempts to connect to the database using the mysqli_connect() function. It takes four arguments:
○ **$bd_host:** The hostname or IP address of the database server (defined earlier).
○ **$bd_user:** The username used to connect to the database (defined earlier).
○ **$bd_password:** The password for the database user (defined earlier).
○ **$bd_name:** The name of the database to connect to (defined earlier).
○ The function returns a connection object ($connection) if successful, or FALSE on failure.
8. **}catch(mysqli_sql_exception){** - This line marks the beginning of a catch block. This block will only execute if an exception (error) occurs within the try block. The specific exception type being caught here is mysqli_sql_exception, which means it will catch any exceptions related to MySQL database interactions.
9. **echo "connection failed";** - This line, inside the catch block, will be executed if the mysqli_connect() function fails. It simply echoes a message "connection failed" to indicate the connection attempt was unsuccessful.

Overall, this code snippet defines database connection details and attempts to establish a connection using mysqli_connect(). It includes a basic error handling mechanism using a try-catch block to catch and display a message if the connection fails.

View appointements*:*

```php
<?php
    include("bdd.php");
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $query = "SELECT * FROM Appointements";
        $result = mysqli_query($connection,$query);

        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                echo "<tbody>";
                echo "<tr>";
                echo "<td>". $row['appointement_id']. "</td>";
                echo "<td>". $row['service_name']. "</td>";
                echo "<td>". $row['doctor_name']. "</td>";
                echo "<td>". $row['appointement_time']. "</td>";
                echo "</tr>";
                echo "</tbody>";
            }
        } else {
            echo "Aucun rendez-vous trouvé";
        }
    }
    mysqli_close($connection);
?>
```

This code snippet retrieves all appointments from a database and displays them in an HTML table format. Here's a breakdown line by line:

1. **num_rows > 0) {** - This line checks if the query execution resulted in any records (i.e., $result->num_rows is greater than 0).

7. **while($row = $result->fetch_assoc()) {** - This line starts a while loop that iterates as long as there are rows to be processed from the $result. Inside the loop, $row = $result->fetch_assoc() retrieves each row as an associative array, where column names are keys and corresponding values are elements.

8. **echo "";** - This line starts an HTML table body () tag. This is likely part of the table structure to display appointment information.

9. **echo "";** - This line starts a table row () tag, creating a new row for each appointment record.

10. **echo "". $row['appointement_id']. "";** - This line echoes (prints) the value of the "appointement_id" column from the current row ($row) within a table data cell (). This repeats for the following lines, displaying each column's value in a separate cell.

11-14. Similar to line 10: These lines follow the same pattern, echoing the values of "service_name", "doctor_name", and "appointement_time" columns from the $row array within table data cells.

16. **echo "";** - This line closes the HTML table body () tag. This likely signifies the end of the table where appointment information is displayed.

17. **echo "Aucun rendez-vous trouvé";** - This line displays a message in French indicating "No appointments found".

22. **mysqli_close($connection);** - This line closes the database connection using mysqli_close().

Affiche patients:

```php
<?php
    include("bdd.php");
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $query = "SELECT * FROM Patient";
        $result = mysqli_query($connection,$query);

        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                echo "<tbody>";
                echo "<tr>";
                echo "<td>". $row['patient_id']. "</td>";
                echo "<td>". $row['fullname']. "</td>";
                echo "<td>". $row['matricule']. "</td>";
                echo "<td>". $row['email']. "</td>";
                echo "<td>". $row['phone']. "</td>";
                echo "<td>". $row['registration_date']. "</td>";
                echo "</tr>";
                echo "</tbody>";
            }
        } else {
            echo "Aucun rendez-vous trouvé";
        }
    }
    mysqli_close($connection);
?>
```

This code snippet retrieves all patient information from a database and displays it in an HTML table format.

Here's a breakdown line by line:

1. **num_rows > 0) {** - This line checks if the query execution resulted in any records (i.e., $result->num_rows is greater than 0).

7. **while($row = $result->fetch_assoc()) {** - This line starts a while loop that iterates as long as there are rows to be processed from the $result. Inside the loop, $row = $result->fetch_assoc() retrieves each row as an associative array, where column names are keys and corresponding values are elements.

8. **echo "";** - This line starts an HTML table body () tag. This is likely part of the table structure to display patient information.

9. **echo "";** - This line starts a table row () tag, creating a new row for each patient record.

10. **echo "". $row['patient_id']. "";** - This line echoes (prints) the value of the "patient_id" column from the current row ($row) within a table data cell (). This repeats for the following lines, displaying each column's value in a separate cell.

11-15. Similar to line 10: These lines follow the same pattern, echoing the values of "fullname", "matricule", "email", "phone", and "registration_date" columns from the $row array within table data cells.

20. **echo "Aucun rendez-vous trouvé";** - This line displays a message in French indicating "No appointments found" (assuming "rendez-vous" refers to appointments in this context).

Search patient:

```php
<?php
    include("bdd.php");
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $matricule = filter_input(INPUT_POST, 'matricule', FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);
        $query = "SELECT * FROM Patient  WHERE matricule= $matricule ";
        $result = mysqli_query($connection,$query);
      if(isset($_POST['search'])){
        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                echo "<tbody>";
                echo "<tr>";
                echo "<td>". $row['patient_id']. "</td>";
                echo "<td>". $row['matricule']. "</td>";
                echo "<td>". $row['fullname']. "</td>";
                echo "<td>". $row['email']. "</td>";
                echo "<td>". $row['phone']. "</td>";
                echo "<td>". $row['registration_date']. "</td>";
                echo "</tr>";
                echo "</tbody>";
            }
        } else {
            echo "Aucun rendez-vous trouvé";
        }
    }
  }
    mysqli_close($connection);
?>
```

This code snippet searches for a patient in a database and displays the information in an HTML table format. Let's break it down line by line:

1. **num_rows > 0) {** - This line checks if the query execution resulted in any records (i.e., $result->num_rows is greater than 0).

9. **while($row = $result->fetch_assoc()) {** - This line starts a while loop that iterates as long as there are rows to be processed from the $result. Inside the loop, $row = $result->fetch_assoc() retrieves each row as an associative array, where column names are keys and corresponding values are elements.

12. **echo "". $row['patient_id']. "";** - This line echoes (prints) the value of the "patient_id" column from the current row ($row) within a table data cell (). This repeats for the following lines, displaying each column's value in a separate cell.

13-17. Similar to line 12: These lines follow the same pattern, echoing the values of "matricule", "fullname", "email", "phone", and "registration_date" columns from the $row array within table data cells. (no results found): 21. } else { - This else block executes if no records were found ($result->num_rows is not greater than 0). 22. echo "Aucun rendez-vous trouvé"; - This line displays a message in French indicating "No appointments found".

Supprimer patient :

```php
<?php
    include("bdd.php");
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if(isset($_POST['delete'])){

            $matricule = filter_input(INPUT_POST, 'matricule', FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);
            $sql = "DELETE FROM Patient WHERE matricule='$matricule'";

            if ($connection->query($sql) === TRUE) {
                echo "<script>alert('Patient Deleted succesfully')</script>";
                header("Location: adminBoard.php");
                exit;
            } else {
                echo "<script>alert('NO Patien with this name ')</script>";
            }
        }
    }
    mysqli_close($connection);
?>
```

This code snippet handles deleting a patient record from a database.

Let's break it down line by line:

1. **query($sql) === TRUE)** { - This line attempts to execute the delete query using the query() method of the $connection object. It checks if the query execution was successful (i.e., $connection->query($sql)) and compares the return value to TRUE using strict comparison (===).

9. **header("Location: adminBoard.php");** - This line sends a header to redirect the user to the "adminBoard.php" page, likely an admin panel.

10. **exit;** - This line exits the script after the redirection.

16. mysqli_close($connection); - This line closes the database connection using mysqli_close(). Overall, this code snippet retrieves the patient ID from a submitted form,

sanitizes it, and attempts to delete the corresponding record from the database. It provides feedback messages based on the success or failure of the deletion query. Here are some additional points to consider:

- Security: Consider adding additional confirmation steps before deleting a patient record to prevent accidental deletion.
- Error Handling: The error message could be improved to provide more specific information about the failure (e.g., database error, patient not found).

Modify patient:

```php
<?php
    include("bdd.php");
    if($_SERVER["REQUEST_METHOD"] == "POST"){
            if(isset($_POST['modify'])){
            $matricule = filter_input(INPUT_POST, 'matricule', FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);
            $fullname = filter_input(INPUT_POST, 'fullname', FILTER_SANITIZE_SPECIAL_CHARS);
            $email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_SPECIAL_CHARS , FILTER_VALIDATE_EMAIL);
            $phone = filter_input(INPUT_POST, 'phone', FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);

            $query = "UPDATE Patient SET fullname='$fullname', email='$email', phone='$phone' WHERE matricule= '$matricule'";
            $result = mysqli_query($connection, $query);
            if ($result === TRUE) {
                echo "Informations of patient modified succesfully";
                header("Location: adminBoard.php");
                exit;
            } else {
                echo "Erreur occured during process!";
            }
        }
    }
    mysqli_close($connection);
?>
```

This code snippet handles updating patient information in a database. Let's break it down line by

line:

1. <?php - This line marks the beginning of a PHP code block.

2. include("bdd.php"); - This line includes a file named "bdd.php" which likely contains your database connection details.

3. **if($_SERVER["REQUEST_METHOD"] == "POST")**{ - This line starts an if statement that checks if the request method is POST. This means the user submitted a form using the

POST method, likely containing the modifications.

4. **if(isset($_POST['modify']))**{ - This line is another if statement that checks if a form field named "modify" exists within the submitted form data using isset(). This field might be a hidden field or a button that indicates an update attempt.

Inside the second if statement (update check):

5. **$matricule = filter_input(INPUT_POST**, 'matricule', **FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);** - This line retrieves the value from the "matricule" form field using filter_input(). It also performs two actions:

○ **FILTER_SANITIZE_SPECIAL_CHARS** removes any special characters from the input, potentially preventing security vulnerabilities.

○ **FILTER_VALIDATE_INT** attempts to validate the input as an integer (whole number). If the validation fails, it might return null depending on PHP configuration.

6. $fullname = filter_input(INPUT_POST, 'fullname', **FILTER_SANITIZE_SPECIAL_CHARS);** - Similar to the previous line, this retrieves and sanitizes the "fullname" form field data, removing special characters.

7. **$email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_SPECIAL_CHARS , FILTER_VALIDATE_EMAIL);** - This line retrieves the "email" form field data, sanitizes it, and attempts to validate it as a valid email address using FILTER_VALIDATE_EMAIL. Again, validation might return null on failure.

8. **$phone = filter_input(INPUT_POST, 'phone', FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);** - This line retrieves the "phone" form field data, sanitizes it, and attempts to validate it as an integer.

Update query execution:

9. **$query = "UPDATE Patient SET fullname='$fullname', email='$email', phone='$phone'**

WHERE matricule= '$matricule'"; : This line constructs an SQL UPDATE query to modify the "Patient" table. It sets the new values for "fullname", "email", and "phone" based on the submitted data, and uses the "matricule" (likely a patient ID) as the condition to identify the

specific patient record to update.

10. **$result = mysqli_query($connection, $query);** - This line attempts to execute the update

query using mysqli_query() and the connection object ($connection). It stores the result (success or failure) in the $result variable.

11. **if ($result === TRUE) {** - This line checks if the query execution was successful (i.e., $result is strictly equal to TRUE).

12. echo "Informations of patient modified succesfully"; - If the update was successful, this line displays a success message.

13. **header("Location: adminBoard.php");** - This line sends a header to redirect the user to

the **"adminBoard.php"** page, likely an admin panel.

14. **exit;** - This line exits the script after the redirection.

16. **echo "Erreur occured during process!";** - This line displays an error message indicating the update process failed.

Ajouter patient :

```php
<?php
  include("bdd.php");
  if($_SERVER["REQUEST_METHOD"] == "POST"){
    if(isset($_POST['signup'])){
      $matricule = filter_input(INPUT_POST, 'matricule', FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);
      $fullname = filter_input(INPUT_POST, 'fullname', FILTER_SANITIZE_SPECIAL_CHARS);
      $email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_SPECIAL_CHARS , FILTER_VALIDATE_EMAIL);
      $phone = filter_input(INPUT_POST, 'phone', FILTER_SANITIZE_SPECIAL_CHARS, FILTER_VALIDATE_INT);
      $password = $_POST['password'];
      if (!is_string($fullname)) {
          echo "<script>alert('Please enter a valid data')</script>";
      }else {
          $hashed_password = password_hash($password, PASSWORD_DEFAULT);
          $query = "INSERT INTO Patient (matricule,fullname,email,phone,user_password) VALUES ('$matricule','$fullname','$email','$phone', '$hashed_password')";
          try{
              $result = mysqli_query($connection, $query);
              if($result){
                  echo"<script>alert('User registration successful')</script>";
                  header("Location: login.php");
                  exit;
              }else{
                  echo "<script>alert('User registration failed')</script>";
              }
          }catch(mysqli_sql_exception){
              echo "<script>alert('User registration failed')</script>";
          }
      }

    }else {
          echo "<script>alert('Please submit your data')</script>";
    }
  }
  mysqli_close($connection);
?>
```

This code snippet handles user registration and includes database interaction. Let's break it down line by line:

1. **alert('Please enter a valid data')";** - If the "fullname" validation fails, this line displays an alert message using JavaScript. Inside the else block of the fullname validation (assuming validation is successful):

12. **$hashed_password = password_hash($password, PASSWORD_DEFAULT);** - This line hashes the password using the **password_hash() function**. This is crucial for secure password storage. The function uses a strong hashing algorithm to create a unique, non-reversible value from the plain text password.

13. **$query = "INSERT INTO Patient (matricule,fullname,email,phone,user_password) VALUES ('$matricule','$fullname','$email','$phone', '$hashed_password')";** : This line constructs an SQL query to insert a new record into the "Patient" table. It includes the sanitized data and the hashed password. Inside a try block (for database interaction):

15. **$result = mysqli_query($connection, $query);** - This line attempts to execute the SQL query using mysqli_query() and the connection object ($connection). It stores the result in the $result variable.

16. **if($result){** - This line checks if the query execution was successful (i.e., $result is not false).

18. **header("Location: login.php");** - This line sends a header to redirect the user to the "login.php" page.

Conclusion:

Summary of Project's Achievements and Challenges:

<u>Achievements:</u>

- Efficient Appointment Scheduling: The implementation of the website has significantly streamlined the appointment scheduling process, reducing administrative burden and improving overall efficiency.
- Enhanced Patient Experience: Patients now enjoy a user-friendly interface that allows for seamless appointment booking, leading to reduced wait times and increased satisfaction.
- Streamlined Clinic Workflow: The website has facilitated better management of appointments, minimizing scheduling conflicts and automating crucial communication processes, thereby optimizing clinic operations.
- Improved Communication: With improved communication channels, patients receive timely reminders and updates, fostering better engagement and reducing missed appointments.

<u>Challenges:</u>

- **Data Security:** Ensuring the robust protection of patient data and maintaining compliance with stringent data protection regulations poses an ongoing challenge.
- **Scalability:** As the clinic expands and patient volumes increase, ensuring the scalability of the website to accommodate growth without compromising performance is crucial.
- **User Adoption:** Encouraging both patients and clinic staff to fully adopt and utilize the new system may require ongoing training and support to overcome resistance to change.

Impact and Benefits of the Website to the Target Audience:

- **Patients:** The website offers increased convenience and accessibility, empowering patients to take control of their healthcare journey and leading to higher satisfaction levels.
- **Clinic Staff:** By streamlining administrative tasks and improving communication, the website enhances staff efficiency, allowing them to focus more on providing quality patient care.
- **Healthcare Providers:** The website's implementation positions the clinic as a modern and patient-centric healthcare provider, leading to improved brand perception and patient trust.

Future Directions and Opportunities for Further Development:

- **Mobile Integration**: Developing a mobile application to complement the website would further enhance accessibility and engagement, catering to the increasing demand for mobile healthcare solutions.
- **Telemedicine Integration:** Incorporating telemedicine capabilities into the website would extend the reach of healthcare services, providing patients with greater flexibility and access to care.
- **Personalization Features:** Implementing personalized features based on patient preferences and behavior would enhance the user experience and foster stronger patient-provider relationships.
- **Continuous Improvement:** Regularly gathering feedback and iterating on the website's features and functionality will ensure it remains aligned with evolving patient needs and technological advancements, maintaining its relevance and effectiveness in the long term.