ABSTRACT

The AJP Hospital project is a web-based application that provides a platform for managing various aspects of a hospital, including receptionist, doctor, and patient/appointment information. The project is designed to simplify administrative tasks, improve communication, and streamline the hospital's operations.

The system consists of multiple web pages that are interconnected and allow different users (admin, receptionist, and doctor) to perform specific tasks based on their roles. The home page serves as the entry point, providing links to the admin, receptionist, and doctor pages.

The admin page provides options to manage receptionist, doctor, and patient/appointment details. The receptionist management section allows adding, deleting, updating, and viewing receptionist information. Similarly, the doctor management section offers the same functionalities for managing doctor details. The patient/appointment management section allows adding, deleting, updating, and viewing patient and appointment information.

The receptionist page provides access to patient/appointment management functionalities. Receptionists can add new patients/appointments, update existing ones, delete appointments, or view a list of all appointments/patient details.

The doctor page allows doctors to view their assigned patient/appointment details. Doctors can access the view functionality to see the list of patients/appointments assigned to them.

Each functionality is implemented using Java Servlets and JDBC for database connectivity. The MySQL database is used to store and retrieve data related to receptionists, doctors, and patients/appointments.

The project aims to enhance the efficiency and organization of hospital operations by providing a user-friendly interface for managing crucial information. It offers a centralized platform for different roles to perform their tasks effectively, facilitating better coordination and communication within the hospital.

PROBLEM STATEMENT

The AJP Hospital faces various challenges in managing its operations efficiently and effectively. The existing manual processes and lack of a centralized system lead to several issues, including miscommunication, delays, and errors in managing patient and appointment information. The hospital staff, including receptionists and doctors, struggle with manual paperwork and outdated methods for handling administrative tasks.

1. Inefficient Administrative Tasks: The hospital's administrative tasks, such as managing patient records, appointments, and doctor schedules, are time-consuming and prone to errors. The lack of a streamlined system hinders the ability to handle a large volume of patients efficiently.

2. Communication and Coordination Issues: Without a centralized platform, communication between receptionists, doctors, and other staff members is challenging. This often results in miscommunication, leading to confusion and delays in providing timely healthcare services.

3. Difficulty in Accessing Information: The absence of a comprehensive database makes it difficult for receptionists and doctors to access patient information quickly. This can impact the quality of patient care and result in delays in diagnosis or treatment.

4. Ineffective Appointment Management: The current appointment management system relies heavily on manual processes. This approach is prone to errors, double-bookings, and inefficiencies in scheduling appointments.

5. Lack of Data Analysis and Reporting: The hospital lacks a system that can generate meaningful reports and analyze data. This hinders the ability to make informed decisions and improve overall operational efficiency.

To address these challenges, the AJP Hospital requires a comprehensive web-based application that centralizes and automates various administrative tasks, enhances communication and coordination among staff members, provides easy access to patient information, streamlines appointment management, and enables data analysis and reporting. Such a solution will significantly improve the hospital's efficiency, accuracy, and patient care quality.

SYSTEM REQUIREMENTS

Language Used	:	HTML & Java Programming Language
IDE Tool	:	Eclipse
Server	:	Tomcat Server Version 9.0
Database	:	MySQL
Web Browser	:	Google Chrome

HTML & Java Programming Language

HTML (Hypertext Markup Language) is the standard markup language for creating web pages. It provides the structure and content of a web page by using various elements and tags. HTML is used to define the layout, headings, paragraphs, images, links, and other components of a webpage. It is a client-side language, meaning it runs on the user's web browser.

Java is a versatile and widely used programming language that allows developers to build robust and scalable applications. It is known for its platform independence, as Java programs can run on any operating system that has a Java Virtual Machine (JVM). Java is commonly used for server-side programming, and it provides powerful libraries and frameworks for web development.

Eclipse

Eclipse is a popular integrated development environment (IDE) that provides a comprehensive set of tools for Java development. It offers features such as code editing, debugging, testing, and deployment capabilities. Eclipse has a user-friendly interface and supports plugins for various programming languages, making it a preferred choice for many developers.

Tomcat

Apache Tomcat is an open-source web server and servlet container that is widely used for hosting Java-based web applications. It provides an environment for running Java Servlets, JavaServer Pages (JSP), and other Java web technologies. Tomcat is known for its simplicity, lightweight nature, and compatibility with different operating systems. Version 9.0 is one of the latest stable releases of Tomcat.

MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is widely used in web development. It provides a reliable and scalable platform for storing and managing structured data. MySQL supports standard SQL queries and offers various features like transactions, indexing, and replication. It is a preferred choice for many web applications due to its performance, ease of use, and community support.

Google Chrome

Google Chrome is a widely used web browser developed by Google. It offers a fast and secure browsing experience and supports modern web technologies. Chrome is known for its compatibility with HTML, CSS, and JavaScript, making it an ideal choice for testing and running web applications. It provides developer tools for debugging and analyzing web pages, making it a preferred browser for web developers.

Using HTML and Java programming language, along with Eclipse as the IDE, Tomcat Server version 9.0 for hosting the application, MySQL as the database, and Google Chrome as the web browser, provides a robust and reliable foundation for developing a web-based application for AJP Hospital. These technologies and tools are well-established, widely used, and offer extensive community support, ensuring a smooth development process and efficient execution of the application.

MODULE DESCRIPTION

ADMIN MODULE:

The admin module provides administrative functionalities for managing the hospital system. It allows the admin to create and manage user accounts, including receptionists, doctors, and other staff members. The module enables the admin to set access privileges, manage system configurations, and monitor system activities. It provides a user-friendly interface for the admin to perform administrative tasks efficiently.

RECEPTIONIST MODULE:

The receptionist module is designed for the receptionists who handle front desk operations. It includes features such as patient registration, appointment scheduling, and patient inquiries. The module enables receptionists to efficiently manage patient appointments, update patient information, and handle patient queries. It ensures smooth operations at the reception desk and effective coordination with other modules.

DOCTOR MODULE:

The doctor module caters to the needs of doctors within the hospital. It provides functionalities for doctors to access patient information, view medical records, and manage appointments. The module allows doctors to update patient diagnoses, prescribe medications, and request medical tests. It facilitates seamless communication and collaboration between doctors and other modules.

PATIENT/APPOINTMENT MODULE:

The patient/appointment module focuses on providing convenient services to patients. It allows patients to register online, request appointments, and view available time slots. The module enables patients to access their medical records, receive appointment reminders, and communicate with doctors or receptionists. It provides a user-friendly interface for patients to manage their appointments and access relevant information.

TABLE DESCRIPTION

Table 1 Receptionist Schema

S.No	FIELD NAME	TYPE	CONSTRAINTS	DESCRIPTION
1.	rid	Integer	Primary Key, Unique Key	Employee Id of the receptionist
2.	rname	Varchar(40)	Not Null	Name of the receptionist
3.	rcont	Long	Not Null	Contact number of the receptionist
4.	rpan	Varchar(10)	Not Null	PAN card number of the receptionist

The Receptionist schema is shown in the above table 1, represents receptionists in an organization, storing their employee ID, name, contact number, and PAN card number.

Table 2 Doctor Schema

S.No	FIELD NAME	TYPE	CONSTRAINTS	DESCRIPTION
1.	did	Integer	Primary Key, Unique Key	Employee Id of the doctor
2.	dname	Varchar(40)	Not Null	Name of the doctor
3.	dcont	Long	Not Null	Contact number of the doctor
4.	dlic	Varchar(10)	Not Null	License number of the doctor

The Doctor schema is shown in the above table 2, represents for doctors, storing their employee id, name, contact number, and license number.

Table 3 Patient Schema

S.No	FIELD NAME	TYPE	CONSTRAINTS	DESCRIPTION
1.	aptno	Integer	Primary Key, Unique Key	Appointment number for the patient
2.	pname	Varchar(40)	Not Null	Name of the patient
3.	pcont	Long	Not Null	Contact number of the patient
4.	aptdate	Varchar(10)	Not Null	Appointment date for the patient
5.	reason	Varchar(50)	Not Null	Reason for the patient to consult doctor
6.	did	Integer	Foreign Key	Consulting doctor’s employee id
7.	rid	Integer	Foreign Key	Receptionist’s employee id who’s helping patient to consult doctor

The Patient schema is shown in the above table 3, a database schema for patient appointments, storing appointment numbers, patient details (name and contact number), appointment date, reason for consultation, and foreign keys referencing the consulting doctor's employee ID and the receptionist's employee ID. The appointment number serves as the primary key, ensuring uniqueness.

HTML CODE

Home Page

```
<!DOCTYPE html>
<html>
<head>
   <h2>
      <a href="#" style="text-decoration: none">AJP Hospital
      </a></h2>
      <title>AJP Hospital</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         background-image: url("image.jpg");
         background-repeat: no-repeat;
         background-size:cover ;
      }
      h2 {
   text-align: center;
   margin-bottom: 20px;
}
      button {
         display: inline-block;
```

```
            padding: 10px 20px;
            font-size: 16px;
            border: none;
            border-radius: 4px;
            background-color: #337ab7;
            color: #fff;
            cursor: pointer;
            margin-right: 10px;
        }
        button:hover {
            background-color: #23527c;
        }
    </style>
</head>
<body><center>
    <div>
    <button onclick= "window.location.href = 'Admin.html';">Admin</button>
    <button onclick= "window.location.href = 'Receptionist.html';">Appointment Booking</button>
    <button onclick= "window.location.href = 'Doctor.html';">Doctor</button>
    </div></center>
</body>
</html>
```

Admin Page

```
<!DOCTYPE html>
<html>
<head>
   <title>Admin Page</title>
   <link rel="stylesheet" href="Style.css">
</head>
<body>
   <div class="container">
      <div class="header">
         <h2>
            <a style="text-decoration: none" href="HomePage.html">AJP HOSPITAL</a>
         </h2>
      </div>
      <div>
         <h4>Receptionist</h4>
         <div class="menu">
            <button onclick="window.location.href = 'AddRecep.html';">Add</button>
            <button onclick="window.location.href = 'DelRecep.html';">Delete</button>
            <button onclick="window.location.href = 'UpdRecep.html';">Update</button>
            <div>
            <form action="ViewRecep" method="post"><input type="Submit"
value="View"></form>
         </div>
```

```
        </div>
    </div>
    <div>
        <h4>Doctor</h4>
        <div class="menu">
            <button onclick="window.location.href = 'AddDr.html';">Add</button>
            <button onclick="window.location.href = 'DelDr.html';">Delete</button>
            <button onclick="window.location.href = 'UpdDr.html';">Update</button>
            <div>
            <form action="ViewDr" method="post"><input type="Submit"
value="View"></form>
            </div>
        </div>
    </div>
    <div>
        <h4>Patient/Appointment</h4>
        <div class="menu">
            <button onclick="window.location.href = 'AddPt.html';">Add</button>
            <button onclick="window.location.href = 'DelPt.html';">Delete</button>
            <button onclick="window.location.href = 'UpdPt.html';">Update</button>
            <div>
            <form action="ViewPt" method="post"><input type="Submit"
value="View"></form>
            </div>
```

```
        </div>
      </div>
   </div>
</body>
</html>
```

Receptionist Page

```
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" href="Style.css">
   <title>Receptionist Page</title>
   <center><h2>
      <a style="text-decoration: none" href="HomePage.html">AJP HOSPITAL
      </a></h2></center>
</head>
<body>
  <div class="container">
   <h3>Patient/Appointment</h3>
      <div class="menu">
      <button id="add" onclick= "window.location.href = 'AddPt.html';">Add</button>
      <button id="update" onclick= "window.location.href =
'UpdPt.html';">Update</button>
      <button id="delete" onclick= "window.location.href = 'DelPt.html';">Delete</button>
```

```
        <div>
        <form action="ViewPt" method="post"><input type="Submit"
value="View"></form>
        </div>
        </div>
    </div>
</body>
</html>
```

CSS CODE

```
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-image: url("MainBG.jpg");
        background-repeat: no-repeat;
        background-size:cover ;
        }
        .container {
            max-width: 960px;
            margin: 0 auto;
            padding: 20px;
        }
        .menu button {
```

```
    display: block;
    padding: 10px 20px;
    border: none;
    background-color: #00A8FF;
    color: #fff;
    cursor: pointer;     }
input[type="text"] {
    width: 100%;
    padding: 5px;
    font-size: 16px;
}
input:hover[type="submit"]
            {
                    background: aquamarine;
            }
*,
*:before,
*:after{
    box-sizing: border-box;
}
input[type="date"]{
    background-color: #00A8FF;
    padding: 3px;
    font-family: "Roboto Mono",monospace;
```

```
    color: black;
    border-radius: 5px;
}
::-webkit-calendar-picker-indicator{
    background-color: #ffffff;
    padding: 5px;
    cursor: pointer;
    border-radius: 3px;
}
```

JAVA CODE

Add Receptionist

```
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.*;
@WebServlet("/AddRecep")
public class AddRecep extends HttpServlet{
    private static final long serialVersionUID = 1L;
    public AddRecep() {
        super();
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
```

```
		PrintWriter pw = response.getWriter();
		int id = Integer.parseInt(request.getParameter("id"));
		String name = request.getParameter("name");
		long cont = Long.parseLong(request.getParameter("cont"));
		String pan = request.getParameter("pan");
		try {
			Class.forName("com.mysql.jdbc.Driver");
		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3307/ajphospital",
"root","subash#faf");
			System.out.println("Connected "+con);
			String sql = "INSERT INTO receptionist VALUES(?,?,?,?)";
			PreparedStatement ps = con.prepareStatement(sql);
			ps.setInt(1, id);
			ps.setString(2, name);
			ps.setLong(3, cont);
			ps.setString(4, pan);
			ps.executeUpdate();
			con.close();
			pw.println("Receptionist "+name+"'s details added Successfully!");
		}
		catch(ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		}
```

```
	}
	protected void doPost(HttpServletRequest req, HttpServletResponse response)
throws ServletException, IOException {
		doGet(req, response);
	}
}
```

Add Doctor

```
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.*;
@WebServlet("/AddDr")
public class AddDr extends HttpServlet{
	private static final long serialVersionUID = 1L;
	public AddDr() {
		super();
	}
	public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
		PrintWriter pw = response.getWriter();
		int id = Integer.parseInt(request.getParameter("id"));
		String name = request.getParameter("name");
```

```
		long cont = Long.parseLong(request.getParameter("cont"));
		String lic = request.getParameter("lic");
		try {
			Class.forName("com.mysql.jdbc.Driver");
			Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:33070/ajphospital",
"root","subash#faf");
			System.out.println("Connected "+con);
			String sql = "INSERT INTO doctor VALUES(?,?,?,?)";
			PreparedStatement ps = con.prepareStatement(sql);
			ps.setInt(1, id);
			ps.setString(2, name);
			ps.setLong(3, cont);
			ps.setString(4, lic);
			ps.executeUpdate();
			con.close();
			pw.println("Dr."+name+"'s details updated Successfully");
		}
		catch(ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		}
	}
	protected void doPost(HttpServletRequest req, HttpServletResponse response)
throws ServletException, IOException {
```

```
		doGet(req, response);
	}}
```

Delete Doctor

```
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.*;
@WebServlet("/DelDr")
public class DelDr extends HttpServlet{
	private static final long serialVersionUID = 1L;
	public DelDr() {
		super();}
	public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
		PrintWriter pw = response.getWriter();
		int id = Integer.parseInt(request.getParameter("id"));
		String name = request.getParameter("name");
		try {
			Class.forName("com.mysql.jdbc.Driver");
			Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:33070/ajphospital",
"root","subash#faf");
```

```
                System.out.println("Connected "+con);
                String sql = "DELETE FROM doctor where did=?";
                PreparedStatement ps = con.prepareStatement(sql);
                ps.setInt(1, id);
                ps.executeUpdate();
                con.close();
                pw.println("Dr."+name+"'s details removed Successfully");
            }
            catch(ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }
        protected void doPost(HttpServletRequest req, HttpServletResponse response)
throws ServletException, IOException {
            doGet(req, response);
        }
}
```

Add Patient/Appointment

```
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.*;
```

```
@WebServlet("/AddPt")
public class AddPt extends HttpServlet{
	private static final long serialVersionUID = 1L;
	public AddPt() {
		super();
	}
	public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
		PrintWriter pw = response.getWriter();
		int aptno = Integer.parseInt(request.getParameter("aptno"));
		String name = request.getParameter("name");
		long cont = Long.parseLong(request.getParameter("cont"));
		String date = request.getParameter("date");
		String reason = request.getParameter("reason");
		int did = Integer.parseInt(request.getParameter("did"));
		String dname = request.getParameter("dname");
		int rid = Integer.parseInt(request.getParameter("rid"));
		String rname = request.getParameter("rname");
		try {
			Class.forName("com.mysql.jdbc.Driver");
			Connection con = DriverManager.getConnection("jdbc:mysql://localhost:33070/ajphospital", "root","subash#faf");
			System.out.println("Connected "+con);
```

```
                    String sql = "INSERT INTO appointment VALUES(?,?,?,?,?,?,?)";
                    PreparedStatement ps = con.prepareStatement(sql);
                    ps.setInt(1, aptno);
                    ps.setString(2, name);
                    ps.setLong(3, cont);
                    ps.setString(4, date);
                    ps.setString(5, reason);
                    ps.setInt(6, did);
                    ps.setInt(7, rid);
                    ps.executeUpdate();
                    con.close();
                    pw.println("Appointment created for patient Mr/Mrs/Ms."+name+"
on "+date);
                    pw.println("Appointment Number: "+aptno);
                    pw.println("Consulting Doctor: "+dname);
                }
        }
        protected void doPost(HttpServletRequest req, HttpServletResponse response)
throws ServletException, IOException {
                doGet(req, response);
                }
        }
```

CONCLUSION

The project aims to develop a hospital management system that streamlines various operations within a hospital. By utilizing HTML and Java programming languages, the system provides a user-friendly interface for different users such as admin, receptionists, doctors, and patients.

The admin module enables efficient management of user accounts, system configurations, and monitoring of system activities. It empowers the admin to effectively control the system and ensure its smooth functioning.

The receptionist module simplifies front desk operations by offering features like patient registration, appointment scheduling, and handling patient inquiries. This module enhances the efficiency of the receptionists and improves patient satisfaction.

The doctor module facilitates easy access to patient information, medical records, and appointment management for doctors. It enables doctors to update diagnoses, prescribe medications, and request medical tests efficiently.

The patient/appointment module provides convenient services to patients, allowing them to register online, request appointments, and access medical records. Patients can receive appointment reminders, communicate with doctors or receptionists, and manage their appointments effectively.

Overall, the project enhances the management of hospital operations, improves communication between staff and patients, and provides a user-friendly experience for all users. It contributes to the efficient functioning of the hospital, leading to improved patient care and satisfaction.

SCREENSHOTS

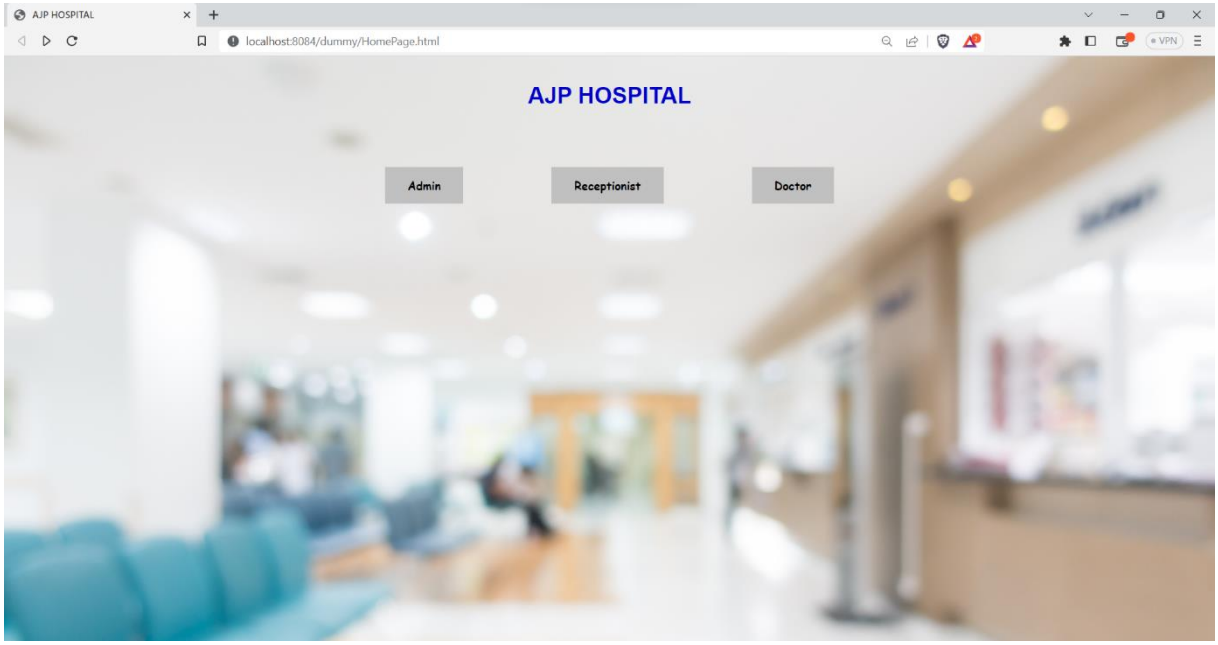

Figure 1 Home Page

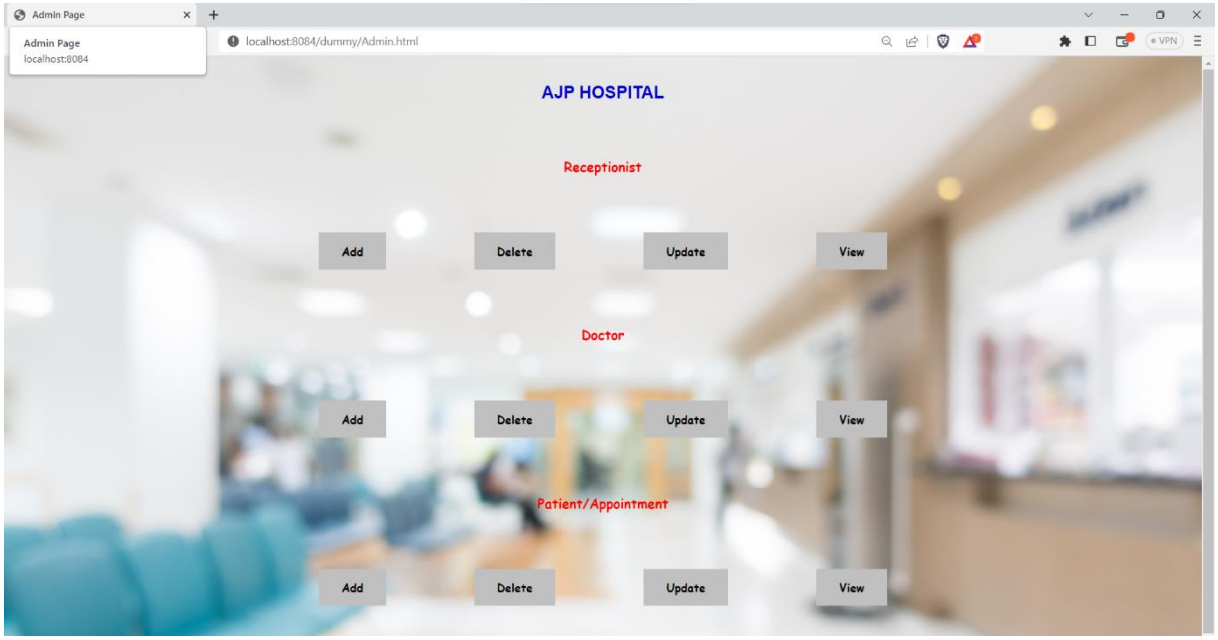

Figure 2 Admin Page

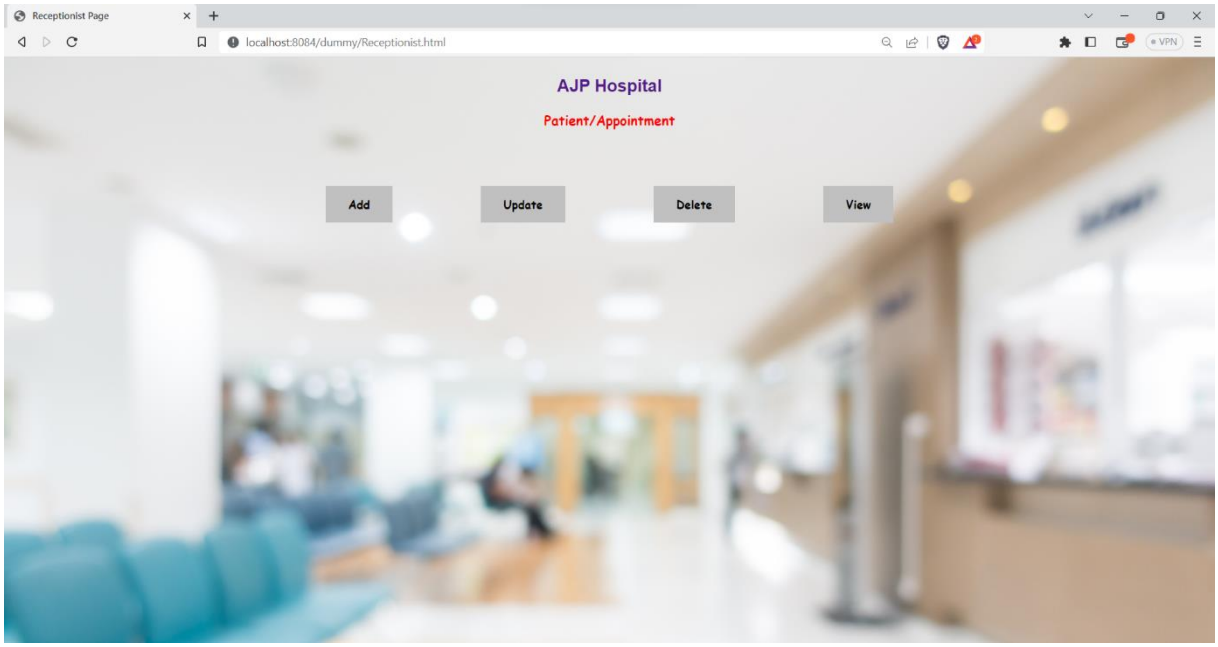

Figure 3 Receptionist Page

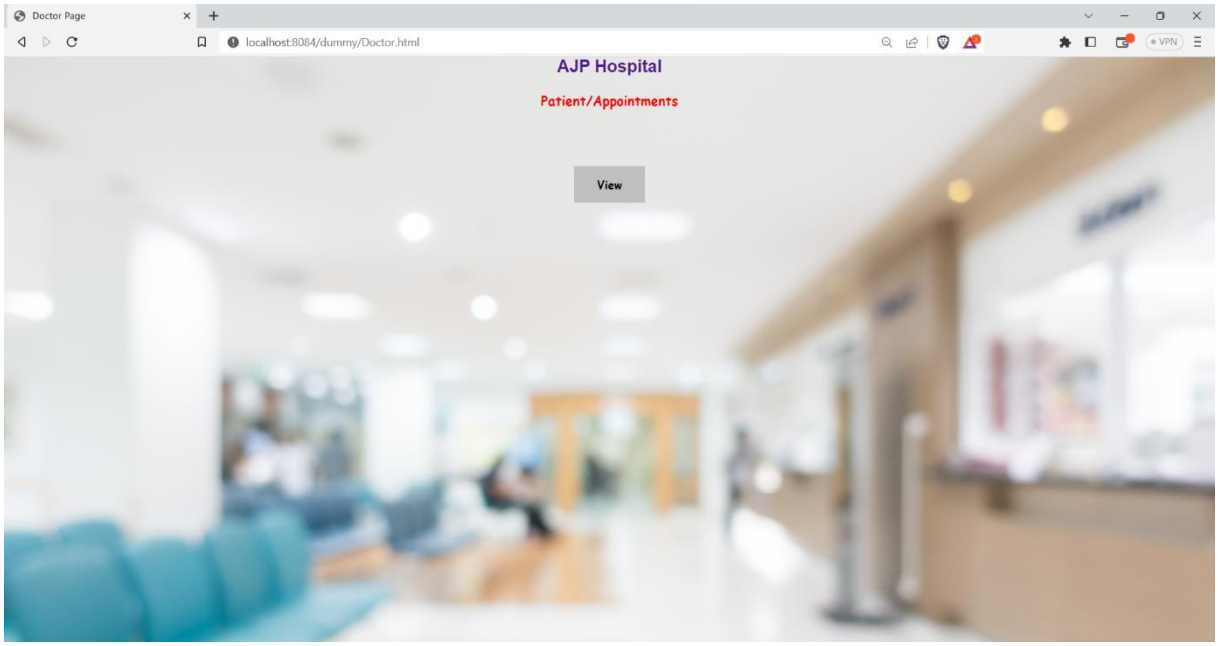

Figure 4 Doctor Page

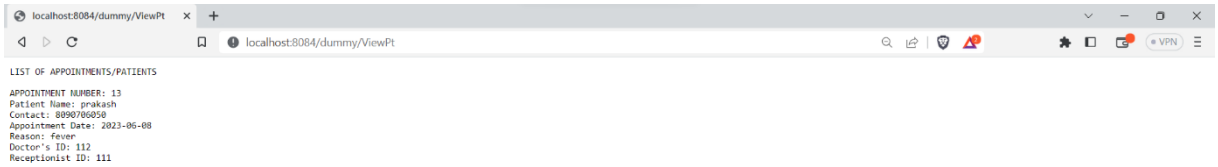

Figure 5 Patients/Appointments View Page

Figure 6 Receptionist View Page

Figure 7 Doctor View Page

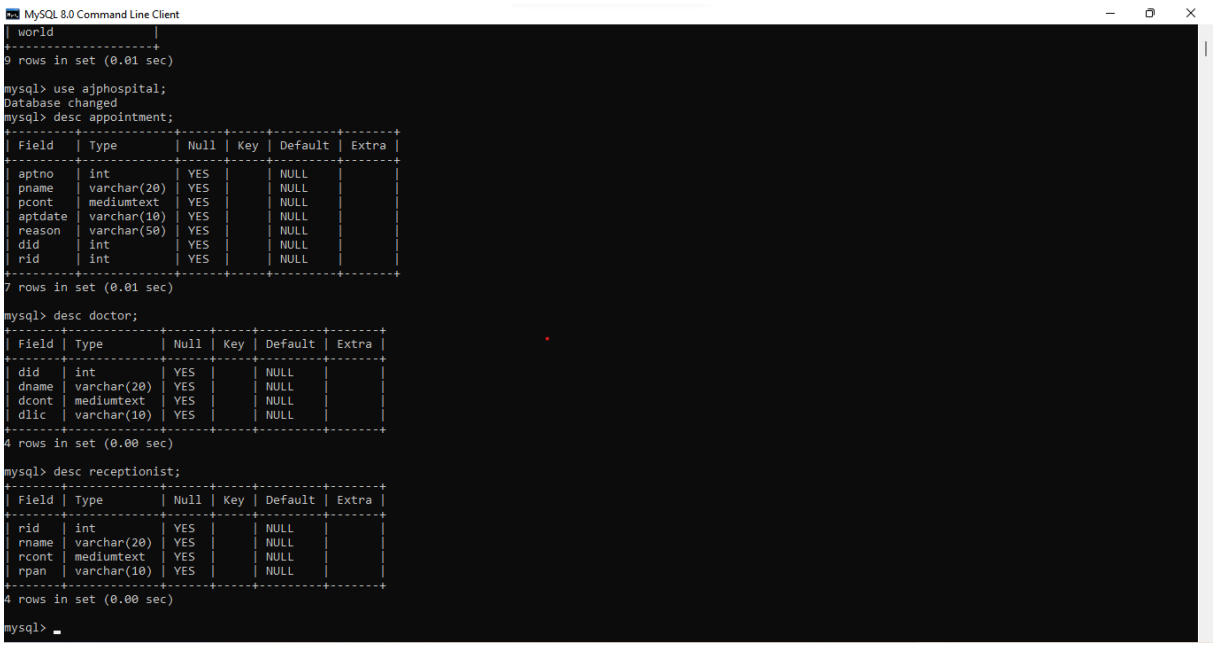

Figure 8 Database Tables Description