🩺 **Phase 6: User Interface Development (Hospital Appointment Management System)**

### **Platform:** Salesforce (Admin + Developer – Aura + Apex)

## Overview

In Phase 6, we develop a **user-friendly interface** to manage hospital appointments. This includes:

* Viewing appointments in a table
* Confirming or rescheduling appointments
* Selecting doctor and date filters
* Integrating with Apex backend logic

The goal is to give doctors, patients, and receptionists an **intuitive, interactive UI** for appointment management.

## 1️⃣ Apex Controller

**Purpose:** Handle backend logic for fetching appointments and updating status.

**Apex Class: DoctorAvailabilityController**

public with sharing class DoctorAvailabilityController {  
  
 @AuraEnabled  
 public static List<Appointment\_\_c> getDoctorAppointments(Id doctorId, Date appointmentDate) {  
 return [SELECT Id, Appointment\_Time\_\_c, Appointment\_End\_Time\_\_c, Status\_\_c  
 FROM Appointment\_\_c  
 WHERE Doctor\_\_c = :doctorId  
 AND Appointment\_Date\_\_c = :appointmentDate  
 ORDER BY Appointment\_Time\_\_c];  
 }  
  
 @AuraEnabled  
 public static void updateAppointmentStatus(Id appointmentId, String newStatus) {  
 Appointment\_\_c app = [SELECT Id, Status\_\_c FROM Appointment\_\_c WHERE Id = :appointmentId LIMIT 1];  
 app.Status\_\_c = newStatus;  
 update app;  
 }  
}

**Explanation:**

* @AuraEnabled → allows Aura/LWC components to call this method
* getDoctorAppointments → fetches appointments for selected doctor & date
* updateAppointmentStatus → updates the Status field when user clicks Confirm or Reschedule

## 2️⃣ Aura Component

**Component Name:** appointmentList

**Purpose:** Display appointments, allow Confirm/Reschedule, filter by doctor & date.

### 2a — Component Markup (appointmentList.cmp)

<aura:component controller="DoctorAvailabilityController" implements="flexipage:availableForAllPageTypes,force:appHostable">  
 <aura:attribute name="appointments" type="Appointment\_\_c[]"/>  
 <aura:attribute name="selectedDoctor" type="Id"/>  
 <aura:attribute name="selectedDate" type="Date"/>  
 <aura:attribute name="availabilityMessage" type="String"/>  
 <aura:attribute name="doctorOptions" type="List"/>  
  
 <aura:handler name="init" value="{!this}" action="{!c.doInit}"/>  
  
 <lightning:card title="Doctor Appointments">  
 <div class="slds-p-around\_medium">  
 <lightning:combobox  
 name="doctor"  
 label="Select Doctor"  
 value="{!v.selectedDoctor}"  
 placeholder="Select Doctor"  
 options="{!v.doctorOptions}"  
 onchange="{!c.handleDoctorChange}" />  
  
 <lightning:input  
 type="date"  
 label="Select Date"  
 value="{!v.selectedDate}"  
 onchange="{!c.handleDateChange}" />  
  
 <p>{!v.availabilityMessage}</p>  
  
 <table class="slds-table slds-table\_cell-buffer slds-table\_bordered">  
 <thead>  
 <tr>  
 <th>Time</th>  
 <th>Status</th>  
 <th>Action</th>  
 </tr>  
 </thead>  
 <tbody>  
 <aura:iteration items="{!v.appointments}" var="app">  
 <tr>  
 <td>{!app.Appointment\_Time\_\_c} - {!app.Appointment\_End\_Time\_\_c}</td>  
 <td>{!app.Status\_\_c}</td>  
 <td>  
 <lightning:button  
 label="Confirm"  
 name="{!app.Id}"  
 onclick="{!c.confirmAppointment}" />  
 <lightning:button  
 label="Reschedule"  
 name="{!app.Id}"  
 onclick="{!c.rescheduleAppointment}" />  
 </td>  
 </tr>  
 </aura:iteration>  
 </tbody>  
 </table>  
 </div>  
 </lightning:card>  
</aura:component>

### 2b — Controller JS (appointmentListController.js)

({  
 doInit : function(component, event, helper) {  
 helper.loadDoctors(component);  
 },  
  
 handleDoctorChange : function(component, event, helper) {  
 helper.loadAppointments(component);  
 },  
  
 handleDateChange : function(component, event, helper) {  
 helper.loadAppointments(component);  
 },  
  
 confirmAppointment : function(component, event, helper) {  
 var appointmentId = event.getSource().get("v.name");  
 helper.updateStatus(component, appointmentId, 'Scheduled');  
 },  
  
 rescheduleAppointment : function(component, event, helper) {  
 var appointmentId = event.getSource().get("v.name");  
 helper.updateStatus(component, appointmentId, 'Reschedule');  
 }  
})

### 2c — Helper JS (appointmentListHelper.js)

({  
 loadDoctors : function(component) {  
 var doctorOptions = [  
 {label: 'Dr. Smith', value: '005XXXXXXXXXXXX'},  
 {label: 'Dr. John', value: '005YYYYYYYYYYYY'}  
 ];  
 component.set("v.doctorOptions", doctorOptions);  
 },  
  
 loadAppointments : function(component) {  
 var action = component.get("c.getDoctorAppointments");  
 action.setParams({  
 doctorId: component.get("v.selectedDoctor"),  
 appointmentDate: component.get("v.selectedDate")  
 });  
  
 action.setCallback(this, function(response) {  
 var state = response.getState();  
 if(state === "SUCCESS") {  
 component.set("v.appointments", response.getReturnValue());  
 component.set("v.availabilityMessage", response.getReturnValue().length > 0 ? "Appointments loaded." : "No appointments. Doctor available.");  
 } else {  
 component.set("v.availabilityMessage", "Error loading appointments.");  
 }  
 });  
  
 $A.enqueueAction(action);  
 },  
  
 updateStatus : function(component, appointmentId, newStatus) {  
 var action = component.get("c.updateAppointmentStatus");  
 action.setParams({  
 appointmentId: appointmentId,  
 newStatus: newStatus  
 });  
  
 action.setCallback(this, function(response) {  
 if(response.getState() === "SUCCESS") {  
 this.loadAppointments(component);  
 } else {  
 console.log("Error updating appointment status");  
 }  
 }.bind(this));  
  
 $A.enqueueAction(action);  
 }  
})

## 3️⃣ Lightning App Builder Integration

**Steps:**

1. Go to **Setup → Lightning App Builder → New Page**
2. Choose **App Page / Home Page / Record Page**
3. Drag the **appointmentList component** to the page
4. Save and **Activate**

.