Уральский федеральный университет имени первого Президента России Б.Н.Ельцина	Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н.Ельцина» (УрФУ) ИРИТ-РТФ Базовая кафедра «Аналитика больших данных и методы видео анализа»

Grade on project ____________________
Project supervisor ______SAIF M.A.___
Commission's members _____________
____

Defense date _______________________

Report on the project

"Development of web applications"

Student: Anshul Diwakar
Group #: RIM-140930

____
(Signature)

Yekaterinburg
2025

Simple Life – Hospital Management System

Project Report

Contents

Introduction

In today's fast-paced environment, accessing quality healthcare quickly and efficiently is crucial. The goal of this project is to develop a **Hospital Management System (HMS)** called **"Simple Life"**, which simplifies doctor-patient interactions through a clean, responsive, and functional web application.

Simple Life allows **patients to register, view doctor profiles, book appointments**, and **contact** the hospital. Doctors can **upload resumes, consultation pricing, and profile photos**. The platform is designed with a focus on **usability, performance, and minimalism**, while offering all essential hospital management features.

The technologies used in this project include:

- **Frontend:** React.js with Material UI
- **Backend:** Django + Django REST Framework
- **Database:** PostgreSQL
- **Containerization & Deployment:** Docker, Docker Compose
- **Caching & Performance:** Redis

Technical Tasks

The development process was divided into several technical tasks as follows:

- **User Authentication and Authorization:** JWT-based login/logout system with role-based access (Doctor/Patient).
- **Doctor Profile Management:** Doctors can upload photos, specialties, pricing, and resumes.
- **Patient Registration & Booking:** Patients can register, browse doctor profiles, and book appointments.
- **Appointments API:** RESTful endpoints for creating, retrieving, and listing appointments.
- **Contact Messaging System:** Users can submit contact forms/messages.
- **Admin Panel Customization:** Full control via Django Admin for managing users, doctors, patients, and appointments.
- **Responsive UI Design:** Using Material UI and React, all components are mobile-optimized.
- **Dockerization:** Backend and database containerized with Docker.
- **Security & Logging:** Basic validation, error handling, and logging integrated.

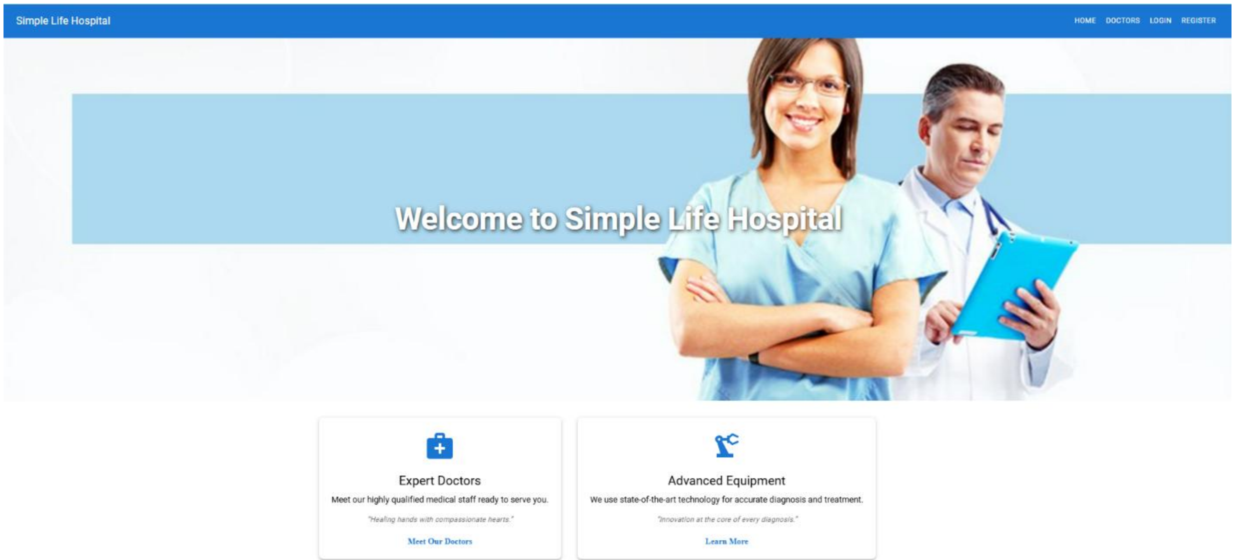
Page Prototypes

Home Page:

The homepage serves as the entry point with the following sections:

- Navigation Bar (Dynamic based on role)
- Hero Section with service slogan
- Featured Doctor Profiles
- How It Works
- Testimonials

- Footer with contact and links



Welcome to Simple Life Hospital

Expert Doctors

Meet our highly qualified medical staff ready to serve you.

"Healing hands with compassionate hearts."

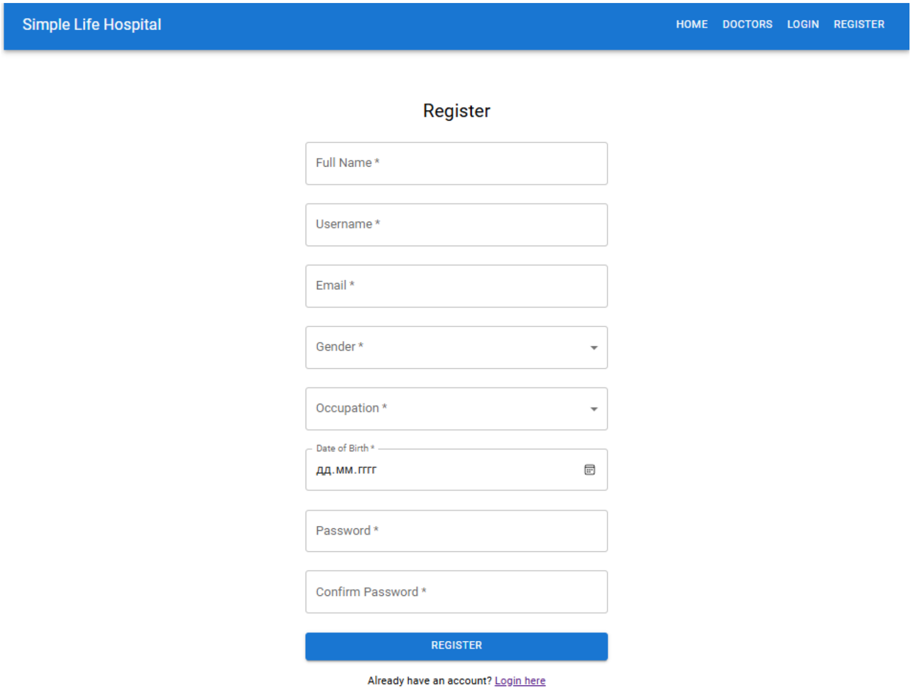
Advanced Equipment

We use state-of-the-art technology for accurate diagnosis and treatment.

"Innovation at the core of every diagnosis."

Fig. 1

Login & Registration: Separate forms for **patients and doctors** to register. Role selection influences field visibility. After login, the UI dynamically changes based on he user role.



Already have an account? Login here

Patient Profile Page:

Patients can:

- View their basic info
- See a list of their upcoming appointments
- Browse available doctors and book appointments

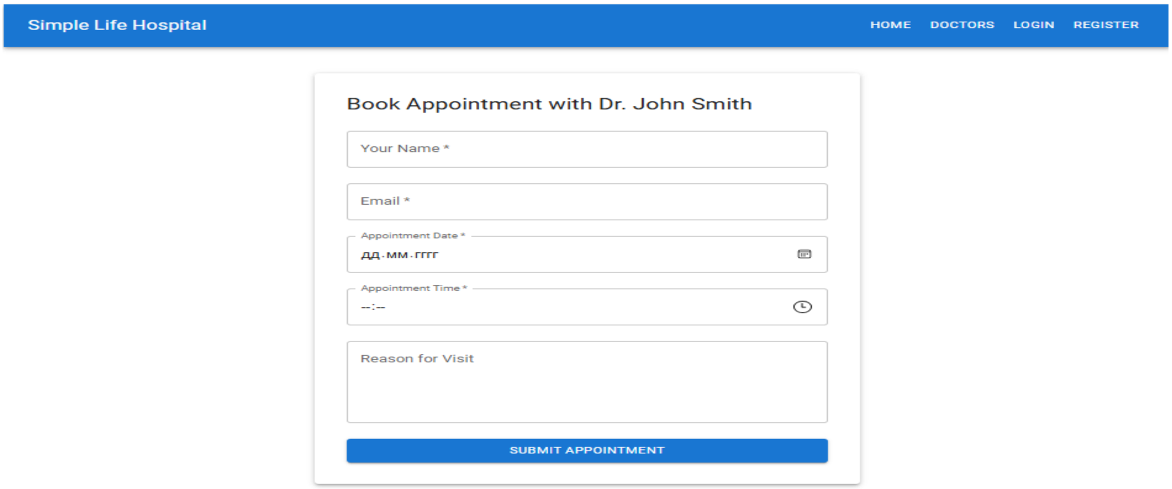
Doctor Profile Page:

Doctors can:

- View their own profile
- See appointments booked with them
- Edit their resume or details

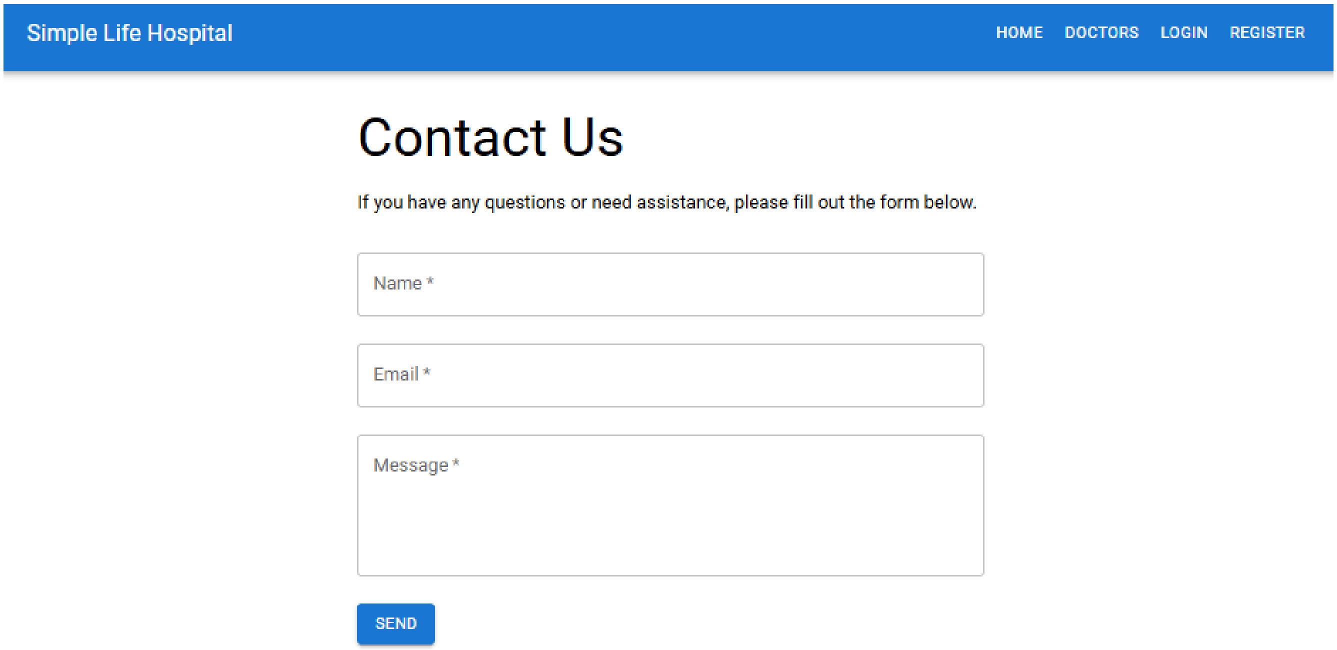
Book Appointment Page:

Patients can choose a doctor from a dropdown, select date & time, and submit a booking.



Contact Page:

Users can leave their name, email, subject, and message. Data is saved in the backend and accessible to admins.



Contact Us

If you have any questions or need assistance, please fill out the form below.

Design Template

The design focuses on:

- **Simplicity:** White background with accent colors for clarity
- **Typography:** "Open Sans" and "Roboto" for modern feel
- **Primary Colors:** Blue (#1976d2), White (#ffffff), Gray (#f5f5f5)
- **Responsiveness:** Layout adapts using CSS Grid/Flexbox and Material UI's Grid system

Navigation Bar & Hero:

[Add your Fig. 7 here: Navigation bar with hero section]

Doctor Cards:

Each doctor's card shows name, specialty, and "Book Now" button with hover effects.

Codes

Backend - Django (Example: Appointment API View):

python

CopyEdit

```
class AppointmentCreateView(generics.CreateAPIView):
    queryset = Appointment.objects.all()
    serializer_class = AppointmentSerializer
    permission_classes = [IsAuthenticated]

    def perform_create(self, serializer):
        serializer.save(patient=self.request.user.patient)
```

This view handles authenticated appointment creation, assigning the logged-in patient.

Frontend - React (Example: Appointment Booking Form):

javascript

CopyEdit

```
const handleSubmit = async () => {
  const response = await axios.post('/api/appointments/', {
    doctor: selectedDoctor,
    date: selectedDate,
    time: selectedTime,
  }, {
    headers: { Authorization: `Bearer ${token}` }
  });

  if (response.status === 201) {
    alert('Appointment booked successfully');
  }
};
```

This form submits appointment data to the backend and alerts the user on success.

Docker Configuration Snippet (docker-compose.yml):

yaml

Copy

```
services:
  backend:
    build: ./backend
    ports:
      - "8000:8000"
    depends_on:
      - db
  db:
    image: postgres
    environment:
      POSTGRES_DB: hospitaldb
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: adminpass
```

Conclusion

This project successfully delivers a **fully functional, role-based hospital management system** using modern web technologies. It meets all functional requirements: doctor registration, patient booking, authentication, appointment scheduling, and contact messaging.

The design prioritizes **usability, accessibility, and simplicity**. All core features were built with attention to security, responsiveness, and code maintainability. The system is containerized and can be deployed efficiently using Docker.