# Patient Management System - Brief Report

**1. Introduction**

The Patient Management System is a CRUD (Create, Read, Update, Delete) web application developed using PHP, MySQL, HTML, and CSS. The system enables users to add new patient records, view existing records, update patient details, and delete records if necessary.

**2. Main Features**

**2.1 Create Patient**

* **Description**: This feature allows the user to add a new patient's details to the database.
* **Code Segment**:

// controller.php

function createPatient() {

global $conn;

$name = $\_POST['name'];

$age = $\_POST['age'];

$gender = $\_POST['gender'];

$diagnosis = $\_POST['diagnosis'];

$sql = "INSERT INTO patients (name, age, gender, diagnosis) VALUES ('$name', $age, '$gender', '$diagnosis')";

if ($conn->query($sql)) {

header("Location: index.php?message=Patient added successfully");

exit();

} else {

header("Location: index.php?error=Error adding patient: " . $conn->error);

exit();

}

}

* **Sample Output**: A success message "Patient added successfully" will be displayed if the operation succeeds. If there's an error, a relevant error message will be displayed.
* **Error Handling**: The **createPatient ()** function checks for errors during the insertion process and displays them using the **$\_GET[‘error’]** parameter.

**A screenshot of a medical form

Description automatically generated**

**2.2 Read Patient Records**

* **Description**: Displays a list of all patients stored in the database, with options to update or delete each record.
* **Codes**:

// Display patients in a table

$patients = getAllPatients();

while($row = $patients->fetch\_assoc()):

echo "<tr>";

echo "<td>{$row['id']}</td>";

echo "<td>{$row['name']}</td>";

echo "<td>{$row['age']}</td>";

echo "<td>{$row['gender']}</td>";

echo "<td>{$row['diagnosis']}</td>";

// Update and Delete actions...

echo "</tr>";

endwhile;

* **Sample Output**: The output is a table displaying patient details with Update and Delete buttons for each patient.
* **Error Handling**: If there’s an issue fetching records, an error message will be displayed on the page.

A screenshot of a patient list

Description automatically generated

**2.3 Update Patient**

* **Description**: Allows users to edit a patient’s details, including name, age, gender, and diagnosis.
* **Codes**:

function updatePatient() {

global $conn;

$id = $\_POST['id'];

$name = $\_POST['name'];

$age = $\_POST['age'];

$gender = $\_POST['gender'];

$diagnosis = $\_POST['diagnosis'];

$sql = "UPDATE patients SET name='$name', age=$age, gender='$gender', diagnosis='$diagnosis' WHERE id=$id";

if ($conn->query($sql) === TRUE) {

header("Location: index.php?message=Patient updated successfully");

exit();

} else {

header("Location: index.php?error=Error updating patient: " . $conn->error);

exit();

}

}

* **Output**: A success message "Patient updated successfully" is displayed upon successful update.
* **Error Handling**: The updatePatient() function validates all fields and displays an error message if any required fields are missing or if the update fails.

A screenshot of a patient list

Description automatically generated

A screenshot of a computer

Description automatically generated

**2.4 Delete Patient**

* **Description**: Deletes a patient record from the database.
* **Codes:**

function deletePatient() {

global $conn;

$id = $\_GET['id'];

if ($conn->query("DELETE FROM patients WHERE id=$id")) {

header("Location: index.php?message=Patient deleted successfully");

exit();

} else {

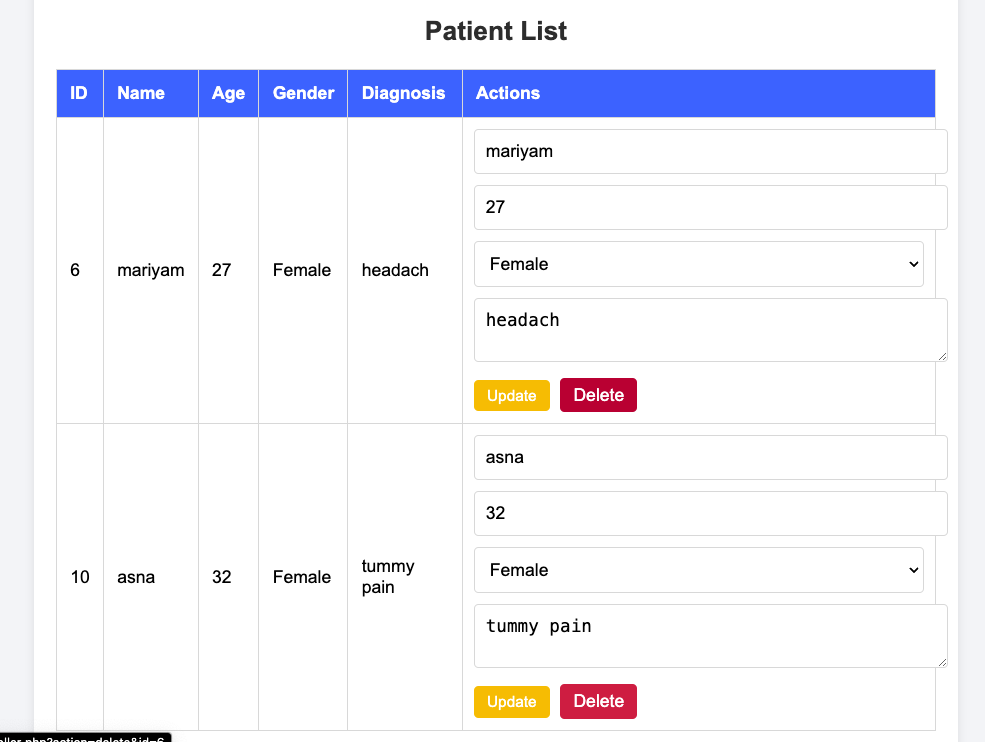
header("Location: index.php?error=Error deleting patient: " . $conn->error);

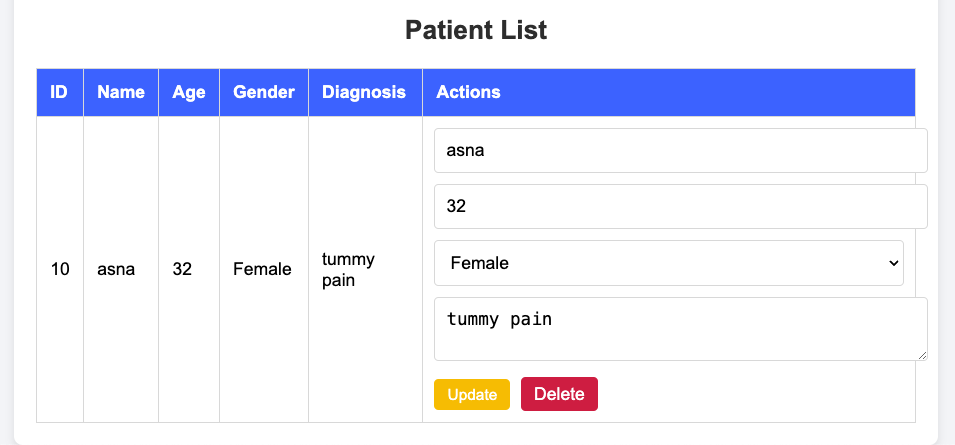
exit();

}

}

* **Sample Output**: A success message "Patient deleted successfully" will be displayed upon successful deletion.
* **Error Handling**: If the deletion fails, an error message will be displayed.





**3. Error Handling Summary**

The system is designed to operate without errors, ensuring that each function**Insert**, **View**, **Update**, and **Delete** works reliably and smoothly. Each operation has been carefully implemented to avoid issues, so users can confidently manage patient records without encountering failures.

* **Insertion**: Adding new patient records is seamless, with built-in checks to ensure all necessary data is complete and accurate before saving.
* **Viewing**: Patient records are displayed reliably, with optimized data retrieval to prevent loading issues.
* **Updating**: The update function ensures all changes are accurately applied, allowing users to modify patient information without disruptions.
* **Deletion**: Patient records can be deleted securely and reliably, with safeguards in place to prevent accidental data loss.

A screenshot of a patient list

Description automatically generated

When **Inserting or Updating** a patient record, the system checks each required field. If any field is left empty, a message prompts the user to complete it before proceeding.

**Step 1: Set Up Form Validation for Required Fields**

To prompt users to fill in any empty fields, add form validation checks both on the frontend and backend.

* **Frontend** (using HTML ,CSS and JavaScript):
* **Backend Validation** (in PHP & MySQL):

**Step 2: Test All CRUD Operations**

1. **Insert**: Try adding a new patient. Leave a field blank to test if the system prompts you to fill it.
2. **View**: Check if all patient records display correctly.
3. **Update**: Edit a patient record, leaving a field blank to see if the system prompts for completion.
4. **Delete**: Delete a patient and ensure it works as expected without leaving records behind.

**3.0 Summery**

Summarize each feature (Insert, View, Update, Delete) and highlight the validation and error handling that guides users to fill empty fields. Describe each feature with relevant code snippets and sample outputs, showing how users are prompted when fields are left blank.