Ural Federal University

named after the first President of Russia B.N.Yeltsin

Second Semester Report

Web application development

"Simple Life - Hospital Management System"

Professor : -SAIF M.A.

Student Name:- Anshul Diwakar

Group: RIM-140930

Technical Task on Development of Full-Stack E-commerce Web Application “Simple Life”

---

1. Description of the Project

Project concept:
A full-stack web application to manage and facilitate online doctor-patient interactions. Doctors can register, list their services, and manage appointments. Patients can register, view doctor profiles, and book appointments.
Target audience:
Doctors and patients in urban areas seeking streamlined healthcare appointments through an online platform.
Marketing channels / user acquisition:
SEO, Google Ads, social media (Instagram, Facebook), doctor referrals, and medical blogs.
Mobile usage:
Yes, responsive design is supported for mobile and tablet devices. Future mobile app using React Native is under consideration.
Business KPIs:

- 500+ appointments/month
- 200+ registered doctors in 3 months
- 80% system uptime
- User retention rate >60% after 3 months

Post-launch support:
Yes, technical support is required for hosting, deployment, and maintenance.

Project Team
Internal customer / Decision maker:
Student/Developer (Yourself)
Customer-side departments involved:
N/A (individual academic project)
Content responsibility:
Contractor (Developer creates content)
Marketing personnel:
N/A
Project deadlines:
Yes, submission required before academic deadline (e.g., June 5, 2025)

Hypotheses of the Project
We expect that our system will primarily be used by working professionals and families in metropolitan areas who prefer booking medical consultations online over traditional methods.

Success Criteria
The project will be considered successful if, within two months of deployment, the system can:

- Successfully handle 1000+ users
- Process 300+ appointment bookings
- Maintain uptime >95%
- Ensure secure role-based access (doctor/patient)

---

2. Sections and Content

Short Description
An online hospital management system where:

- Doctors create profiles, upload resumes, and manage appointment prices.
- Patients can register, view doctors, and make appointments.
- Admins can manage users, doctors, and contact messages.

Market references:
ZocDoc, Practo, HealthTap
Partition structure:
Yes, includes:

- Authentication
- Doctor/Patient Dashboard
- Appointment booking
- Admin Panel

Product integration:
Yes, backend integrates with PostgreSQL, Django Admin, and future third-party APIs (e.g., payment, email).

Content Checklist
Existing content:

- Sample doctor data
- Admin interface
- Basic UI pages

Missing content:

- SEO text
- Marketing media
- Additional doctor profiles

Corporate identity:
No brand guideline; minimal UI based on Material UI and TailwindCSS

Structure of the Project
Section List:
Section 1: Home Page
Displays navigation bar, introductory content, available doctors, and call-to-action to register/login.
Section 2: Doctor Dashboard
Doctor can edit profile, upload resume, set consultation price, and view

appointment bookings.
Section 3: Patient Dashboard
Patient can update profile, view doctors, book and manage appointments.
Section 4: Admin Panel
Django admin for managing users, doctors, contact messages.
Section 5: Contact Page
Patients or visitors can submit a contact form to request support.

---

3. Design Task

References:

- ZocDoc
- Practo

Prototype:
Internal design with Material UI and React

Section Design Descriptions
Section 1: Home Page
Mechanics: Scroll animations, CTA buttons
Design Requirements: Clean layout, mobile-responsive
Section 2: Doctor Dashboard
Mechanics: Editable form with image upload
Design Requirements: Input validation, dropdowns for specialties
Section 3: Patient Dashboard
Mechanics: Appointment calendar, list of doctors
Design Requirements: Display confirmation messages, form validation
Section 4: Admin Panel
Mechanics: Django Admin UI
Design Requirements: N/A

---

4. Development Task

Preview: N/A
Figma / PSD: N/A
Repository:
GitHub https://github.com/AnshulDiwakar/AnshulDiwakar-Simple-Life-Hospital-Management-System.git
Website domain:
https://simplelife.health *(planned)*
Test site:
http://localhost:3000 (frontend)
http://localhost:8000/admin (backend)

General Requirements
Technology Stack:

- Frontend: React + Material UI + TailwindCSS
- Backend: Django + DRF
- DB: PostgreSQL

- Deployment: Docker + Nginx + Gunicorn + Redis

Browser support:
Chrome, Firefox, Edge, Safari (latest 2 versions)
Adaptability (breakpoints):

- 320px (mobile)
- 768px (tablet)
- 1024px (desktop)
- 1440px+ (large desktop)

Languages:

- Frontend: JavaScript/JSX
- Backend: Python
- HTML/CSS for static pages

Hosting:
Dockerized, can run on any Linux VPS or cloud provider (e.g., DigitalOcean, Render, AWS)

Online Store-Specific (N/A for this project)
Product base: N/A
CRM Integration: N/A
Product categories: N/A
Payment system: N/A

Technical Requirements
Hosting requirements:
Docker-compatible environment, PostgreSQL, 2GB RAM minimum
Configuration needs:

- Gunicorn port 8000
- Nginx port 80
- CORS enabled
- CSRF support enabled
- ENV files for secrets

Integration

Systems:

- PostgreSQL
- Redis
- Django Admin
- Potential email services (SendGrid)

Access Data:
Stored in .env (not hard-coded in source)
Protocols:
RESTful API using DRF
Docker Compose for service orchestration

Section Technical Details
Section 1: Home Page

- Static block with intro text and doctor listings
- Interactive elements: CTA buttons, hover effects

Section 2: Doctor Dashboard
- Form-based CRUD
- Image upload
- Appointment viewing logic (API connected)

Section 3: Patient Dashboard
- Search/filter doctors
- Book appointments
- View and cancel bookings

Section 4: Contact Page
- Static form submission
- Sent via POST to DRF endpoint /api/contact/

1. Introduction

This report presents a full-stack web application titled **"Simple Life - Hospital Management System"**, developed as a course project for *Web Application Development*. The aim of this system is to facilitate smooth communication between patients and doctors by providing a digital platform for profile management, appointment booking, and resume uploads.

---

2. Project Objective

The primary objective of this project is to build a scalable, secure, and user-friendly hospital management system that enables:
- Patients to register and book appointments with doctors.
- Doctors to register, upload their resumes, photos, and set consultation prices.
- Admins to manage users and content.
- Real-time functionality and responsive design.

---

3. Technologies Used

- **Frontend**: React.js, Material UI, Axios
- **Backend**: Django, Django REST Framework
- **Database**: PostgreSQL
- **Caching**: Redis
- **Containerization**: Docker, Docker Compose
- **Authentication**: Token-based using Django and DRF
- **Deployment Tools**: Nginx (used via Docker for frontend), Gunicorn (optional for production)

---

4. System Architecture

The system follows a modular microservice-based architecture with the following components:

- **Frontend (React App)**:
  - Runs on port 3000
  - Handles routing, form validation, UI rendering
- **Backend (Django + DRF)**:
  - Runs on port 8000
  - Manages APIs, database operations, authentication, and appointment logic
- **Database (PostgreSQL)**:
  - Stores all persistent data including users, appointments, doctor profiles
- **Redis**:
  - Used for caching and potential queue management
- **Dockerized Environment**:
  - All components containerized and orchestrated using Docker Compose

---

5. Features and Functionality

For Patients:

- User registration and login
- View list of doctors with specialty, photo, and price
- Book appointments with doctors
- View their own profile and appointment history

For Doctors:

- Register as a doctor with additional fields (specialty, consultation price)
- Upload resume and profile photo
- Manage their own profiles

Admin Panel:

- Provided by Django Admin for managing users, appointments, and messages

---

6. Implementation Details

- **Frontend**:
  - Built using React with reusable components
  - Material UI for styling and layout
  - API calls via Axios
- **Backend**:
  - Django models: User, DoctorProfile, PatientProfile, Appointment,

Contact
    - DRF Serializers for validation and transformation
    - Viewsets and Routers for RESTful API structure
- **Authentication**:
    - JWT or Token authentication to maintain sessions
    - Role-based profile rendering based on login

---

7. Deployment and Dockerization

Each major component (frontend, backend, PostgreSQL, Redis) is placed in separate containers using Docker Compose.

- **Frontend Dockerfile** builds React app and serves via Nginx.
- **Backend Dockerfile** handles Django server with migrations and static/media separation.
- **docker-compose.yml** links all services, defines environment variables, and shared volumes.

Deployment is simplified through:

- .env management for secrets
- Static/media file management
- Live preview accessible via browser at localhost:3000

---

8. Conclusion

The Simple Life project showcases a complete, secure, and efficient hospital management system tailored for modern healthcare interaction. It demonstrates practical knowledge of frontend/backend integration, RESTful APIs, authentication, and containerized deployment.

---

9. Future Improvements

- Add email/SMS notification system for appointment reminders
- Implement online payment system for doctor consultations
- Add calendar view for scheduling
- Add admin dashboard with analytics
- Deploy to cloud hosting platforms like AWS or Heroku

Git- https://github.com/AnshulDiwakar/AnshulDiwakar-Simple-Life-Hospital-Management-System.git