

# FastCARE

Team members:

Rodrigo Azevedo, Kelly Khalil

GitHub repository: [https://github.com/rodrigobivarazevedo/EHR\\_SYSTEM](https://github.com/rodrigobivarazevedo/EHR_SYSTEM)

## 1. Introduction

Taking a look at Electronic Health Records (EHR) takes us to a new and exciting world where the documentation, administration, and use of healthcare data are changing dramatically. The use of EHR systems is becoming increasingly important for patients, administrators, and healthcare professionals in the constantly changing healthcare sector. The purpose of this introduction is to provide a thorough understanding of Electronic Health Records (EHRs), including what they are, how they work in practice, their relevance, historical evolution, and practical applications.

### 1.1 What is EHR?

First, let's dispel the myths surrounding Electronic Health Records (EHR), which are sometimes referred to be the digital version of paper patient files. A whole patient's medical history, diagnosis, prescriptions, treatment plans, immunisation records, allergies, radiological images, and laboratory test results are all stored in an electronic health record (EHR). In essence, the goal of these records is to give a comprehensive, up-to-date picture of a patient's health in order to facilitate effective healthcare delivery and well-informed decision-making.

### 1.2 Importance of EHR:

Firstly, let us debunk some common misconceptions regarding Electronic Health Records (EHRs), which are commonly called the digital equivalent of paper patient records. An electronic health record (EHR) contains a patient's complete medical history, diagnoses, prescriptions, treatment plans, records of immunisations, allergies, radiographic pictures, and laboratory test results. These records essentially aim to provide a complete and current picture of a patient's health in order to support efficient healthcare delivery and informed decision-making.

Moreover, EHRs improve the effectiveness of healthcare systems by automating a number of procedures, cutting down on paperwork, and eliminating pointless effort. In addition to saving time, this allows medical staff to concentrate more on patient care and less on paperwork.

EHRs are essential for clinical research, public health programmes, and the general improvement of medical knowledge in a larger sense. EHR aggregated data may be anonymized and applied to epidemiological research, which advances our knowledge of and ability to treat a range of medical disorders.

### 1.3 Usage of EHR:

Electronic health records are used across the whole healthcare system. EHR systems allow healthcare providers—from doctors to nurses and allied health specialists—to actively enter and retrieve patient data. By using their own health information and actively participating in their healthcare process, patients also become active participants.

EHRs provide safe communication and data sharing across different healthcare organisations, allowing data to move seamlessly between clinics, hospitals, labs, and pharmacies. Interoperability is a key notion that guarantees information interchange between various EHR systems, promoting patient outcomes and continuity of treatment.

## 1.4 Development and History of EHR:

The digitalization of medical records in the second part of the 20th century is where Electronic Health Records have their origins. The first move to electronic databases from paper-based databases was made with the goal of managing and storing patient data more effectively. Early in the twenty-first century, the idea of electronic health records (EHRs) became popular due to a number of factors, including the need for a more accessible and integrated healthcare information system, growing knowledge of its advantages, and technical improvements. Globally, governments and healthcare institutions started allocating funds for the creation and deployment of standardised EHR systems.

EHR development is taking place in the areas of functionality, usability, and security. The progression from basic digital archives to advanced, cross-platform solutions signifies a dedication to boosting patient outcomes, augmenting healthcare provision, and stimulating creativity in the healthcare industry.

To sum up, electronic health records have cemented their place as essential elements of contemporary healthcare. They act as the pillars of effective, data-driven, patient-centered healthcare systems. Our investigation will reveal the characteristics, advantages, difficulties, and upcoming developments influencing the field of healthcare information management as we go further into the nuances of EHRs.

## 2. Website design and technical architecture

The EHR system consists of 9 main pages.

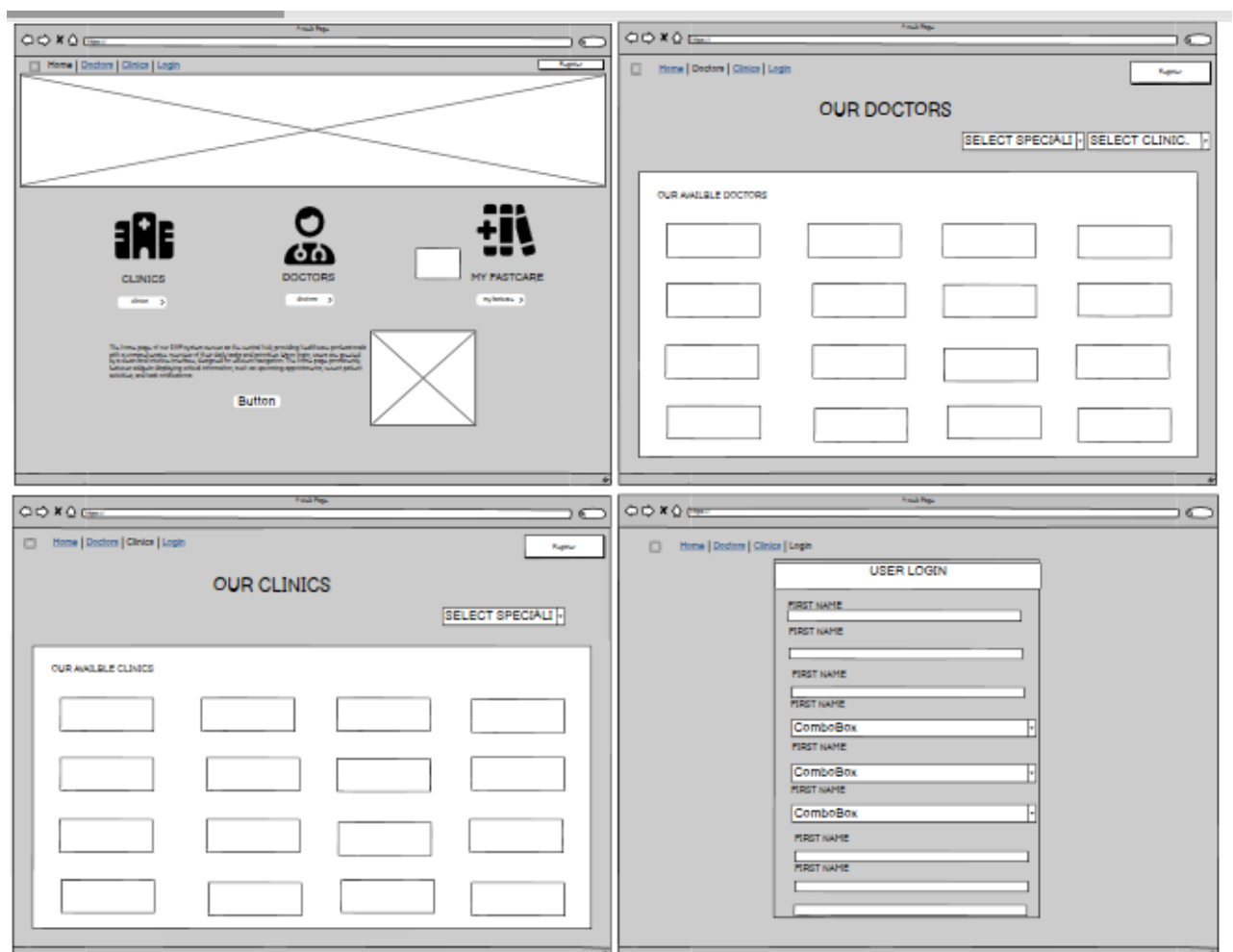
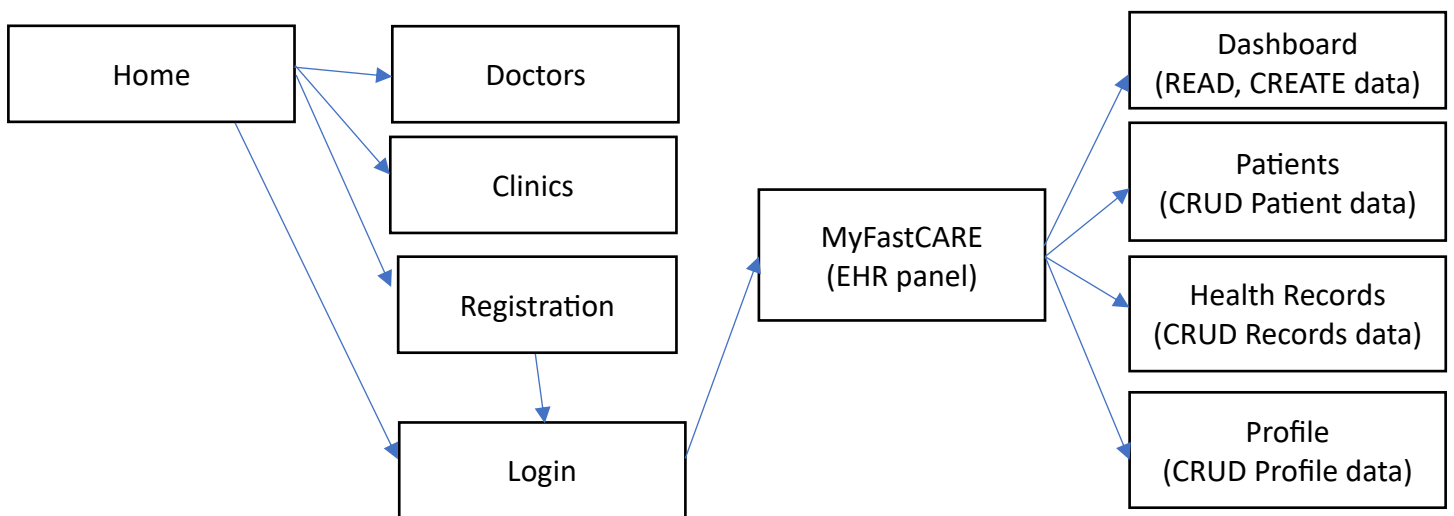
Before logging in the user can visit the following pages:

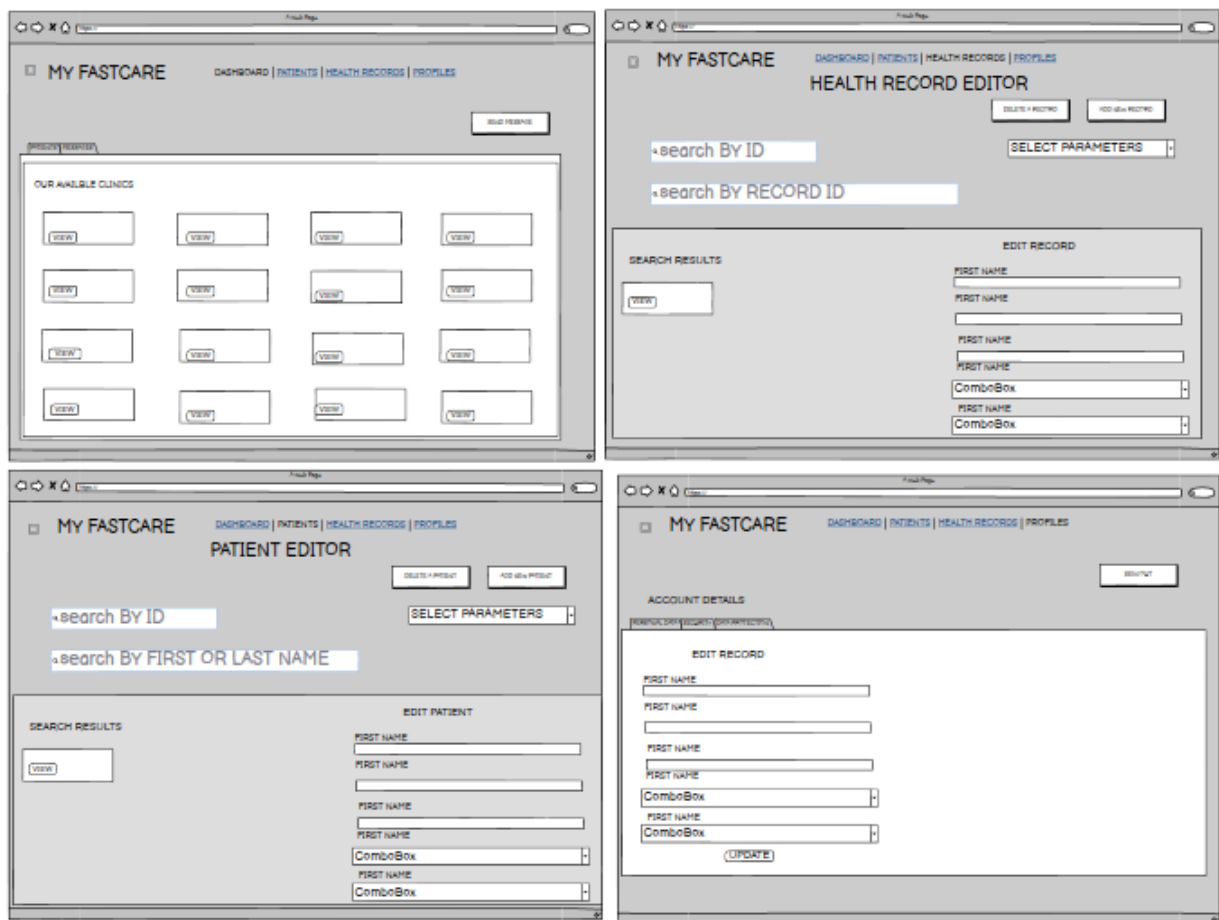
- **Home:** About us info and quick clickable buttons for redirecting the users to desired pages
- **Doctors:** Info about all the current registered doctors in the HER system
- **Clinics:** Info about all the clinics that use the system
- **Registration:** Page for a new doctor to register into the system
- **Login:** Page for existing doctors to log into the EHR operation panel (MyFastCARE)

Once a doctor is registered and logs in, he/she gets access to the EHR operation panel (MyFastCARE) and can perform CRUD operations on EHR data:

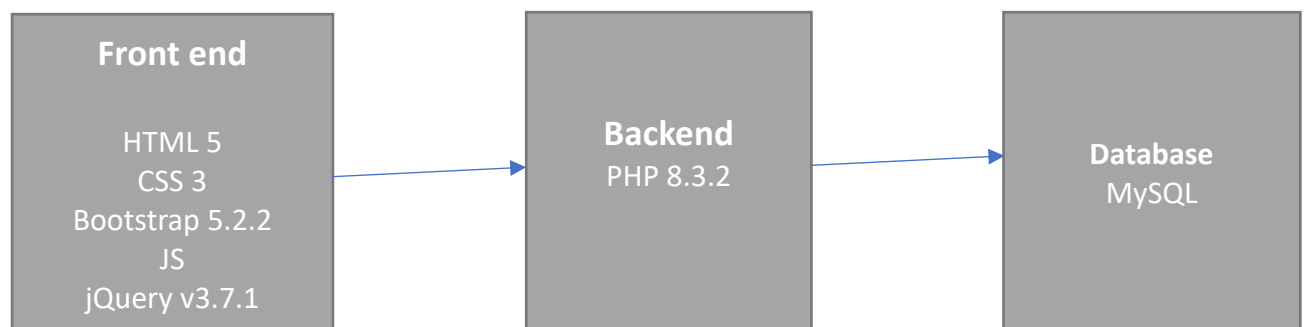
- **Dashboard:** The Doctor can view all his patients at a glance, view each patient's more detailed data, can also read messages from other doctors, and send messages to other doctors.
- **Patients:** This page is dedicated to being able to make full CRUD operations about patient's data
- **Health Records:** This page is dedicated to being able to make full CRUD operations about Health Records data.
- **Profile:** This page is dedicated to being able to make CRUD operations about the profile data of the respective doctor that is logged in

Inside the EHR operation panel the Doctor who logged in can only perform CRUD operations on his/her respective patients. Below is an illustration of the user flow with the System:





## 2.1 The Technical Stack



## 3. EHR information system

### 3.1 System Architecture

For our system we use the following structure for connecting front end to the backend:



### 3.2 Front end

All the .php files that correspond to the front end are all linked to their respective .js files with the same or similar name. Instead of only using PHP for connecting the frontend with the backend we decided to have JavaScript as an intermediate between the frontend and backend improving the responsiveness of the pages. Each time a certain type of information is requested or posted the correspondent JS code which the page is connected, performs an Ajax call to the correspondent AJAX.php file. Below is the format of the code used to perform the AJAX call in the login.js file:

```
function login() {
    var UsernameOrEmail = document.getElementById("UsernameOrEmail").value;
    var password = document.getElementById("password").value;

    $.ajax({
        url: "/EHR_system/ajax/loginAJAX.php",
        type: "POST",
        dataType: "json",
        data: { UsernameOrEmail: UsernameOrEmail, password: password, action:
"login"},
        success: function(response) {
            // Check if the login was successful
            if (response.UserID && response.success ==="Login successful") {
                // Redirect to patient_portal.php with UserID as a query parameter
                const redirectURL = `/EHR_system/ui/MyFastCARE/doctor_portal.php`;
                window.location.href = redirectURL;
            } else if (response.message) {
                // Handle unsuccessful login (e.g., display an error message)
                alert(response.message);
            }
        },
        error: function(xhr) {
            // Log detailed error information to the console
            console.log(xhr.responseText);

            // Display a user-friendly error message
            alert("Server request failed.");
        }
    });
}
```

### 3.3 Backend

For better consistency, less repetition, faster debugging, and security we followed an object-oriented approach for creating the backend logic. For each type of table on our database or type of data being used we created a PHP class, which all the classes have a reference to the main class of the system which is the Database class. The Database class simply holds the required logic for connecting to our MySQL database. In total we have 7 classes for specific types of data and as well our Database class for connection.

We have tried to use the same pattern for each class and their respective logic, the following are the general steps taken in the backend:

- Create an instance of the class Database for connecting the database and assigned it to the variable \$dbo
- Create an instance of the class to whom the type of action and data used belongs to and assigned it to the variable \$pdo
- Create and prepare the query statement.
- Bind the parameters to the query statement.
- Execute the query statement.
- Fetch the query results as a php associative array.

Each page and their .js file has a respective AJAX.php file, in the example above the JS code belongs to the login page which performs a call to the loginAJAX.php file. All the AJAX calls always have a specific action as a key in the data sent so that only the desired action is performed in the AJAX file. Also, the request type is checked for either GET or POST.

Now that the call has been made the following is a snippet of the loginAJAX.php file:  
(for login and registration only, POST was used for better security and privacy)

```
<?php
session_start(); // Start the session
$root = $_SERVER["DOCUMENT_ROOT"];
include_once $root . "/EHR_system/db/database.php";
include_once $root . "/EHR_system/db/backend.php";

$action = $_POST["action"];

if ($action === "login") {

    $UsernameOrEmail = isset($_POST["UsernameOrEmail"]) ? $_POST["UsernameOrEmail"]
: null;
    $password = isset($_POST["password"]) ? $_POST["password"] : null;

    if (empty($UsernameOrEmail) || empty($password)) {
        echo json_encode(["message" => "All required fields must be provided."]);
        exit();
    }

    $dbo = new Database();
    $pdo = new Users();
    $result = $pdo->login($dbo, $UsernameOrEmail, $password);

    // Check if the result is an error
    if (isset($result["error"])) {
        // Handle the error, for example, send an appropriate response to the
client
        echo $result;
        exit();
    }

    $data = json_decode($result, true);
```

```

// Check if the login was successful
if (isset($data['UserID'])) {
    // Store user information in session variables
    $_SESSION["UserID"] = $data['UserID'];
    $_SESSION["Username"] = $data["username"];
    $_SESSION["Role"] = $data["role"];
}

echo $result;
exit();
}

```

As you can see above since our action key was “login” the first if statement is triggered and two class instances are created. One of the database class and one of the Users class. Now that we created the two objects, we can directly call the public function called login from the class Users. Below is a snippet of how the backend logic looks like, in this case for the login function from the Users class:

```

public function login($dbo, $UsernameOrEmail, $password) {
    try {
        // Check if the username or email exists
        $checkUserStatement = $dbo->conn->prepare("SELECT * FROM users
WHERE Username = :UsernameOrEmail OR Email = :UsernameOrEmail");
        $checkUserStatement->bindParam(':UsernameOrEmail',
$UsernameOrEmail, PDO::PARAM_STR);
        $checkUserStatement->execute();
        $user = $checkUserStatement->fetch(PDO::FETCH_ASSOC);
        if ($user) {
            // User exists, now verify the password
            $hashed_password = $user['Password'];
            if (password_verify($password, $hashed_password)) {
                // Password is correct, user authenticated successfully
                return json_encode(["success" => "Login successful",
"UserID" => $user["UserID"], "username" => $user["Username"], "role" =>
$user["Role"]]);
            } else {
                // Incorrect password
                return json_encode(["message" => "Invalid login
credentials"]);
            }
        } else {
            // User not found
            return json_encode(["message" => "Invalid login credentials"]);
        }
    } catch (PDOException $e) {
        // Handle the exception (e.g., log, display an error message)
        return json_encode(["error" => $e->getMessage()]);
    }
}

```

All the responses from the backend follow the same structure for better consistency. All responses are JSON encoded, if the response contains a “message” key it means the request was not successful, otherwise if there is a “success” key or no “message” key then the request was successful.

This was an example of a POST request workflow from the login page. Almost all the pages from the EHR system are dynamic using JavaScript for dynamically populating each web page with the relevant data.

We used almost always the same structure and JS functions to show the response data from an AJAX call. Below is an example from the JS code for displaying the patients of a doctor in his dashboard:

```
function get_all_patients() {
    $.ajax({
        url: "/EHR_system/ajax/doctor_dashboardAJAX.php",
        type: "GET",
        dataType: "json",
        data: { action: "get_all_patients" },
        success: function(response) {
            if (response.message){
                document.getElementById('doctor_patients').textContent =
` ${response.message}`;
            }else{
                updateCardUI(response)
            }
        },
        error: function(xhr) {
            console.log(xhr.responseText);
            alert("Server request failed.");
        }
    });
}
```

The AJAX call is made and then from the response the frontend page is dynamically populated:

```
function updateCardUI(data) {
    // Clear existing cards
    const content = document.getElementById('patients');
    content.innerHTML = '';
    // Create and append new cards based on the data from the backend
    data.forEach(patient => {
        const card = `
            <div class="col">
                <div class="card shadow-sm">
                    <div class="card-body">
                        <h5 class="card-title">${patient.FirstName}
${patient.LastName}</h5>
                        <p class="card-text">ID: ${patient.PatientID}, Email:
${patient.Email}</p>
                        <p class="card-text">Contact Number:
${patient.ContactNumber}</p>
                    </div>
                </div>
            </div>
        `;
        content.innerHTML += card;
    });
}
```



```

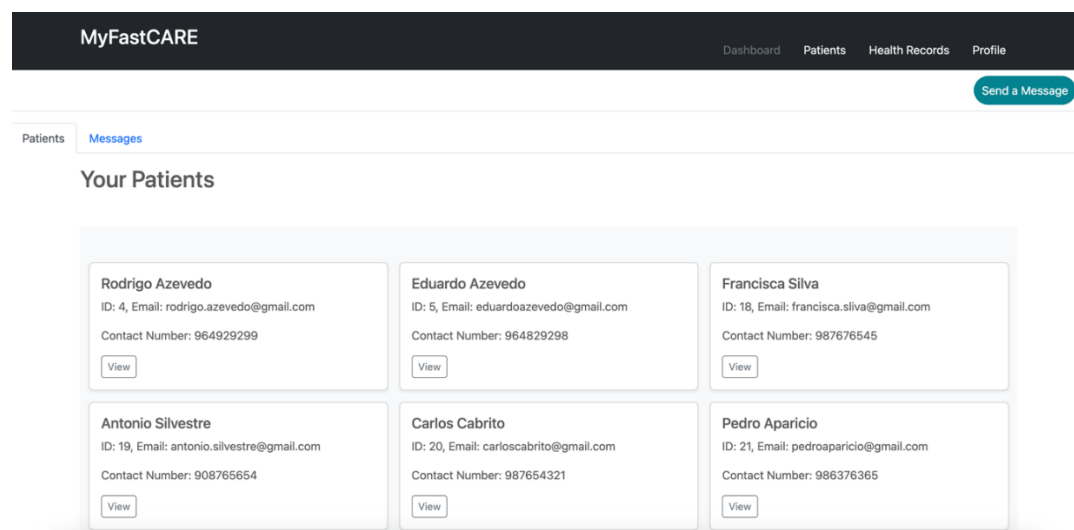
        <div class="d-flex justify-content-between align-items-
center">
            <div class="btn-group">
                <button type="button" class="btn btn-sm btn-outline-
secondary" id="view-edit-btn"
onclick="editPatient('${patient.PatientID}', '${patient.FirstName}',
 '${patient.LastName}', '${patient.Email}', '${patient.Birthdate}',
 '${patient.Gender}', '${patient.Address}', '${patient.ContactNumber}',
 '${patient.Smoker}')">View/Edit</button>
            </div>
        </div>
    </div>
</div>
</div>
</div>
    `;

    const cardElement = document.createElement('div');
    cardElement.innerHTML = card;
    content.appendChild(cardElement);
    });
}

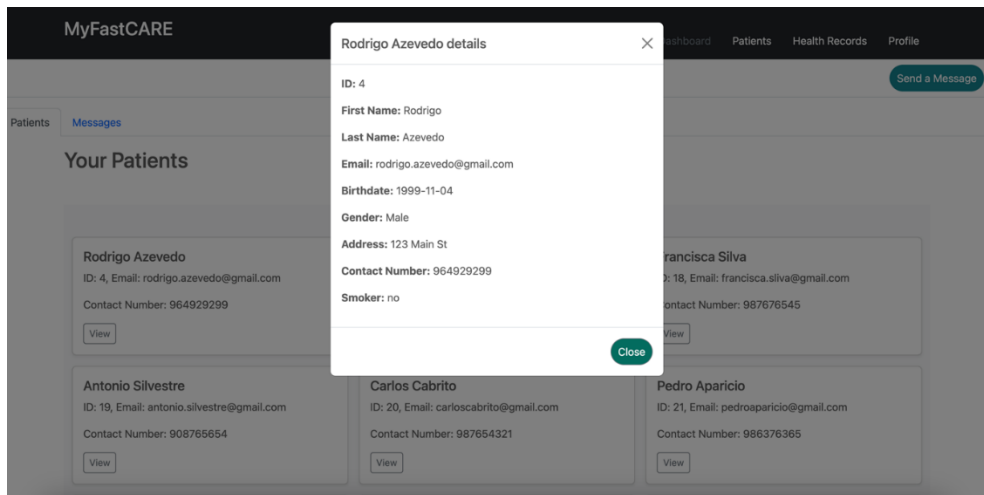
```

## 4. How to use the system

### 4.1 Dashboard



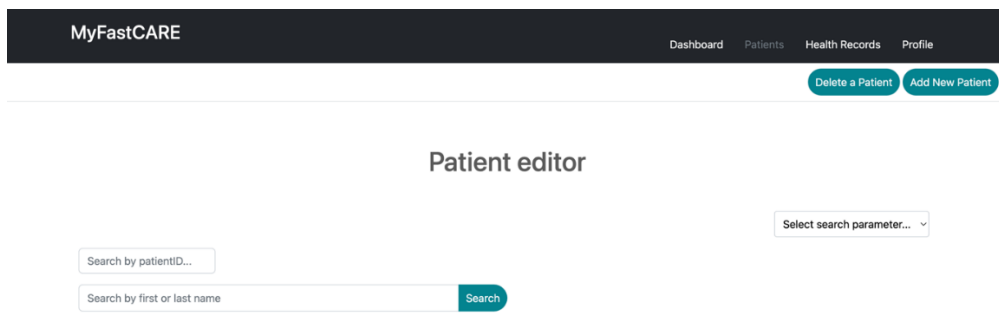
In the Dashboard the doctor can view all his/her patients at one glance, as well as check each patient individually by clicking the view button which will display the patient info in more detail.



Additionally, but not required for the project we decided it would be extra helpful if doctors could message each other through the platform, so we added a messages tab with all messages received and a button that will trigger a modal for messaging another doctor.

## 4.2 Patients page

### 4.2.1 Searching patient



In the first part of the patient's page, the doctor will see two forms for searching patients. We used two forms, the first one for quick and precise search with only patient ID, the second one where the doctor has several parameters, he/she can decide to search the patient: First or last name if not selected a parameter or full name, email, and contact number if selected a parameter from the dropdown. Doctors will only get search results from their list of patients.

## 4.2.2 Updating a patient

**Search Results**

Rodrigo Azevedo  
ID: 4, Email: rodrigo.azevedo@gmail.com  
Contact Number: 964929299  
[View/Edit](#)

**Edit Rodrigo Azevedo details**

Patient ID: 4

First Name: Rodrigo

Last Name: Azevedo

Patient Email: rodrigo.azevedo@gmail.com

Date of Birth: 04.11.1999

Gender: Male

Patient smokes? ☒ Yes ☐ No

Address: 123 Main St

Contact Number: 964929299

[Update Patient](#)

After getting the search results the doctor can click the “View/Edit” button to add all the selected patient info to the Edit form. After making changes simply press the update button and confirm to update the patient data. In the backend it is always checked if the patient in the form belongs to the doctor, getting access denied if the patient is not his/her.

## 4.2.3 Creating or Deleting Patient

For creating and deleting a patient the doctor just needs to press one of the two green buttons on top of the page and it will display a modal like such:

**Create a New Patient**

First Name: Enter patient first name

Last Name: Enter patient last name

Patient Email: Enter patient email

Date of Birth: dd.mm.yyyy

Gender: ☐ Male ☐ Female ☐ Other

Patient smokes? ☐ Yes ☐ No

Address: Enter patient address

Contact Number: Enter contact number

[Close](#) [Create Patient](#)

**Delete a Patient**

Patient ID: Enter patient ID

[Close](#) [Delete Patient](#)

After inserting all the correct data with no forms missing the doctor just needs to press either create patient or delete patient. In both cases, the doctors are always prompted with a confirmation submission to be sure he/she would like to perform the desired action.

Also in the update action, doctors can only delete his/her patients, getting a “patient not yours” message if it’s not their patient. Also, if for some reason someone gets access to the doctor’s platform it is always checked if the user is a doctor or not, getting access denied if it’s not a doctor. Below is the code for this:

```
session_start();
$UserID = $_SESSION["UserID"];

$stmt = $db->conn->prepare(
    "SELECT DoctorID FROM doctors WHERE UserID = :UserID"
);
$stmt->bindParam(':UserID', $UserID, PDO::PARAM_INT);
$stmt->execute();

$Doctor = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$Doctor) {
    echo json_encode(["message" => "Access denied"]);
    exit(); // Terminate script execution after sending the response
}
```

Also, the code for checking if the patient is from the doctor:

```
$stmt = $db->conn->prepare(
    "SELECT DoctorID FROM patients WHERE PatientID = :patientID"
);
$stmt->bindParam(':patientID', $PatientID, PDO::PARAM_INT);
$stmt->execute();

$Patient_doctor = $stmt->fetch(PDO::FETCH_ASSOC);
$Patient_doctorID = $Patient_doctor["DoctorID"];
if ($DoctorID !== $Patient_doctorID) {
    echo json_encode(["message" => "Patient is not yours"]);
    exit(); // Terminate script execution after sending the response
}
```

Each web page in the doctor’s platform is protected checking for if a SESSION “UserID” key exists. Getting redirected to the login page if someone tries to reach the URL manually:

```
<?php
session_start();
// Check if the user is not logged in
if (!isset($_SESSION['UserID'])) {
    // Redirect to the login page
    header('Location: /EHR_SYSTEM/ui/MyFastCARE/login.php');
    exit(); // Make sure to stop execution after the redirect
}
?>
```

## 4.3 Health Records page

### 4.3.1 Searching Health records

The screenshot shows the 'MyFastCARE' application interface. At the top, there is a dark navigation bar with the logo 'MyFastCARE' on the left and links for 'Dashboard', 'Patients', 'Health Records', and 'Profile' on the right. Below the navigation bar, there are two buttons: 'Delete Record' and 'Add New Record'. The main content area is titled 'HealthRecords editor'. Below the title, there are two search input fields: 'Search by patientID:...' and 'Search by RecordID:...'. The background is a light gray color.

For the health records page, a doctor can find a health record by patient ID or by record ID. Doctors will only get search results from their list of patients.

### 4.3.2 Updating Health records

The screenshot shows the 'MyFastCARE' application interface. On the left, there is a 'Search Results' section with four cards. Each card displays the date, ID, patient ID, and diagnosis, along with a 'View/Edit' button. The cards are:

- Date: 2022-01-01, ID: 1, PatientID: 4, Diagnosis: Pain in shoulder
- Date: 2022-02-05, ID: 2, PatientID: 4, Diagnosis: Pain in arm
- Date: 2022-03-10, ID: 3, PatientID: 4, Diagnosis: Condition 3
- Date: 2022-04-15, ID: 4, PatientID: 4, Diagnosis: Condition 4

On the right, there is an 'Edit RecordID: 1 from PatientID: 4' form. The form contains the following fields:

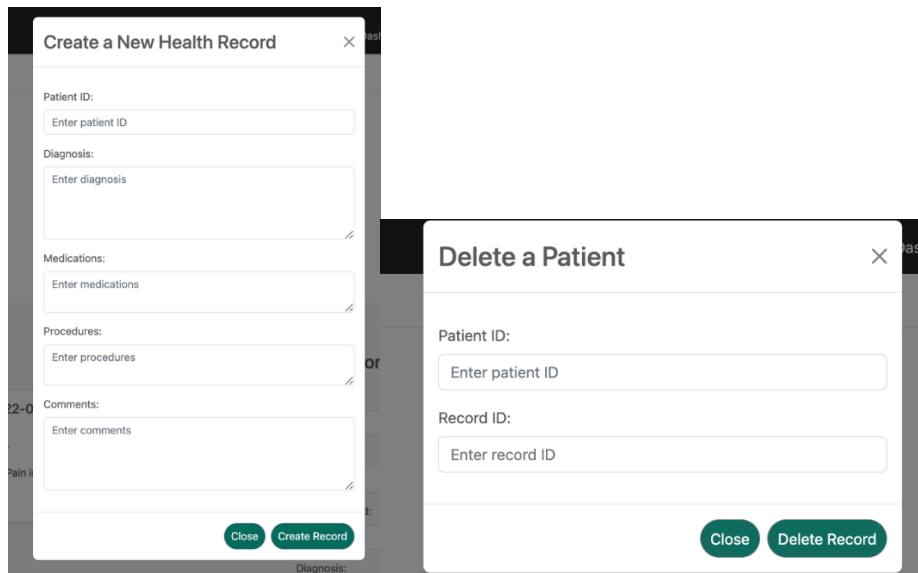
- Record ID: 1
- Patient ID: 4
- Date Recorded: 01.01.2022
- Diagnosis: Pain in shoulder
- Medications: Medication 1
- Procedures: Procedure 1
- Comments: Comments 1

At the bottom of the form, there is an 'Update Record' button.

After getting the search results for the health records of a patient the doctor can click on the “View/Edit” button to add all the selected health record info into the Edit form. After making changes simply press the update button and confirm to update the health record data. Like in the patient’s page, in the backend it is always checked if the patient in the form belongs to the doctor, getting a “patient not yours” message if the patient is not his/her.

### 4.3.3 Creating or Deleting a health record

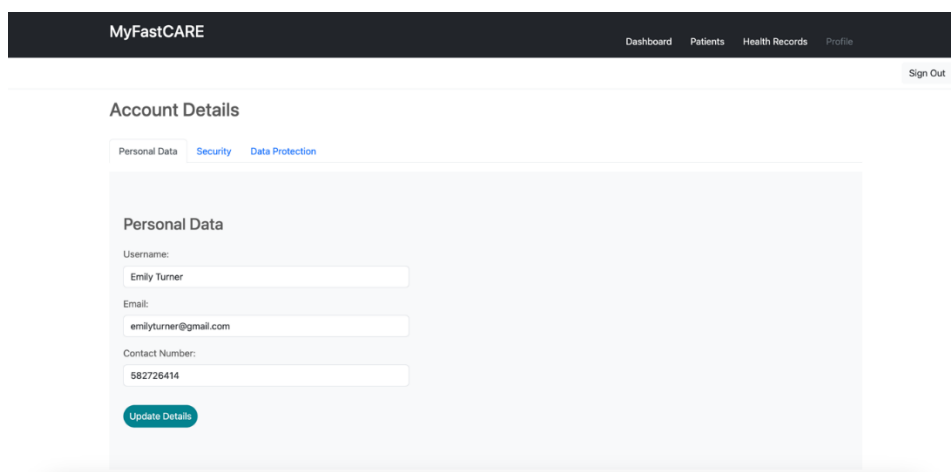
Like the patient's page to create a health record or delete a health record the doctor simply needs to press one of the two green buttons on top of the page. After clicking, a modal will appear where the doctor must insert all the required data. After pressing either delete or create the doctor must confirm that he/she would like to perform the desired action. Similar to the patient's page, the doctor can only delete health records from his/her patients, getting a "patient not yours" message if he/she tries to delete a record from another doctor's patient.



The image shows two overlapping modal windows. The left modal, titled "Create a New Health Record", contains five text input fields: "Patient ID:", "Diagnosis:", "Medications:", "Procedures:", and "Comments:". Each field has a placeholder text "Enter [field name]". At the bottom right of this modal are two green buttons: "Close" and "Create Record". The right modal, titled "Delete a Patient", contains two text input fields: "Patient ID:" and "Record ID:", both with placeholder text "Enter [field name]". At the bottom right of this modal are two green buttons: "Close" and "Delete Record".

## 4.4 Profile Page

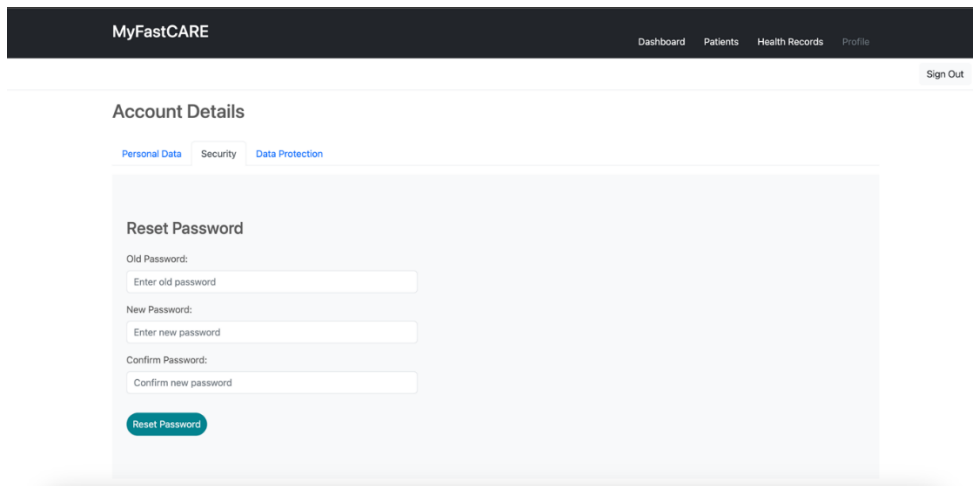
### 4.4.1 Updating profile data



The image shows a web application interface for "MyFastCARE". At the top is a dark navigation bar with the logo "MyFastCARE" on the left and links for "Dashboard", "Patients", "Health Records", and "Profile" on the right. A "Sign Out" link is located in the top right corner. Below the navigation bar is the "Account Details" section. It has three tabs: "Personal Data" (selected), "Security", and "Data Protection". The "Personal Data" tab is active, showing a form with the following fields: "Username:" with the value "Emily Turner", "Email:" with the value "emilyturner@gmail.com", and "Contact Number:" with the value "582726414". At the bottom of the form is a green "Update Details" button.

On the profile page, the doctor can update his/her profile details in the Personal Data tab. Simply change the desired info for the update and click the update button.

#### 4.4.2 Updating password



The screenshot shows the 'MyFastCARE' web application interface. At the top, there is a dark navigation bar with the logo 'MyFastCARE' on the left and links for 'Dashboard', 'Patients', 'Health Records', and 'Profile' on the right. A 'Sign Out' button is located in the top right corner. Below the navigation bar, the 'Account Details' section is visible, with tabs for 'Personal Data', 'Security', and 'Data Protection'. The 'Security' tab is active, displaying the 'Reset Password' form. This form includes three input fields: 'Old Password:' (with placeholder 'Enter old password'), 'New Password:' (with placeholder 'Enter new password'), and 'Confirm Password:' (with placeholder 'Confirm new password'). A teal 'Reset Password' button is positioned at the bottom of the form.

The doctor can also update his/her password. When resetting the password, the old one will be checked if it is valid allowing then the doctor to reset the password. The Data Protection tab simply contains privacy terms links. If the doctor wishes to log out, he/she can simply press the sign-out button which will destroy the current session. The code for that is below:

```
<?php
session_start();
// Unset all session variables
$_SESSION = array();
// Destroy the session
session_destroy();
// Redirect to the login page or any other desired page
header("Location: /EHR_system/ui/MyFastCARE/login.php");
exit();
?>
```

#### 4.5 Extra functionalities of the system:

- Registration email confirmation and reset password with email using PHPMailer
- Strong password suggestion using JS

```
function generateStrongPassword() {
    const length = 8; // password length
    const uppercaseChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    const lowercaseChars = "abcdefghijklmnopqrstuvwxyz";
    const numberChars = "0123456789";
    const specialChars = "!@#$%^&*()_-=+<>?";

    const getRandomChar = (charset) => {
        const randomIndex = Math.floor(Math.random() * charset.length);
        return charset.charAt(randomIndex);
    };
```

```
// Ensure at least one character from each character set
let password =
  getRandomChar(uppercaseChars) +
  getRandomChar(lowercaseChars) +
  getRandomChar(numberChars) +
  getRandomChar(specialChars);

for (let i = password.length; i < length; i++) {
  const allChars = uppercaseChars + lowercaseChars + numberChars +
specialChars;
  password += getRandomChar(allChars);
}

// Shuffle the characters in the password
const shuffledPassword = password.split('').sort(() => Math.random() -
0.5).join('');
return shuffledPassword;
}
```



## 5. Database

For our database we used 10 tables. The following is a short description of each table:

### **HealthRecords Table:**

- Relation to patients table: Many-to-One (Many health records can be associated with one patient, but one patient can have many health records).
- Relation to Doctors table: Many-to-One (Many health records can be associated with one doctor, but one doctor can have many health records).

### **Patients Table:**

- Relation to Doctors table: Many-to-One (Many patients can be associated with one doctor, but one doctor can have many patients).

### **Users Table:**

- No direct relationships. It serves as a base table for Doctors and potentially other user roles.

### **Doctors Table:**

- Relation to Users table: One-to-One (Each doctor is associated with one user).

### **Clinics Table:**

- Relation to ClinicSpecialities table: One-to-Many (One clinic can have multiple specialities, but each speciality belongs to one clinic).
- Relation to DoctorClinic table: One-to-Many (One clinic can be associated with multiple doctors, but each doctor is associated with one clinic).

### **ClinicSpecialities Table:**

- Relation to Clinics table: Many-to-One (Many specialities can belong to one clinic, but one speciality can only belong to one clinic).
- Relation to Specialities table: Many-to-One (Many clinic specialities can be associated with one speciality, but one speciality can have many clinic specialities).

### **Specialities Table:**

- No direct relationships. It serves as a lookup table for clinic specialities.

### **DoctorClinic Table:**

- Relation to Doctors table: Many-to-One (Many doctors can be associated with one clinic, but one clinic can have many doctors).
- Relation to Clinics table: Many-to-One (Many clinics can be associated with one doctor, but one doctor can be associated with many clinics).

### **Messages Table:**

- Relation to Users table (SenderID and ReceiverID): Many-to-One (Many messages can be associated with one sender or one receiver, but one sender or receiver can have many messages).

## EHR Information System

SUS - A quick and dirty usability scale for EHR system made for doctors.

kellykhalil048@gmail.com
 [Konto wechseln](#)

Nicht freigeben

1. Using this EHR system enhances my efficiency in patient care.

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

2. The complexity of the EHR system negatively impacts my workflow.

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

3. I find it easy to locate and update patient information within the EHR system

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

4. I am not confident in using the EHR system without the need for technical support. \*

Beschreibung

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

5. The integration of functions in the EHR system supports effective patient care coordination. \*

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

6. Inconsistencies in the EHR system complicate my use and understanding of patient data. \*

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

7. I believe most doctors would quickly adapt to and benefit from using this EHR system.

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

8. The EHR system's interface is cumbersome, particularly when accessing specific patient records

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

9. I feel confident in making medical decisions based on information from the EHR system.

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

10. Learning to effectively use this EHR system required a substantial investment of time and effort

Strongly disagree      1      2      3      4      5      Strongly agree

☐   ☐   ☐   ☐   ☐

Senden

Alle Eingaben löschen

## 6. Evaluation of the EHR information system:

Link for the survey:

[https://docs.google.com/forms/d/e/1FAIpQLSdWWXPcyphHdKHaekTPYJf8t4OEdqYI7UTzT7ulr\\_X6nL-quQ/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSdWWXPcyphHdKHaekTPYJf8t4OEdqYI7UTzT7ulr_X6nL-quQ/viewform?usp=sf_link)

The System Usability Scale (SUS) questionnaire was administered to assess the usability and user satisfaction of our Electronic Health Records (EHR) system. The questionnaire aimed to gather insights into the users' experiences, efficiency, and overall satisfaction with the EHR system.

### Methodology:

The SUS questionnaire consisted of ten statements, each scored on a scale from 1 to 5, ranging from Strongly Disagree to Strongly Agree. The respondents were healthcare professionals actively using the EHR system in their daily workflows.

### SUS Score Calculation:

#### MAIN RESULTS:

1)5 - 2)1 - 3)5 - 4)5 - 5)5 - 6)1 - 7)5 - 8)1 - 9)5 - 10)1

#### Sum of Adjusted Scores:

$(5-1=4), (5-1=4), (5-1=4), (5-5=0), (5-1=4), (5-1=4),$   
 $(5-1=4), (5-1=4), (5-1=4), (5-1=4)$

Total score = 36

Final SUS Score:

$36 * 2.5 = 90$

### Results:

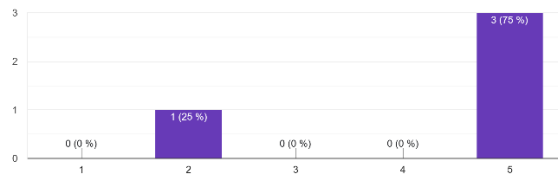
Users overwhelmingly found the EHR system easy to navigate, praising its user-friendly interface. Satisfaction scores reflected users' high regard for the system's functionality, performance, and positive impact on overall satisfaction. Confidence when starting was not great and would need improvement in the future.

### Interpretation:

The SUS score indicates the perceived usability of the EHR system. A score above 72.5 suggests above-average usability, while scores below 72.5 may indicate areas for improvement. Thus, the system can be considered above average in terms of usability.

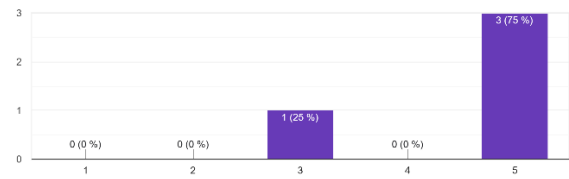
1. Using this EHR system enhances my efficiency in patient care.

4 Antworten



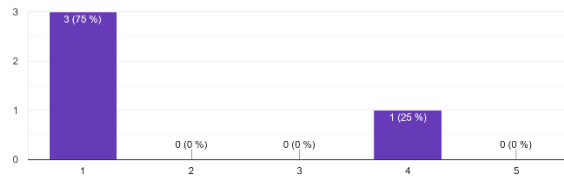
7. I believe most doctors would quickly adapt to and benefit from using this EHR system.

4 Antworten



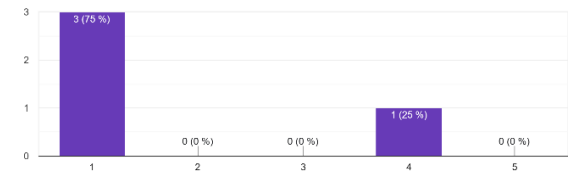
2. The complexity of the EHR system negatively impacts my workflow.

4 Antworten



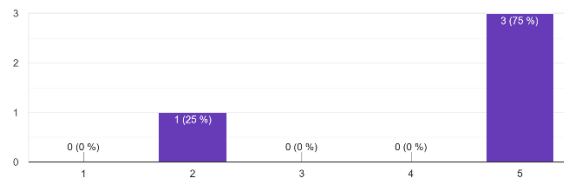
8. The EHR system's interface is cumbersome, particularly when accessing specific patient records

4 Antworten



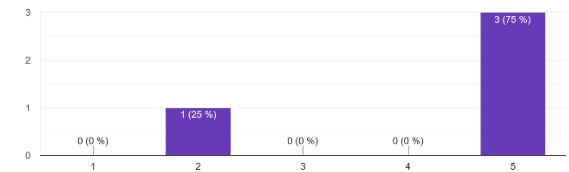
3. I find it easy to locate and update patient information within the EHR system

4 Antworten



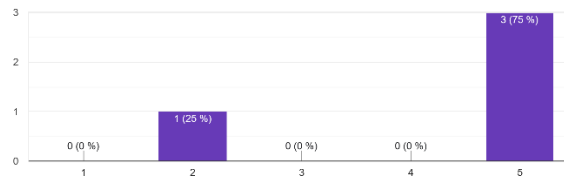
9. I feel confident in making medical decisions based on information from the EHR system.

4 Antworten



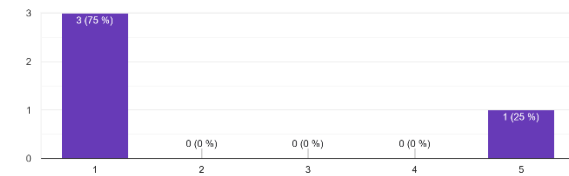
4. I am not confident in using the EHR system without the need for technical support.

4 Antworten



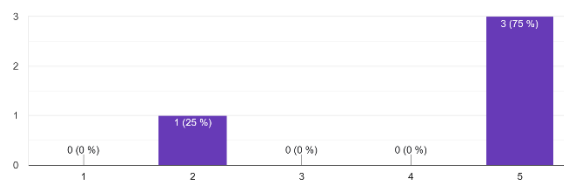
10. Learning to effectively use this EHR system required a substantial investment of time and effort

4 Antworten



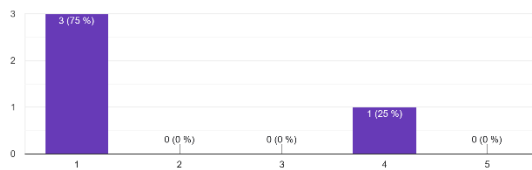
5. The integration of functions in the EHR system supports effective patient care coordination.

4 Antworten



6. Inconsistencies in the EHR system complicate my use and understanding of patient data.

4 Antworten



## 7. Conclusion

The EHR system has two main parts: The landing website where information about the system is displayed and registration for a new doctor is possible. The EHR operation panel where when logged in the doctor can perform CRUD operations on his patients EHR data. The full system is dynamic with responsive webpages and secured using several methods for preventing SQL injection attacks, unauthorized access to certain pages and actions on the database.

Tasks performed by each team member:

### **Kelly Khalil**

- Front end: HTML and CSS for all webpages
- Front end: Javascript code for account.js, clinics.js, doctors.js
- Backend AJAX: PHP code for accountAJAX.php, clinicsAJAX.php, doctorsAJAX.php, signout.PHP
- Backend: PHP code for the classes Doctorsinfo, Clinicsinfo, All\_Info, Messages
- Report: SUS, website design, introduction, How to use the system

### **Rodrigo Azevedo**

- Front end: JS code for doctor\_dashboard.js, doctor\_patients.js, doctor\_health\_records.js, login.js
- Backend AJAX: PHP code for doctor\_dashboardAJAX.php, doctor\_patientsAJAX.php, health\_recordsAJAX.php, loginAJAX.php.
- Backend: PHP code for the classes Users, Patients, Records
- Reset password and Registration Emailing functionalities and pages.
- Database design and structure
- Report: Technical architecture and description of EHR system code