PROJECT REPORT OF INDUSTRY ORIENTED HANDS-ON EXPERIENCE (IOHE)

ON

MediSphereX

submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By:
Avinash Mehta(2010990142)
Harshdeep Singh(2010990280)
Lakshay Garg(2010991606)

Supervised By:
Mr.Abhishek Bhardwaj

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHITKARA UNIVERSITY

CHANDIGARH-PATIALA NATIONAL HIGHWAY, RAJPURA, PUNJAB, INDIA

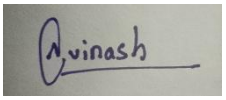
CONTENTS

DECLARATION

We hereby declare that the project work titled, MediSphereX submitted as part of Bachelor's degree in CSE, at Chitkara University, Punjab, is an authentic record of our own work carried out under the supervision of Mr. Abhishek Bhardwaj .

Signature(s):

Avinash Mehta



ACKNOWLEDGEMENT

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

We express our sincere gratitude to all for providing me an opportunity to undergo this Project as the part of the curriculum.

We are thankful to Mr. Abhishek Bhardwaj for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.He provided his valuable suggestions and precious time in accomplishing our Integrated project report.

Lastly, We would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to day experience and received lots of suggestions that improve our quality of work.

Harshdeep Singh	**Avinash Mehta**	**Lakshay Garg**
ID No 2010990280	**ID No 2010990142**	**ID No 2010991606**

LIST OF FIGURES AND TABLES

ABSTRACT

The purpose of the project entitled "MediSphereX" is to computerize the Front Office Management of a hospital by developing user-friendly, simple, fast, and cost-effective software. It deals with the collection of patient information, diagnosis details, and other relevant data, which was traditionally done manually.

The main functions of the system are to register and store patient details, doctor details, and retrieve this information as and when required. It also allows for the meaningful manipulation of these details.

The system input includes patient details and diagnosis information, while the system output displays these details on the screen. The Hospital Management System can be accessed using a username and password, either by an administrator or receptionist. Only they can add data into the database, which can be easily retrieved. The data is well-protected for personal use, making the data processing very fast.

INTRODUCTION

The **MediSphereX** project is a hospital management system that includes patient registration, storage of patient details, and computerized appointment booking. The software assigns a unique ID to every patient and automatically stores the details of patients and staff. It includes a search feature to check the current status of each patient appointment. Users can search patient details using the unique patient ID.

The Hospital Management System can be accessed using a username and password, either by an administrator or receptionist. Only they can add data into the database, which can be easily retrieved. The interface is user-friendly, and the data is well-protected for personal use, making the data processing very fast.

The Hospital Management System is a powerful, flexible, and user-friendly solution designed to provide tangible benefits to hospitals. It is an integrated end-to-end system that covers a wide range of hospital administration and management processes, providing relevant information across the hospital to support effective decision-making for patient care, hospital administration, and critical financial accounting.

The Hospital Management System is designed for multi-specialty hospitals to improve the quality and management of hospital operations, including clinical process analysis and activity-based costing. It enables organizations to develop and improve their effectiveness and efficiency.

1.1 Problem Introduction:

Lack of immediate retrievals: - The information is very difficult to retrieve and to find particular information like- E.g. - To find out about the patient's history, the user has to go through various registers. This results in in convenience and wastage of time.

Lack of immediate information storage: - The information generated by various transactions takes time and efforts to be stored at right place.

Lack of prompt updating: - Various changes to information like patient details or immunization details of child are difficult to make as paper work is involved.

Error prone manual calculation: - Manual calculations are error prone and take a lot of time this may result in incorrect information. For example calculation of patient's bill based on various treatments.

Preparation of accurate and prompt reports: - This becomes a difficult task as information is difficult to collect from various register.

CHITKARA UNIVERSITY

1.2 Goals:

- User friendly
- Simple fast
- Low cost and effective
- It deals with the collection of patient's information
- Diagnosis

1.3 Objectives

The primary objectives of the MediSphereX hospital management system are:

- To provide a comprehensive and integrated platform for managing various aspects of hospital operations, including patient records, appointment scheduling, inventory control, and billing.
- To enhance the overall patient experience by enabling efficient communication, reducing wait times, and improving the delivery of care.
- To empower healthcare professionals with intuitive and user-friendly tools that optimize operational efficiency and facilitate the provision of high-quality care.
- To leverage modern web technologies, database management, and the Spring MVC framework to create a scalable and adaptable solution that meets the evolving needs of healthcare institutions.
- To facilitate seamless integration with existing hospital information systems and enable efficient data management through the utilization of Hibernate.
- Ensure compatibility with mobile devices to allow healthcare providers and patients to access the system on-the-go, enhancing convenience and accessibility.
- Implement robust security measures to protect patient data against unauthorized access or breaches.

METHODOLOGY

Requirement Analysis: The first phase involves a comprehensive analysis of the requirements of healthcare facilities, including hospitals, clinics, and medical centers. This involves gathering input from healthcare professionals, administrators, and IT staff to understand the specific needs and challenges faced in managing hospital operations.

System Design: Based on the gathered requirements, the system design phase focuses on creating a detailed blueprint for the MediSphereX hospital management system. This includes designing the

database schema using MySQL, creating wire frames for the user interface, and planning the architecture of the system using Java, Spring MVC, and Hibernate.

Frontend Development: The frontend development phase involves the implementation of the user interface using HTML, CSS, JavaScript, and Bootstrap. This includes creating intuitive and responsive interfaces for patient registration, appointment scheduling, billing, and other user interactions.

Backend Development: Simultaneously, the backend development phase focuses on implementing the server-side logic using Java and the Spring MVC framework. This includes developing the business logic, data processing, and integration with the MySQL database using Hibernate for efficient data management.

Integration and Testing: Once the frontend and backend components are developed, they are integrated to form a cohesive system. Rigorous testing is then conducted to ensure the functionality, performance, and security of the MediSphereX system. This includes unit testing, integration testing, and user acceptance testing to validate the system against the initial requirements.

Deployment and Training: After successful testing, the MediSphereX hospital management system will be deployed in the healthcare facilities. Training sessions will be conducted for the staff to familiarize them with the system's features and functionalities, ensuring a smooth transition to the new system.

Maintenance and Support: Post-deployment, ongoing maintenance and support are provided to address any issues, implement updates, and incorporate feedback from users. This ensures the continued reliability and effectiveness of the MediSphereX system in meeting the evolving needs of healthcare institutions.

The entire project mainly consists of:

1. **User Module(Patient)**
2. **Admin Module**
3. **Doctor Module(Staff)**

User Module:

The user module consists of various functionalities for the user (patient) such as navigating throughout website of the hospital where the user can find all the details related to the hospital.The user module consists of:

1. About section
2. Doctor’s Info section
3. Departments section
4. Pricing section

5. Blog section
6. User Registration/Login section
7. Book Appointment section

Admin Module:

The admin module allows the hospital staff to manage the information of patients,doctors,appointments,etc.It can be accessed using unique login id and password.

It consists of:

1. ADD-DOCTOR
2. VIEW-DOCTOR
3. VIEW PATIENT
4. UPDATE DOCTOR
5. DELETE DOCTOR
6. TOTAL NO. OF USERS
7. TOTAL APPOINTMENTS BOOKED
8. ADD SPECIALIST DETAILS

Doctor Module:

The doctor module consists of doctor login page where the doctor will be assigned a unique id and password by the admin.The doctor can then login using the id, password and view the doctor's dashboard which consists of the appointments booked to that particular doctor and can comment on the patient details and treatment given.

Database Analyzing, design and implementation

Sr No.	Column Name	Data Type	Description
1.	Id	int	Contains user id
2	full_name	Varchar(45)	Contains full name of the user

3.	email	Varchar(45)	Contains email id of the user
4.	password	Varchar(45)	Contains password of the user

Table 1.1 User Details

Sr No.	Column Name	Data Type	Description
1.	Id	int	Contains serial id
2	user_id	int	Contains user id of the user
3.	fullname	Varchar(45)	Contains name of the patient
5.	gender	Varchar(45)	Contains gender of the patient
6.	age	Varchar(45)	Contains age of the patient
7.	appointDate	Varchar(45)	Contains the appointment date
8.	email	Varchar(45)	Contains the email of the patient
9.	phno	int	Contains the contact of the patient
10.	diseases	Varchar(45)	Contains the disease acquired by the

			patient
11.	doctor_id	int	Contains the doctor id
12.	status	Varchar(45)	Contains the status of the patient's appointment

Table 1.2 Appointment

Sr No.	Column Name	Data Type	Description
1.	Id	int	Contains id
2	full_name	Varchar(45)	Contains full name of the doctor
3.	dob	Varchar(45)	Contains dob of the doctor.
4.	qualification	Varchar(45)	Contains qualification of the doctor
5	specialist	Varchar(45)	contains speciality of the doctor
6	email	Varchar(45)	contains email of the doctor
7	mobno	varchar(10)	contains mobile no. of the doctor

8	password	Varchar(45)	contains password of thr doctor login id

Table 1.3 Doctor

Sr No.	Column Name	Data Type	Description
1.	Id	int	Contains id
2	spec_name	Varchar(45)	Contains speciality of the doctor.

Table 1.4 Speciality

TOOLS AND TECHNOLOGIES

HTML: HTML is used to create the structure and content of the web pages within the MediSphereX system. It is responsible for defining the various elements of the user interface, such as forms for patient registration, input fields for medical records, and layout structures for displaying information.

CSS: CSS is utilized to style and format the HTML elements, ensuring a consistent and visually appealing presentation of the MediSphereX interface. It is used to define the colors, fonts, spacing, and overall layout of the web pages, enhancing the user experience.

JavaScript: JavaScript is employed to add interactivity and dynamic functionality to the MediSphereX system. It facilitates tasks such as form validation, real-time data updates, and interactive user interfaces, enhancing the responsiveness and usability of the application.

Bootstrap: Bootstrap is used to leverage its pre-built design templates and components to create a responsive and mobile-friendly user interface for MediSphereX. It provides a framework for building consistent and visually appealing web pages across different devices and screen sizes.

MySQL: MySQL serves as the database management system for the MediSphereX system, storing and organizing the structured data related to patient records, appointments, medical history, and other essential information. It enables efficient data storage, retrieval, and management for the hospital management system.

Java Server Pages (JSP): JSP is utilized to create dynamic web pages within the MediSphereX system, allowing for the integration of Java code with HTML to generate dynamic content. It enables the presentation of data-driven information and the execution of server-side logic within the web application.

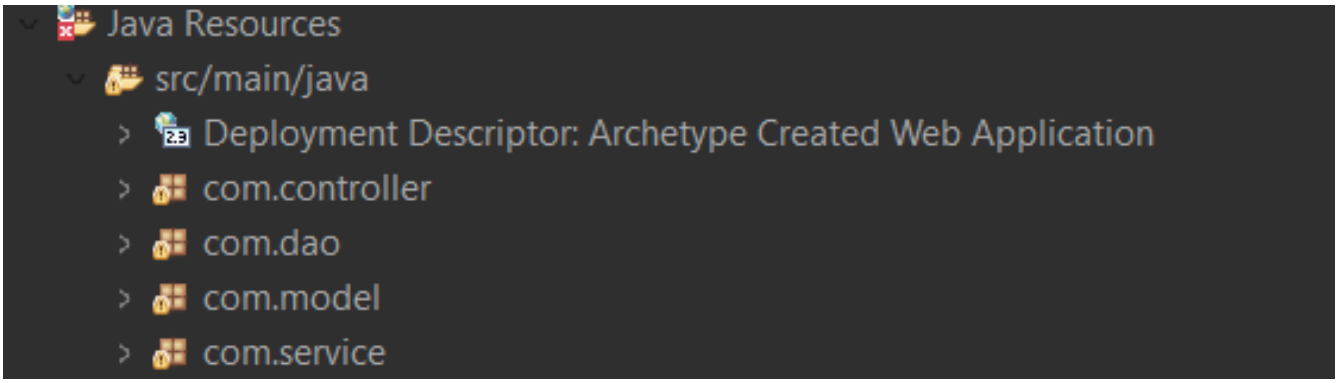
Java: Java is used for the server-side development of the MediSphereX system, providing the foundation for implementing business logic, data processing, and system integration. It enables the creation of robust and scalable server-side components for the hospital management system.

Spring MVC: The Spring MVC framework is employed to simplify the development of the server-side components of the MediSphereX system. It provides a robust and scalable architecture for handling web requests, managing the application's business logic, and facilitating the integration with the MySQL database using Hibernate.

Hibernate: Hibernate is used as the Object-Relational Mapping (ORM) framework to simplify the interaction between the Java objects and the MySQL database. It enables efficient data management, mapping database tables to Java classes, and providing a seamless way to store, retrieve, and update data.

IMPLEMENTATION

Java Resources implemented in the project:



1. **com.Controller**

```java
package com.controller;

import com.model.User;

@Controller
public class AdminController {

    @PostMapping("/adminLogin")
    public String adminLogin(@RequestParam("email") String email,
                             @RequestParam("password") String password,
                             HttpSession session) {

        try {
            if ("admin@gmail.com".equals(email) && "admin".equals(password)) {
                session.setAttribute("adminObj", new User());
                return "redirect:/admin/index.jsp";
            } else {
                session.setAttribute("errorMsg1", "Invalid email or password");
                return "redirect:/admin_login.jsp";
            }
        } catch (Exception e) {
            e.printStackTrace();
            return "redirect:/admin_login.jsp";
        }
    }
    @GetMapping("/adminLogout")
    public String adminLogout(HttpSession session) {
        // Invalidating session
        session.invalidate();

        // Redirect to login page
        return "redirect:/admin_login.jsp";
    }
```

AdminController.java

```java
package com.controller;

import java.util.List;

@Controller
public class AppointmentController {

    @Autowired
    private AppointmentService appointmentService;



    @PostMapping("/saveAppointment")
    public String saveAppointment(@ModelAttribute("appointment") Appointment appointment) {
        boolean saved = appointmentService.saveAppointment(appointment);
        if (saved) {

            return "redirect:/appointments";
        } else {

            return "redirect:/appointmentForm";
        }
    }


    @GetMapping("/appointments")
    public String getAllAppointments(Model model) {
        List<Appointment> appointments = appointmentService.getAllAppointment();
        model.addAttribute("appointments", appointments);
        return "appointment_list";
    }

    @GetMapping("/appointments/{id}")
    public String getAppointmentById(@PathVariable("id") int id, Model model) {
        Appointment appointment = appointmentService.getAppointmentById(id);
        model.addAttribute("appointment", appointment);
        return "appointment_details";
}
}
```

AppointmentController.java

```java
package com.controller;

import java.util.List;

@Controller
@RequestMapping("/doctor")
public class DoctorController {

    @Autowired
    private DoctorService doctorService;

    @GetMapping("/list")
    public String getAllDoctors(Model model) {
        List<Doctor> doctors = doctorService.getAllDoctors();
        model.addAttribute("doctors", doctors);
        return "admin/viewdoctor";     }

    @GetMapping("/{id}")
    public String getDoctorById(@PathVariable("id") int id, Model model) {
        Doctor doctor = doctorService.getDoctorById(id);
        model.addAttribute("doctor", doctor);
        return "doctor_details";
    }

    @GetMapping("/add")
    public String showAddDoctorForm() {
        return "add_doctor";
    }

    @PostMapping("/add")
    public String addDoctor(@RequestParam("fullName") String fullName,
                            @RequestParam("dob") String dob,
                            @RequestParam("qualification") String qualification,
                            @RequestParam("specialist") String specialist,
                            @RequestParam("email") String email,
                            @RequestParam("password") String password,
                            @RequestParam("mobno") String mobno) {
        Doctor doctor = new Doctor(fullName, dob, qualification, specialist, email, mobno, password);
        doctorService.registerDoctor(doctor);
        return "redirect:/doctor/list";
    }

    @GetMapping("/edit/{id}")
```

DoctorController.java

```java
@GetMapping("/edit/{id}")
public String showEditDoctorForm(@PathVariable("id") int id, Model model) {
    Doctor doctor = doctorService.getDoctorById(id);
    model.addAttribute("doctor", doctor);
    public String showEditDoctorForm(@PathVariable("id") int id, Model model
}                                                                Press 'F2' for focus

@PostMapping("/edit")
public String editDoctor(Doctor doctor) {
    doctorService.updateDoctor(doctor);
    return "redirect:/doctor/list";
}

@GetMapping("/delete/{id}")
public String deleteDoctor(@PathVariable("id") int id) {
    doctorService.deleteDoctor(id);
    return "redirect:/doctor/list";
}

@GetMapping("/login")
public String showLoginForm() {
    return "doctor_login";
}

@PostMapping("/login")
public String loginDoctor(@RequestParam("email") String email,
                          @RequestParam("password") String password,
                          Model model) {
    Doctor doctor = doctorService.login(email, password);
    if (doctor != null) {

        return "redirect:/doctor/dashboard";
    } else {
        model.addAttribute("errorMsg", "Invalid email or password");
        return "doctor_login";
    }
}
```

DoctorController.java

```java
package com.controller;

import com.model.Specialist;

@Controller
public class SpecialistController {

    @Autowired
    private SpecialistService specialistService;



    @GetMapping("/addspecialist")
    public String showAddSpecialistForm(Model model) {
        model.addAttribute("specialist", new Specialist());
        return "add_specialist";
    }

    // POST mapping to handle form submission for adding a specialist
    @PostMapping("/addspecialist")
    public String addSpecialist(@ModelAttribute("specialist") Specialist specialist, Model model) {
        boolean isSuccess = specialistService.addSpecialist(specialist.getSpecialistName());
        if (isSuccess) {
            model.addAttribute("succMsg", "Specialist added successfully");
        } else {
            model.addAttribute("errorMsg", "Failed to add specialist");
        }
        return "redirect:/specialists";
    }


    @GetMapping("/specialists")
    public String getAllSpecialists(Model model) {
        List<Specialist> specialists = specialistService.getAllSpecialists();
        model.addAttribute("specialists", specialists);
        return "specialist_list";
    }

}
```

SpecialistController.java

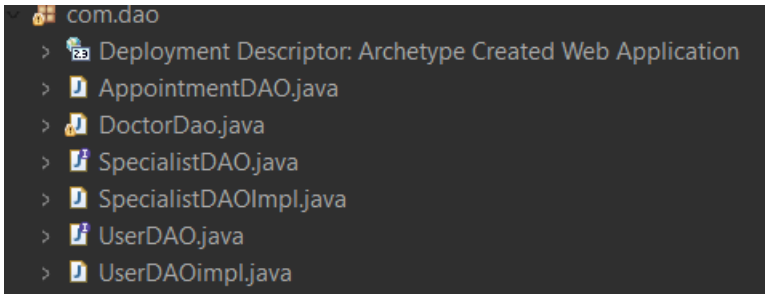
```java
package com.controller;

import com.dao.AppointmentDAO;

@Controller
public class UpdateStatusController {

    @Autowired
    private AppointmentDAO appointmentDAO;

    @PostMapping("/updateStatus")
    public String updateStatus(@RequestParam("id") int id,
                               @RequestParam("did") int doctorId,
                               @RequestParam("comm") String comment,
                               HttpSession session) {

        try {
            if (appointmentDAO.updateCommentStatus(id, doctorId, comment)) {
                session.setAttribute("succMsg", "Comment Updated");
            } else {
                session.setAttribute("errorMsg", "Failed to update comment");
            }
        } catch (Exception e) {
            e.printStackTrace();
            session.setAttribute("errorMsg", "Exception occurred: " + e.getMessage());
        }

        return "redirect:/doctor/patient.jsp";
    }
}
```

UpdateStatusController.java

```java
import com.model.User;

@Controller
public class UserController {

    @Autowired
    private UserService userService;



    @GetMapping("/register")
    public String registerUser(@ModelAttribute("user") User user, Model model) {
        boolean isSuccess = userService.saveUser(user);
        if (isSuccess) {
            model.addAttribute("succMsg", "User registered successfully");
        } else {
            model.addAttribute("errorMsg", "Failed to register user");
        }
        return "redirect:/login";
    }

    @GetMapping("/login")
    public String showLoginForm(Model model) {
        model.addAttribute("user", new User());
        return "login";
    }

    @PostMapping("/login")
    public String loginUser(@ModelAttribute("user") User user, Model model) {
        User loggedInUser = userService.login(user.getEmail(), user.getPassword());
        if (loggedInUser != null) {
            model.addAttribute("succMsg", "Login successful");

            return "redirect:/index";
        } else {
            model.addAttribute("errorMsg", "Invalid email or password");
            return "login";
        }
    }
    @GetMapping("/logout")
    public String logout(HttpServletRequest request) {
        HttpSession session = request.getSession(false);
```

UserController.java

2. com.dao



```java
package com.dao;

import java.util.ArrayList;

@Repository
public class AppointmentDAO {

    @Autowired
    private SessionFactory sessionFactory;

    public boolean saveAppointment(Appointment appointment) {
        Transaction transaction = null;
        try (Session session = sessionFactory.getCurrentSession()) {
            transaction = session.beginTransaction();
            session.save(appointment);
            transaction.commit();
            return true;
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
            return false;
        }
    }

    public ArrayList<Appointment> getAllAppointmentByLoginUser(int userId) {
        Transaction transaction = null;
        ArrayList<Appointment> list = null;
        try (Session session = sessionFactory.getCurrentSession()) {
            transaction = session.beginTransaction();
            Query<Appointment> query = session.createQuery("from Appointment where userId = :userId", Appointment.c
            query.setParameter("userId", userId);
            list = (ArrayList<Appointment>) query.getResultList();
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return list;
    }
```

AppointmentDAO.java

```java
package com.dao;

import java.util.List;

@Repository
public class DoctorDao {

    @Autowired
    private SessionFactory sessionFactory;

    @Transactional
    public boolean registerDoctor(Doctor doctor) {
        try {
            sessionFactory.getCurrentSession().save(doctor);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    @Transactional(readOnly = true)
    public List<Doctor> getAllDoctor() {
        try {
            Session session = sessionFactory.getCurrentSession();
            Query<Doctor> query = session.createQuery("from Doctor order by id desc", Doctor.class);
            return query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @Transactional(readOnly = true)
    public Doctor getDoctorById(int id) {
        try {
            Session session = sessionFactory.getCurrentSession();
            return session.get(Doctor.class, id);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
```

DoctorDAO.java

```java
package com.dao;
import java.util.List;

    public interface SpecialistDAO {
        boolean addSpecialist(String spec);
        List<Specialist> getAllSpecialists();
    }
```

SpecialistDAO.java

```java
package com.dao;

import org.hibernate.Session;

@Repository
public class SpecialistDAOImpl implements SpecialistDAO {

    @Autowired
    private SessionFactory sessionFactory;

    @Override
    @Transactional
    public boolean addSpecialist(String spec) {
        try {
            Session session = sessionFactory.getCurrentSession();
            Specialist specialist = new Specialist(spec);
            session.save(specialist);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    @Override
    @Transactional
    public List<Specialist> getAllSpecialists() {
        try {
            Session session = sessionFactory.getCurrentSession();
            Query<Specialist> query = session.createQuery("from Specialist", Specialist.class);
            return query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

SpecialistDAOImpl.java

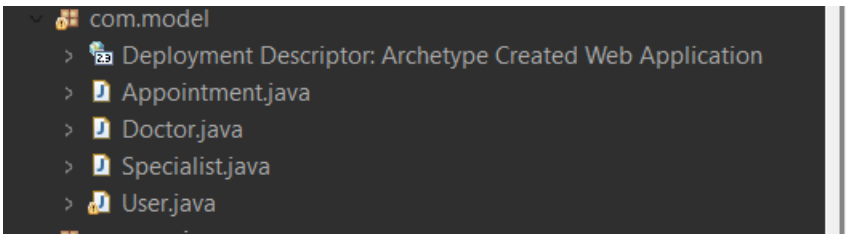
```java
package com.dao;

import com.model.User;

public interface UserDAO {
    boolean saveUser(User user);
    User login(String email, String password);
}
```

UserDAO.java

```java
package com.dao;
import org.hibernate.Session;

public class UserDAOimpl {


    @Repository
    public class UserDAOImpl implements UserDAO {

        @Autowired
        private SessionFactory sessionFactory;

        @Override
        @Transactional
        public boolean saveUser(User user) {
            try {
                Session session = sessionFactory.getCurrentSession();
                session.save(user);
                return true;
            } catch (Exception e) {
                e.printStackTrace();
                return false;
            }
        }

        @Override
        @Transactional(readOnly = true)
        public User login(String email, String password) {
            try {
                Session session = sessionFactory.getCurrentSession();
                String hql = "FROM User WHERE email = :email AND password = :password";
                Query<User> query = session.createQuery(hql, User.class);
                query.setParameter("email", email);
                query.setParameter("password", password);
                return query.uniqueResult();
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
        }
    }

}
```

UserDAOImpl.java

3. com.model



```java
package com.model;
import javax.persistence.*;

@Entity
public class Appointment {
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getUserId() {
        return userId;
    }
    public void setUserId(int userId) {
        this.userId = userId;
    }
    public String getFullName() {
        return fullName;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public String getAge() {
        return age;
    }
    public void setAge(String age) {
        this.age = age;
    }
    public String getAppoinDate() {
        return appoinDate;
    }
    public void setAppoinDate(String appoinDate) {
        this.appoinDate = appoinDate;
    }
    public String getEmail() {
        return email;
```

Appointment.java

```java
package com.model;

import javax.persistence.*;

@Entity
public class Doctor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;
    private String fullName;
    private String dob;
    private String qualification;
    private String specialist;
    private String email;
    private String mobNo;
    private String password;





    public Doctor() {
        super();
        // TODO Auto-generated constructor stub
    }





    public Doctor( String fullName, String dob, String qualification, String specialist, String
            String mobNo, String password) {
        super();

        this.fullName = fullName;
        this.dob = dob;
        this.qualification = qualification;
        this.specialist = specialist;
        this.email = email;
        this.mobNo = mobNo;
```

Doctor.java

```java
package com.model;

import javax.persistence.Entity;

@Entity
public class Specialist {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;
    private String specialistName;

public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getSpecialistName() {
        return specialistName;
    }
    public void setSpecialistName(String specialistName) {
        this.specialistName = specialistName;
    }
    public Sp                                    istName) {
        super();
        this.id = id;
        this.specialistName = specialistName;
    }
    public Specialist() {
        super();
        // TODO Auto-generated constructor stub
    }
    public Specialist(String spec) {
        this.specialistName = spec;

        // TODO Auto-generated constructor stub
    }



}
```

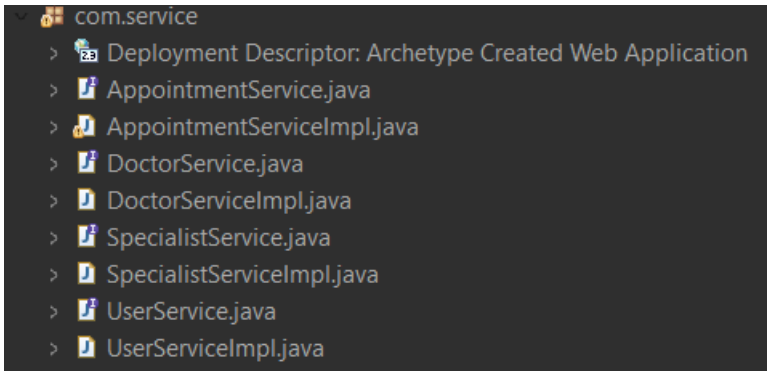
Specialist.java

```java
package com.model;

import javax.persistence.*;

@SuppressWarnings("restriction")
@Entity
@Table(name = "user_dtls")

public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "id")
    private int id;
    @Column(name = "full_name")
    private String fullname;
    @Column(name = "email")
    private String email;

    @Column(name = "password")
    private String password;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFullname() {
        return fullname;
    }

    public void setFullname(String fullname) {
        this.fullname = fullname;
    }

    public String getEmail() {
        return email;
    }
```

User.java

CHITKARA UNIVERSITY

4. com.service




```java
package com.service;

import java.util.List;

public interface AppointmentService {

    boolean saveAppointment(Appointment appointment);

    List<Appointment> getAllAppointmentByLoginUser(int userId);

    List<Appointment> getAllAppointmentByDoctorLogin(int doctorId);

    Appointment getAppointmentById(int id);

    boolean updateCommentStatus(int id, int doctorId, String status);

    List<Appointment> getAllAppointment();
}
```

AppointmentService.java

```java
package com.service;

import java.util.ArrayList;

@Service
public class AppointmentServiceImpl implements AppointmentService {

    @Autowired
    private AppointmentDAO appointmentDAO;

    @Override
    @Transactional
    public boolean saveAppointment(Appointment appointment) {
        return appointmentDAO.saveAppointment(appointment);
    }

    @Override
    @Transactional(readOnly = true)
    public List<Appointment> getAllAppointmentByLoginUser(int userId) {
        return appointmentDAO.getAllAppointmentByLoginUser(userId);
    }

    @Override
    @Transactional(readOnly = true)
    public List<Appointment> getAllAppointmentByDoctorLogin(int doctorId) {
        return appointmentDAO.getAllAppointmentByDoctorLogin(doctorId);
    }

    @Override
    @Transactional(readOnly = true)
    public Appointment getAppointmentById(int id) {
        return appointmentDAO.getAppointmentById(id);
    }

    @Override
    @Transactional
    public boolean updateCommentStatus(int id, int doctorId, String status) {
        return appointmentDAO.updateCommentStatus(id, doctorId, status);
    }

    @Override
    @Transactional(readOnly = true)
```

AppointmentServiceImpl.java

```java
package com.service;

import java.util.List;

public interface DoctorService {

    boolean registerDoctor(Doctor doctor);

    List<Doctor> getAllDoctors();

    Doctor getDoctorById(int id);

    boolean updateDoctor(Doctor doctor);

    boolean deleteDoctor(int id);

    Doctor login(String email, String password);

    int countAppointment();

    int countUser();

    int countDoctor();

    int countSpecialist();

    int countAppointmentByDoctorId(int doctorId);
}
```

DoctorService.java

```java
package com.service;

import java.util.List;

@Service
public class DoctorServiceImpl implements DoctorService {

    @Autowired
    private DoctorDao doctorDao;

    @Override
    @Transactional
    public boolean registerDoctor(Doctor doctor) {
        return doctorDao.registerDoctor(doctor);
    }

    @Override
    @Transactional(readOnly = true)
    public List<Doctor> getAllDoctors() {
        return doctorDao.getAllDoctor();
    }

    @Override
    @Transactional(readOnly = true)
    public Doctor getDoctorById(int id) {
        return doctorDao.getDoctorById(id);
    }

    @Override
    @Transactional
    public boolean updateDoctor(Doctor doctor) {
        return doctorDao.updateDoctor(doctor);
    }

    @Override
    @Transactional
    public boolean deleteDoctor(int id) {
        return doctorDao.deleteDoctor(id);
    }

    @Override
    @Transactional(readOnly = true)
    public Doctor login(String email, String password) {
```

DoctorServiceImpl.java

```java
package com.service;


import java.util.List;

public interface SpecialistService {
    boolean addSpecialist(String spec);
    List<Specialist> getAllSpecialists();
}
```

SpecialistService.java

```java
package com.service;

import com.dao.SpecialistDAO;

@Service
public class SpecialistServiceImpl implements SpecialistService {

    @Autowired
    private SpecialistDAO specialistDAO;

    @Override
    @Transactional
    public boolean addSpecialist(String spec) {
        return specialistDAO.addSpecialist(spec);
    }

    @Override
    @Transactional
    public List<Specialist> getAllSpecialists() {
        return specialistDAO.getAllSpecialists();
    }
}
```

SpecialistServiceImpl.java

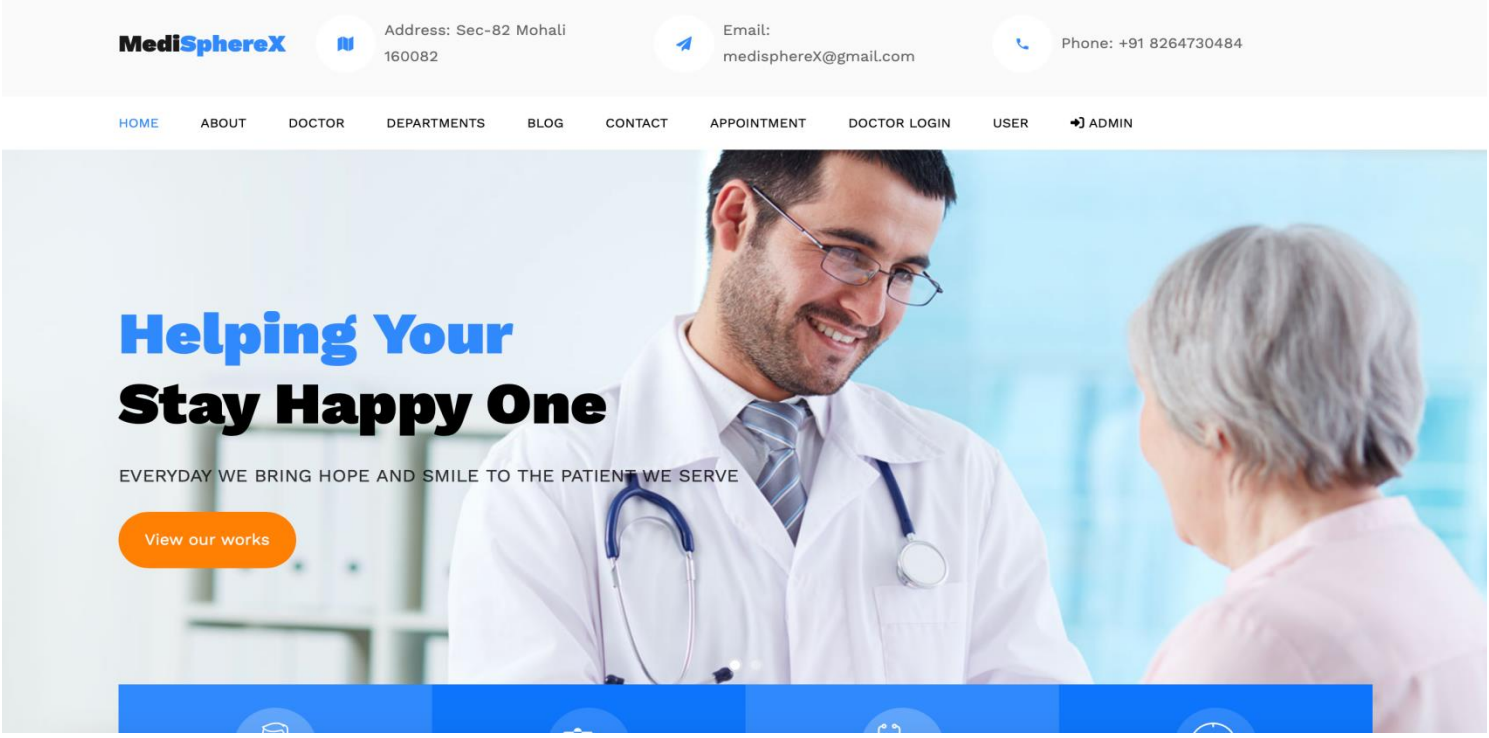
```java
package com.service;
import com.model.User;

public interface UserService {




        boolean saveUser(User user);
        User login(String email, String password);
    }
```

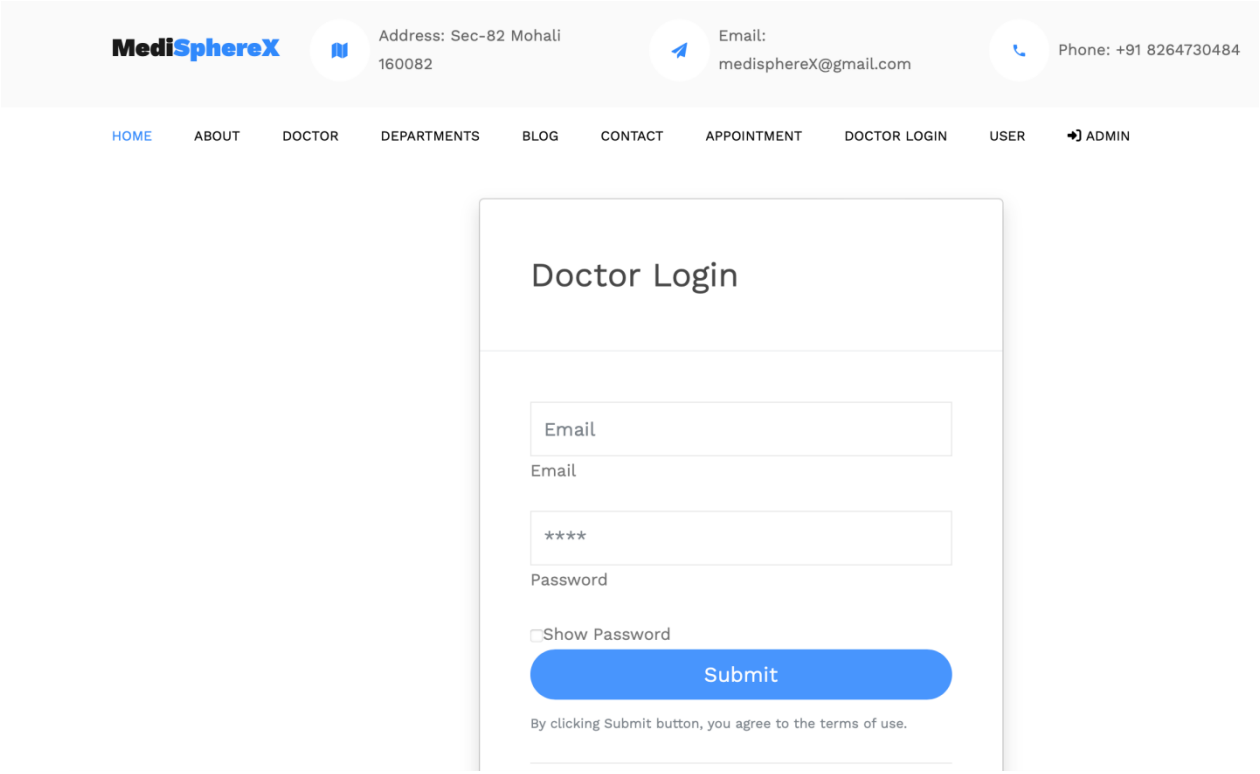
UserService.java

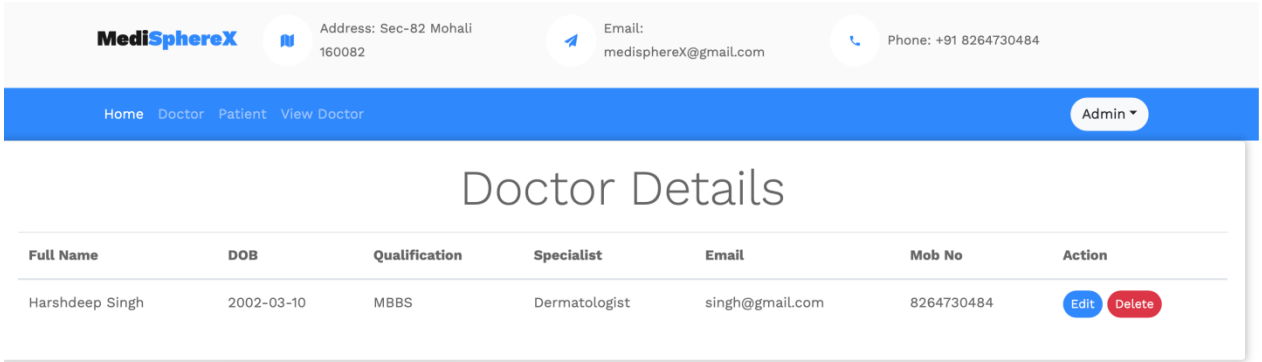
```java
package com.service;
import com.dao.UserDAO;
@Service
public class UserServiceImpl implements UserService {




        @Autowired
        private UserDAO userDAO;

        @Override
        @Transactional
        public boolean saveUser(User user) {
            return userDAO.saveUser(user);
        }

        @Override
        @Transactional(readOnly = true)
        public User login(String email, String password) {
            return userDAO.login(email, password);
        }
    }
```

UserServiceImpl.java

RESULTS


MediSphereX
Address: Sec-82 Mohali 160082
Email: medisphereX@gmail.com
Phone: +91 8264730484
HOME
ABOUT
DOCTOR
DEPARTMENTS
BLOG
CONTACT
APPOINTMENT
DOCTOR LOGIN
USER
ADMIN
Helping Your
Stay Happy One
EVERYDAY WE BRING HOPE AND SMILE TO THE PATIENT WE SERVE
View our works


MediSphereX
Address: Sec-82 Mohali 160082
Email: medisphereX@gmail.com
Phone: +91 8264730484
HOME
ABOUT
DOCTOR
DEPARTMENTS
BLOG
CONTACT
APPOINTMENT
DOCTOR LOGIN
USER
ADMIN
Doctor Login
Email
Email
****
Password
Show Password
Submit
By clicking Submit button, you agree to the terms of use.


MediSphereX
Address: Sec-82 Mohali 160082
Email: medisphereX@gmail.com
Phone: +91 8264730484
Home
Doctor
Patient
View Doctor
Admin
Doctor Details
Full Name
DOB
Qualification
Specialist
Email
Mob No
Action
Harshdeep Singh
2002-03-10
MBBS
Dermatologist
singh@gmail.com
8264730484
Edit
Delete

CHITKARA UNIVERSITY

MediSphereX

Address: Sec-82 Mohali 160082

Email: medisphereX@gmail.com

Phone: +91 8264730484

Home Doctor Patient View Doctor Admin

Admin Dashboard

Doctor
5

User
43

Total Appointment
453

Specialist
34

User Appointment

Full Name

Gender
Male

Age

Appointment Date
16/05/2024

Email

Phone No

Diseases

Doctor
--select--

Full Address

Submit

CHITKARA UNIVERSITY

MediSphereX | Address: Sec-82 Mohali 160082 | Email: medisphereX@gmail.com | Phone: +91 8264730484

Home Doctor Patient View Doctor Admin

Add Doctor

Full Name

DOB

dd-mm-yyyy

Qualification

Specialist

--Select Specialist--

Email

MediSphereX | Address: Sec-82 Mohali 160082 | Email: medisphereX@gmail.com | Phone: +91 8264730484

Home Doctor Patient View Doctor Admin

Patient Details

Full Name	Gender	Age	Appointment	Email	Mob No	Diseases	Doctor Name	Address	Status
Muskaan	female	33	2024-06-17	muskaan@gmail.com	234565425	unknown	Harshdeep Singh	Kakrain Primary school Kakrain	pending
Avinash Mehta	male	22	2024-06-26	avinash@gmail.com	8353023907	unknown	Harshdeep Singh	Sullah	take pain killers
ddtb	male	66	2024-05-19	wdfd@gmail	123333	refer	Harshdeep Singh	refer	pending
harshdeep	male	23	2024-05-21	singh123@gmail.com	9417024654	fever	Harshdeep Singh	hig 1543	pending

MediSphereX | Address: Sec-82 Mohali 160082 | Email: medisphereX@gmail.com | Phone: +91 8264730484

Home Doctor Patient View Doctor Admin

Doctor Details

Full Name	DOB	Qualification	Specialist	Email	Mob No	Action
John Doe	1990-01-01	MBBS	Cardiology	john.doe@example.com	1234567890	Edit Delete
John	1990-01-01	MBBS	Cardiology	john.doe@example.com	1234567890	Edit Delete
John Doe	1985-01-01	MBBS	Cardiologist	johndoe@example.com	1234567890	Edit Delete
Jane Doe	1995-05-05	MBBS	Pediatrician	jane@example.com	0987654321	Edit Delete
Harshdeep Singh	2002-03-10	MBBS	Dentist	singh@gmail.com	8264730484	Edit Delete

CHITKARA UNIVERSITY

MediSphereX Address: Sec-82 Mohali 160082 Email: medisphereX@gmail.com Phone: +91 8264730484

Home Patient Harshdeep Singh

Doctor Dashboard

Doctor
5

Total Appointment
4

MediSphereX Address: Sec-82 Mohali 160082 Email: medisphereX@gmail.com Phone: +91 8264730484

Home Patient Harshdeep Singh

Patient Details

Full Name	Gender	Age	Appointment Date	Email	Mob No	Diseases	Status	Action
harshdeep	male	23	2024-05-21	singh123@gmail.com	9417024654	fever	pending	Comment
ddtb	male	66	2024-05-19	wdfd@gmail	123333	refer	pending	Comment
Avinash Mehta	male	22	2024-06-26	avinash@gmail.com	8353023907	unknown	take pain killers	Comment
Muskaan	female	33	2024-06-17	muskaan@gmail.com	234565425	unknown	pending	Comment

MediSphereX Address: Sec-82 Mohali 160082 Email: medisphereX@gmail.com Phone: +91 8264730484

Home Patient Harshdeep Singh

Patient Comment

Patient Name: harshdeep

Age: 23

Mob No: 9417024654

Diseases: fever

Comment:

Submit

Appointment List

Full Name	Gender	Age	Appoint Date	Diseases	Doctor Name	Status
Avinash Mehta		22	2024-06-26	unknown	Harshdeep Singh	take pain killers
Muskaan		33	2024-06-17	unknown	Harshdeep Singh	pending

CHITKARA UNIVERSITY

Neurology | **Surgical** | **Dental** | **Ophthalmology** | **Cardiology**

- Among first in India to use vasculomimetic stents in the femoral artery
- Tailored treatments that fit your unique needs and preferences
- Dedicated team providing world-class care for children with neurological conditions
- Leading the way in genetic testing and personalized treatment plans
- Specialized units and top-notch expertise for critical neurological conditions

World-Class Neuro Care

DOCTORS

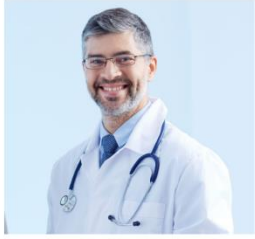
Our Qualified Doctors

Separated they live in. A small river named Duden flows by their place and supplies it with the necessary regelialia. It is a paradisematic country

Dr Sandeep Vaishya

NEUROLOGIST

Renowned Neurologist in India with more than 20 years of experience in the field having worked with some of the top institution and hospitals of India. .



Speciality: Ophthalmologist

OPHTHALMOLOGIST

With over 28 years of experience in practice and a specialization on ocular oncology, Dr. P Vijay Anand Reddy is a well-known and highly skilled ophthalmologist in India.

Dr Aman Popli

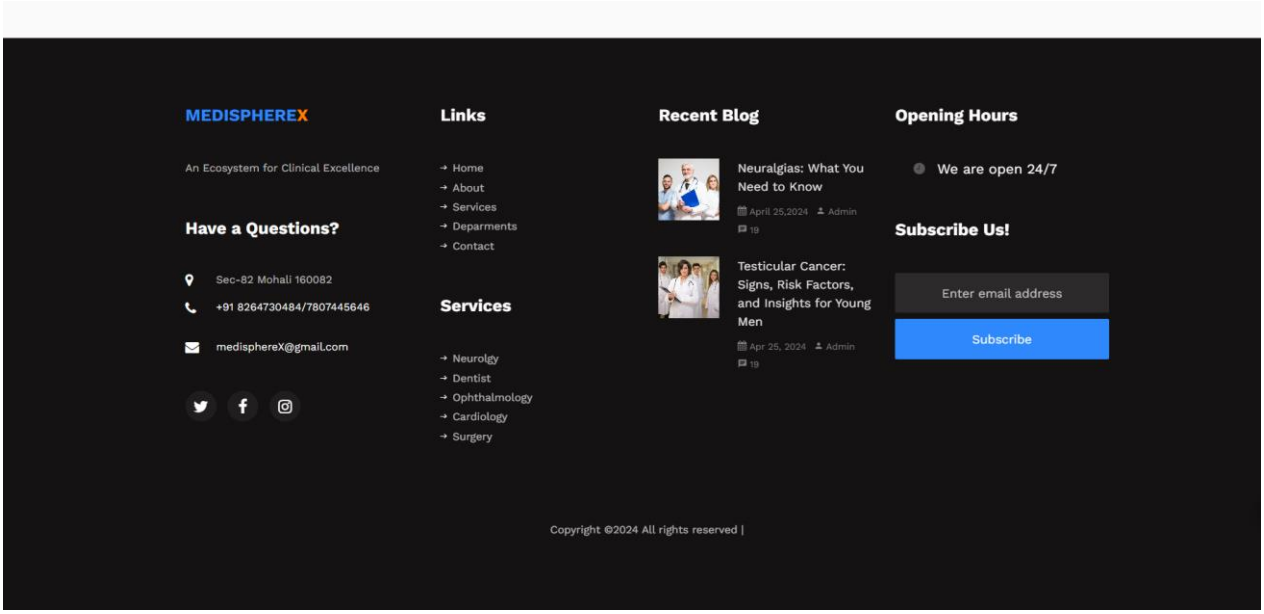
DENTIST

One of the best Prosthodontist and amongst the top 10 dentists in India.

Dr Aathira Ravindranath

PEDIATRICIAN

Paediatric Gastroenterologist MBBS, MD (Pediatrics), DM (Pediatric Gastroenterology) with 2 years of experience



CONCLUSION

Working on the project was an excellent experience. It helped us to understand the importance of planning, designing and implementation so far we have learned in our theory books. It helped us unleashing our creativity while working in a team. It also realized the importance of team working, communication as a part of this project.

The project was successfully completed after a lot of efforts and work hours. This project underwent number of compiling, debugging, removing errors, making it bug free, adding more facilities in Hospital Management System and interactivity making it more reliable and useful.

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the hospital. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

At the end it is concluded that we have made effort on following points...

- A description of the background and context of the project and its relation to work already done in the area.

- Made statement of the aims and objectives of the project.
- The description of Purpose, Scope, and applicability.
- We have defined the problem on which we are working in the project.
- We have described the requirement Specifications of the system and the actions that can be done on these things.

REFERENCES

- Google for problem solving
- https://www.tutorialspoint.com/jdbc/index.htm
- https://www.tutorialspoint.com/java/
- https://www.javatpoint.com/java-tutorial
- https://docs.oracle.com/javase/tutorial/
- https://getbootstrap.com/
- https://www.JSP.net/
- https://www.tutorialspoint.com/mysql/
- https.apache.org/docs/2.0/misc/tutorials.html