PROJECT REPORT

ON

HEALTHCARE MANAGEMENT SYSTEM USING AGILE METHODOLOGY

Submitted By:

Nikhil Harwani

1. ABSTRACT

The rapid digitization of the healthcare sector demands efficient systems for managing patient data, appointments, and medical records. This project report outlines the development of a comprehensive Healthcare Management System (HMS) utilizing the Agile software development methodology. The primary objective is to create a seamless interface that bridges the gap between patients, doctors, and hospital administration.

By adopting the Agile Scrum framework, the development team effectively handled dynamic requirements such as telemedicine integration and patient privacy regulations (HIPAA). The system features modules for Appointment Scheduling, Electronic Health Records (EHR), Doctor Prescriptions, and Billing Management.

This report documents the software development lifecycle, emphasizing how iterative development cycles enabled the rapid delivery of critical features. The resulting application improves operational efficiency, reduces manual errors in medical records, and enhances the overall patient experience.

2. INTRODUCTION

2.1 Introduction

A Healthcare Management System is a web-based application designed to automate the administrative and clinical operations of hospitals and clinics. It serves as a central repository for patient information, appointment histories, and diagnostic reports. This project focuses on building a secure HMS using the Java Spring Boot framework and React.js, adhering to Agile principles.

2.2 Problem Identification

Traditional manual healthcare management faces significant challenges:

- **Data Fragmentation:** Patient records are stored in physical files, leading to loss or damage.
- **Inefficient Scheduling:** Phone-based booking leads to double-booking and long waiting times.
- **Lack of Accessibility:** Doctors cannot access patient history remotely during emergencies.
- **Billing Errors:** Manual calculation of invoices often results in discrepancies.

2.3 Need of the Project

To address these issues, the proposed system aims to:

1. Digitize all patient records into a secure Electronic Health Record (EHR) database.
2. Provide an online portal for patients to book and view appointments.
3. Enable doctors to prescribe medicines and view lab reports digitally.
4. Automate billing and generate financial reports for administration.

2.4 Project Scheduling (Agile Timeline)

The project was executed in four 2-week sprints:

Phase	Duration	Deliverable
Sprint 1	2 Weeks	Requirement Analysis, DB Schema, User Authentication (Admin/Doctor/Patient)
Sprint 2	2 Weeks	Patient Registration Module, Appointment Booking System
Sprint 3	2 Weeks	Doctor Dashboard, Prescription Module, EHR Integration
Sprint 4	2 Weeks	Billing Module, System Testing, Deployment

2.5 Objectives

- To develop a responsive healthcare platform for managing hospital operations.
- To implement Role-Based Access Control (RBAC) for data security.
- To streamline the appointment booking process to reduce wait times.
- To ensure data integrity and backup for critical medical records.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Purpose

This SRS document defines the functional and non-functional requirements of the Healthcare System. It serves as a contract between the stakeholders and developers, ensuring the final product meets clinical and administrative needs.

3.2 Scope

The scope includes a **Patient Portal** (Booking, History), a **Doctor Portal** (Consultations, Prescriptions), and an **Admin Portal** (Staff Mgmt, Inventory). It does not currently cover insurance claims processing or ambulance dispatch tracking.

3.3 System Requirements

Hardware Requirements (Server)

- **Processor:** Intel Core i7 or equivalent
- **RAM:** 16 GB (for handling concurrent users)
- **Storage:** 500 GB SSD (for medical records/images)
- **Network:** Secure SSL/TLS enabled connection

Software Requirements

- **OS:** Windows Server or Linux (Ubuntu)
- **Frontend:** React.js, Bootstrap
- **Backend:** Java Spring Boot / Node.js
- **Database:** MySQL or PostgreSQL
- **Browser:** Chrome, Edge, or Firefox

3.4 Tools Used

- **IntelliJ IDEA / VS Code:** Development Environment.
- **GitHub:** Version Control.

- **Trello:** Kanban Board for Agile Tasks.
- **Selenium:** Automated Testing.

3.5 Software Process Model

Agile Scrum: The project utilizes the Agile Scrum model. In healthcare, requirements often change based on regulatory updates or staff feedback. Agile allows for iterative testing, ensuring that critical features like patient data security are validated frequently throughout the lifecycle.

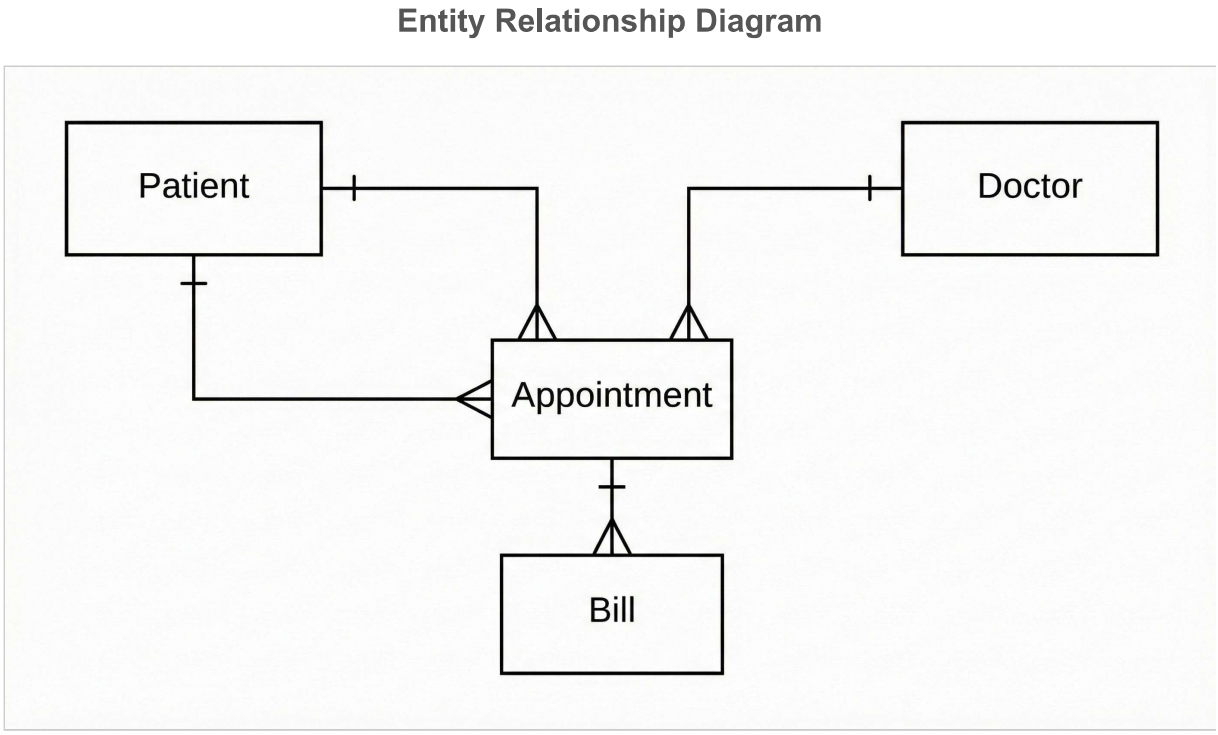
4. SYSTEM DESIGN

4.1 Data Dictionary

Table: Appointments

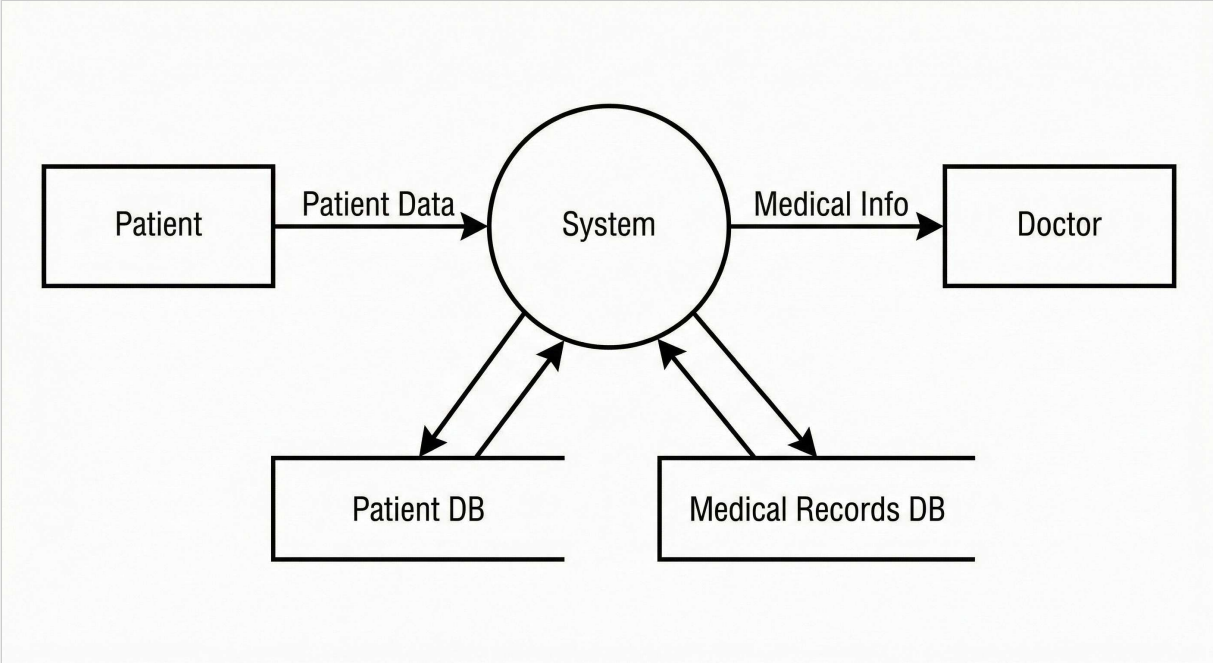
Field	Type	Description
appt_id	INT (PK)	Unique Appointment ID
patient_id	INT (FK)	Reference to Patient
doctor_id	INT (FK)	Reference to Doctor
appt_date	DATETIME	Scheduled Time
status	ENUM	'Scheduled', 'Completed', 'Cancelled'

4.2 ER Diagram



4.3 Data Flow Diagram (DFD)

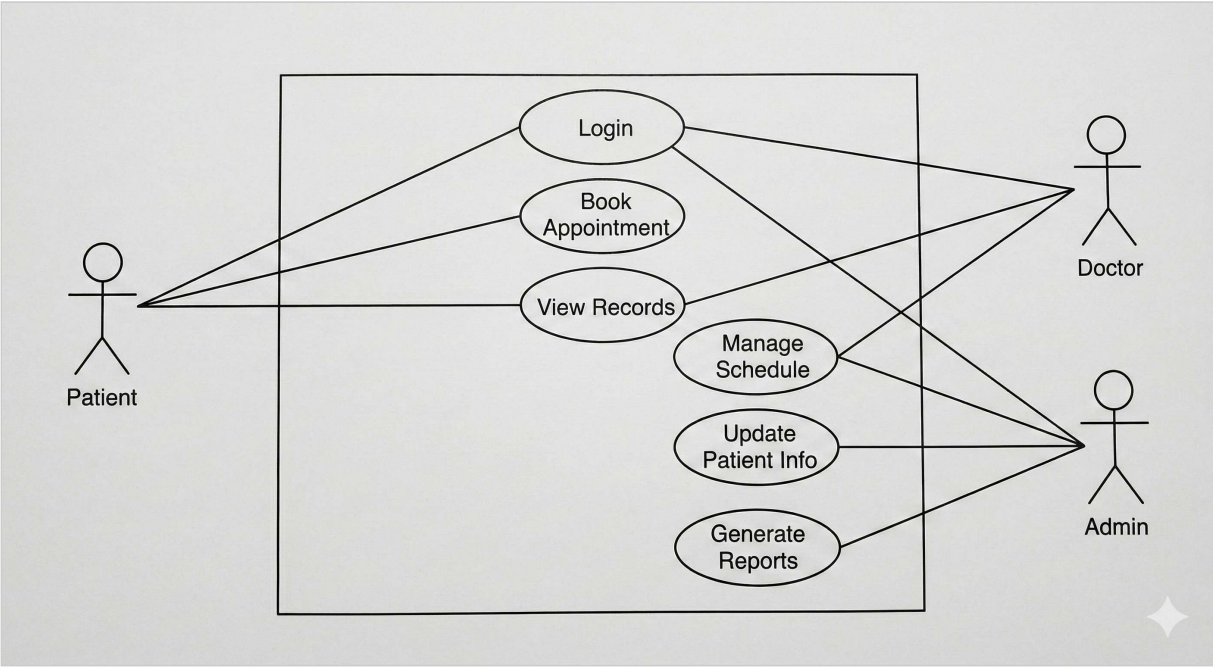
Data Flow Diagram (Level 1)



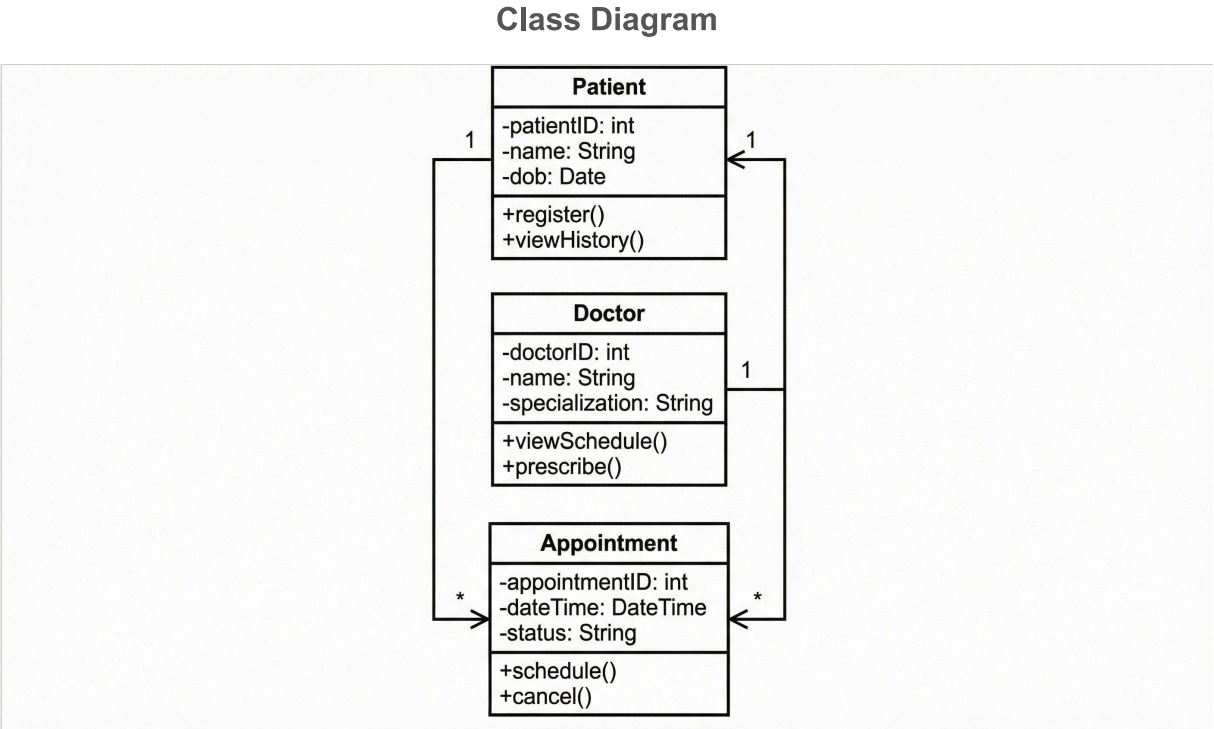
4.4 UML Diagrams

Use Case Diagram

Use Case Diagram



Class Diagram



5. IMPLEMENTATION

5.1 Program Code

Below are core snippets of the backend logic for appointment management.

Backend: Appointment Controller

```
// AppointmentController.java
@RestController
@RequestMapping("/api/appointments")
public class AppointmentController {

    @Autowired
    private AppointmentService service;

    @PostMapping("/book")
    public ResponseEntity bookAppointment(@RequestBody AppointmentRequest
req) {
        // 1. Check Doctor Availability
        if (!service.isSlotAvailable(req.getDoctorId(), req.getDate())) {
                        return  ResponseEntity.badRequest().body("Slot  not
available");
        }

        // 2. Create Appointment
        Appointment appt = new Appointment();
        appt.setPatientId(req.getPatientId());
        appt.setDoctorId(req.getDoctorId());
        appt.setDate(req.getDate());
        appt.setStatus("SCHEDULED");

        return ResponseEntity.ok(service.save(appt));
    }
}
```

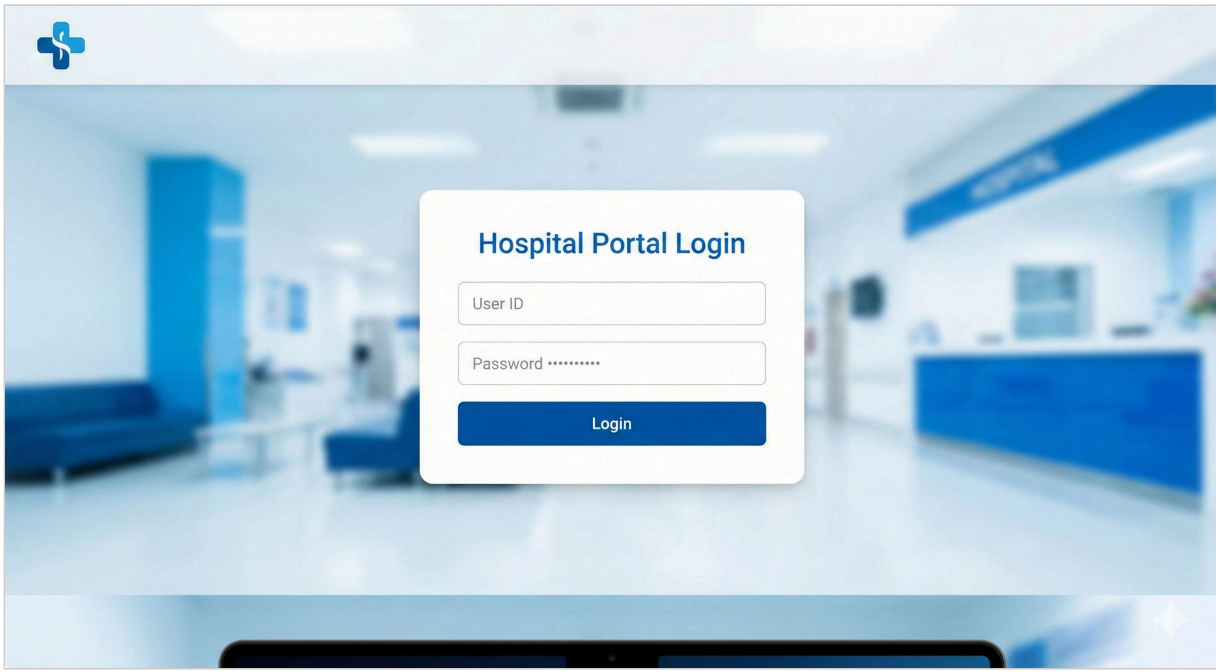
Frontend: Doctor Selection

```
// components/DoctorSelect.js
import React, { useState, useEffect } from 'react';

const DoctorSelect = ({ onSelect }) => {
    const [doctors, setDoctors] = useState([]);

    useEffect(() => {
        fetch('/api/doctors')
            .then(res => res.json())
            .then(data => setDoctors(data));
    }, []);

    return (
        -- Select Specialist --
    );
};
```

5.2 Output Screens

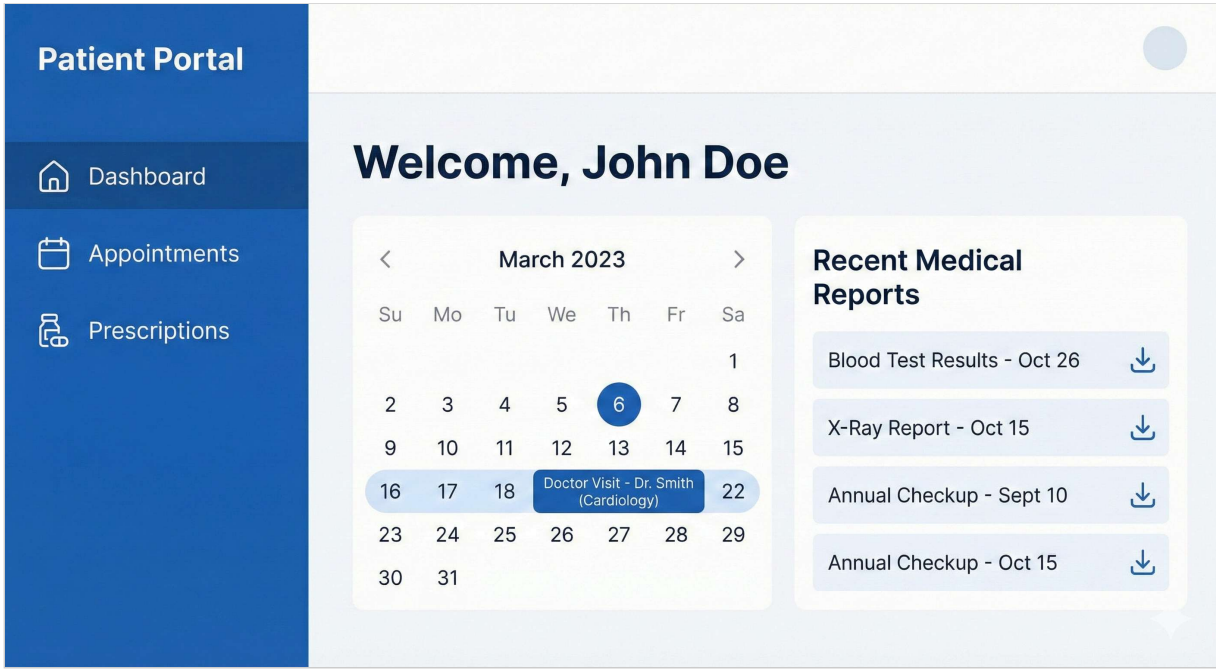
Login Screen:

Login Interface Screenshot



Patient Dashboard:

Dashboard Screenshot



6. TESTING

6.1 Test Data & Strategy

The testing strategy focused on Black Box Testing to validate user workflows (e.g., booking an appointment). Security testing was also conducted to ensure patient data is not accessible to unauthorized users.

6.2 Test Results

Test Case ID	Description	Input Data	Expected Output	Status
TC_AUTH_01	Valid Patient Login	User: pat01 / Pass: 1234	Dashboard Access	PASS
TC_APPT_01	Book Slot	Dr. Smith, 10:00 AM	Confirmation SMS/Email	PASS
TC_APPT_02	Double Booking	Dr. Smith, 10:00 AM (Already booked)	Error: "Slot Unavailable"	PASS
TC_SEC_01	Access Restricted Page	Patient trying to access Admin URL	Access Denied / 403	PASS

7. USER MANUAL

7.1 How to use project guidelines

1. **Registration:** New patients must register with personal and medical history details.
2. **Login:** Use the unique Patient ID to log in.
3. **Book Appointment:** Navigate to 'New Appointment', select Department > Doctor > Time Slot.
4. **View Reports:** Go to 'Medical Records' to download lab results.

7.2 Screen Layouts and Description

Home Page: Emergency contact numbers and login options for Patients/Doctors.
Doctor Dashboard: View today's appointments and patient history cards.
Admin Panel: Manage staff, inventory, and billing reports.

8. PROJECT APPLICATIONS AND LIMITATIONS

Applications

- **Hospitals & Clinics:** Managing daily OPD and IPD operations.
- **Telemedicine:** Can be extended for remote video consultations.
- **Pharmacies:** Integrated inventory management for medicines.

Limitations

- The system requires high-speed internet for real-time synchronization.
- Integration with legacy machinery (old X-Ray machines) is not supported.
- Mobile application is currently in development (Web-only for now).

9. CONCLUSION AND FUTURE ENHANCEMENT

The "Healthcare Management System" project successfully demonstrated the value of Agile development in building critical care applications. By delivering features incrementally, the team ensured that the core functionality of booking and patient management was stable before adding complex features like billing. The system effectively solves the problem of data fragmentation.

Future Enhancements:

- **AI Diagnosis:** Integrating ML models to suggest preliminary diagnoses based on symptoms.
- **IoT Integration:** Connecting with wearable devices to monitor patient vitals in real-time.
- **Chatbot:** AI-powered bot for answering patient queries regarding timings and doctors.

10. BIBLIOGRAPHY & REFERENCES

Books:

- Sommerville, I. (2015). *Software Engineering*. Pearson Education.
- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill.

Websites:

- Spring Boot Docs (https://spring.io/projects/spring-boot)
- React.js Official Site (https://reactjs.org/)
- HIPAA Guidelines (https://www.hhs.gov/hipaa)