Smart Hospital Appointment CRM Project  **Salesforce Project Implementation Phases with Concepts (Admin + Developer)**

**Phase 1: Problem Understanding & Industry Analysis**

**Goal:** Define scope, success metrics, constraints and stakeholder needs.

* **Requirement Gathering (deliverables):**
  + Functional: Track Doctors, Patients, Appointments, Prescriptions, Bills; prevent overlapping appointments; appointment notifications; automated approval for high-value surgeries; reports/dashboards.
  + Non-functional: Response time SLAs for booking, data retention policy, audit logging, HIPAA-like considerations for PII.
  + Acceptance criteria: e.g., “No double-booking for same Doctor in same timeslot”, “Surgery costing > ₹50,000 routes to manager approval”.
* **Stakeholder Analysis (roles & expectations):**
  + Admin — org config, users, security.
  + Receptionist — create/manage appointments, patient check-in.
  + Doctor — view schedule, update prescriptions.
  + Manager — approve surgeries, review KPIs.
  + Patient — self-booking, view prescriptions, pay bills.
* **Business Process Mapping (visual + text):**
  + Map flows with swimlanes: Patient → Reception → System checks → (Approval?) → Notifications → Service → Billing → Follow-up.
  + Identify exceptions: emergency walk-ins, reschedules, cancellations, no-shows.
* **Industry-specific Use Cases & Constraints:**
  + Emergency slots, follow-up scheduling rules (e.g., follow-up within X days), blocked slots for surgeries, pre-op approval, insurance pre-authorization.
* **AppExchange Exploration:**
  + Search for hospital scheduling / healthcare CRM packages — note integrations, features you can reuse vs build.

**Phase 2: Org Setup & Configuration**

**Goal:** Prepare a secure, testable Salesforce environment.

* **Salesforce Editions & Org Choice**
  + Use Developer Edition or a Trial Enterprise for dev; use Full sandbox for system testing; Partial/Full sandboxes for integration/regression.
* **Company Profile Setup**
  + Company information, default locale, default currency, business hours (e.g., 09:00–18:00), timezones for branches.
* **Business Hours & Holidays**
  + Create BusinessHours record for hospital; configure Holiday object records for holidays.
* **Fiscal Year**
  + Set to Standard Jan–Dec (if finance reports follow calendar year).
* **User Setup & Licenses**
  + Create test users for Receptionist, Doctor, Manager, Admin. Name conventions: [first.last@hospital.test](mailto:first.last@hospital.test).
  + Assign appropriate licenses (Salesforce Platform / Sales/Service).
* **Profiles**
  + Receptionist: object-level read/create on Appointment and Patient, restricted on sensitive fields.
  + Doctor: view/edit own Appointments, read Prescriptions.
  + Manager: full access.
  + Admin: full system permissions.
* **Roles**
  + Manager (top) → Doctor → Receptionist. Use sharing based on role hierarchy where appropriate.
* **Permission Sets**
  + Reports\_Viewer: give report + dashboard access to receptionists or other roles temporarily.
* **Org-Wide Defaults (OWD)**
  + Patients: Private.
  + Appointments: Private.
  + Prescriptions: Controlled by parent (Appointment).
  + Bills: Private.
* **Sharing Rules**
  + Criteria-based rule: share Appointment with Doctor role when Appointment.Doctor\_\_c = User.Contact or via Owner-based sharing. Consider Apex-managed sharing for complex rules.
* **Login Access Policies**
  + Set profile login hours (Receptionist only 08:00–20:00) and IP restrictions for sensitive roles if needed.
* **Dev Org Setup & Sandboxes**
  + Develop in Developer Sandbox -> push to Full sandbox for integrated testing -> Production via Change Sets or SFDX.
* **Deployment Basics**
  + Use SFDX for source control, metadata deployment. Keep a release checklist (validation, test coverage, backup).

**Phase 3: Data Modeling & Relationships**

**Goal:** Define objects, fields, and relationships.

* **Custom Objects & Suggested Fields**
  + Patient\_\_c: Name (standard), DOB (Date), Age (Formula), Contact\_Number\_\_c (Phone), Email\_\_c, Medical\_History\_\_c (LongText), Insurance\_\_c (Lookup to Account or custom).
  + Doctor\_\_c: Name (standard), Specialty\_\_c (Picklist), Contact\_\_c (Phone), Availability\_\_c (Text or structured), Active\_\_c (Checkbox).
  + Appointment\_\_c: Patient\_\_c (Lookup to Patient), Doctor\_\_c (Lookup to Doctor), Start\_Time\_\_c (DateTime), End\_Time\_\_c (DateTime), Type\_\_c (Picklist: Checkup,Surgery,Emergency), Status\_\_c (Picklist), Reason\_\_c (Text), Estimated\_Cost\_\_c (Currency).
  + Prescription\_\_c: Appointment\_\_c (Lookup), Medicine\_Name\_\_c, Dosage\_\_c, Frequency\_\_c (Text).
  + Bill\_\_c: Appointment\_\_c (Lookup), Amount\_\_c (Currency), Payment\_Status\_\_c (Picklist).
* **Field Types & Examples**
  + Estimated\_Cost\_\_c — Currency(16,2)
  + Start\_Time\_\_c / End\_Time\_\_c — DateTime
  + Medical\_History\_\_c — Long Text Area (32000)
* **Record Types**
  + Appointment\_\_c record types: Checkup, Surgery. Different page layouts and required fields (e.g., Surgery requires pre-op consent, estimated cost).
* **Page Layouts**
  + Appointment layout shows related lists: Prescriptions, Bills, Notes. Make fields required where needed.
* **Compact Layouts**
  + Show Patient, Doctor, Start\_Time, Status for mobile quick glance.
* **Schema Builder**
  + Create a visual diagram for interviews and presentation.
* **Relationship types**
  + Use Lookup for Appointment→Patient and Appointment→Doctor (so patient deletion can be controlled).
  + Prescription and Bill → Master-Detail to Appointment if you want cascade deletes; otherwise Lookup.
* **Junction Objects**
  + If an appointment can have multiple doctors (e.g., multi-specialist consult), build Appointment\_Doctor\_\_c junction.
* **External Objects**
  + Use for linked hospital billing DB or insurance system (Salesforce Connect).

**Phase 4: Process Automation (Admin)**

**Goal:** Automate routine tasks, enforce business rules.

* **Validation Rules**
  + Example: End must be after Start:
  + End\_Time\_\_c <= Start\_Time\_\_c

Error message: “End time must be after start time.”

* **Overlapping Appointment Prevention**
  + Preferred: Record-Triggered Flow (Before Save) or Apex Trigger.
  + Flow logic pseudocode:
    1. On create/update Appointment (Before Save).
    2. Query for existing appointments for same Doctor\_\_c where Id != $Record.Id and  
       Start\_Time\_\_c < $Record.End\_Time\_\_c AND End\_Time\_\_c > $Record.Start\_Time\_\_c.
    3. If any found → add error to record "Overlapping appointment exists".
* **Screen Flow**
  + Build a step-by-step booking form for Receptionist or patient self-booking (choose Doctor → pick available slot → confirm).
* **Approval Process**
  + Criteria: Appointment.Type\_\_c = 'Surgery' && Estimated\_Cost\_\_c > 50000
  + Approver: Manager role or specific user queue.
  + Actions: Lock record while pending, email to approver, upon approval auto-update status and notify parties.
* **Email Alerts**
  + Templates (merge fields): Appointment confirmation to patient; reminder 24 hours before; doctor reminder 1 hour before.
  + Use Time-based Action (Scheduled Path in Flow) or Scheduled Flow.
* **Field Updates**
  + Example: Auto-set Appointment.Status\_\_c = 'Checked-in' when Receptionist marks arrived.
* **Tasks**
  + Create Task to prepare patient file when surgery scheduled and approved.
* **Custom Notifications**
  + Use Salesforce Mobile Push or In-App Notifications to doctors (e.g., 30 minutes before).

**Phase 5: Apex Programming (Developer)**

**Goal:** Implement complex logic, reusable services, and integrations.

* **Apex Trigger: Prevent Overlap (example outline)**
* trigger AppointmentTrigger on Appointment\_\_c (before insert, before update) {
* if (Trigger.isBefore) {
* AppointmentService.preventOverlaps(Trigger.new);
* }
* }
* **Apex Class: AppointmentService (pseudo)**
* public with sharing class AppointmentService {
* public static void preventOverlaps(List<Appointment\_\_c> appts) {
* // collect doctor ids & time ranges
* // query existing appointments with SOQL using IN and time filters
* // check overlap condition: existing.Start < new.End && existing.End > new.Start
* // addError on conflicting record
* }
* public static List<Doctor\_\_c> findAvailableDoctors(DateTime start, DateTime end, String specialty) {
* // SOQL to filter doctors not having overlapping appointments
* }
* }
* **SOQL Example: find overlapping**
* SELECT Id, Start\_Time\_\_c, End\_Time\_\_c FROM Appointment\_\_c
* WHERE Doctor\_\_c = :doctorId
* AND Id != :currentId
* AND Start\_Time\_\_c < :newEnd
* AND End\_Time\_\_c > :newStart
* **Collections**
  + Use Maps keyed by DoctorId to group appointments efficiently.
* **Asynchronous Apex**
  + Batch Apex: mark missed appointments nightly and send follow-up.
  + Queueable: perform billing calculations or callouts.
  + Future/Platform Events: async insurance API callouts.
* **Scheduled Apex**
  + Daily summary to manager script:
  + global class DailySummarySched implements Schedulable { ... }
* **Exception Handling**
  + Catch exceptions, log to custom Error\_Log\_\_c object, and provide user-friendly messages.
* **Test Classes**
  + Create test patients, doctors, create appointments covering:
    - Successful booking
    - Overlap rejection
    - Approval flow triggering
    - Async job invocation (use Test.startTest()/Test.stopTest()).
  + Use @isTest methods and assert expected results. Aim 75%+ coverage.

**Phase 6: User Interface Development**

**Goal:** Build intuitive UIs for staff and patients.

* **Lightning App Builder**
  + App: “Hospital CRM” with tabs: Patients, Doctors, Appointments, Prescriptions, Bills.
  + Home page components: Today’s Appointments, Quick Actions (Book Appointment), KPI cards.
* **Record Pages**
  + Appointment Record Page: related lists (Prescriptions, Bills), quick action to create Prescription and Bill.
* **Tabs**
  + Create tab for each custom object; group for easy navigation.
* **Home Page Layouts**
  + Manager Home: chart components (Revenue by treatment).
  + Doctor Home: upcoming appointments list filtered by current user.
* **Utility Bar**
  + Add “Quick Patient Search” and “Book Appointment” components.
* **LWC (Lightning Web Component) — Example Components**
  + doctorSearch — search doctors by specialty and availability (uses wire to Apex).
  + appointmentBooking — screen to pick slot and call Apex to create appointment.
  + prescriptionHistory — display list of prescriptions for logged-in patient.
* **LWC Implementation Notes**
  + Wire adapter example:
  + @wire(getAvailableDoctors, { start: '$start', end: '$end', specialty: '$specialty' })
  + availableDoctors;
  + Imperative Apex:
  + createAppointment({ data: this.formData })
  + .then(result => { /\* navigate to record \*/ })
  + .catch(error => { /\* show toast \*/ });
* **Navigation Service**
  + Use NavigationMixin.Navigate to open appointment records after creation.
* **Events in LWC**
  + Publish appointment-created events for other components to refresh.

**Phase 7: Integration & External Access**

**Goal:** Integrate pharmacy, insurance, external billing, and patient portal securely.

* **Named Credentials**
  + Configure for Pharmacy API with JWT or OAuth.
* **External Services & Callouts**
  + REST callout to fetch medicine information (dosage, interactions).
  + SOAP if legacy system requires.
* **Platform Events**
  + Publish PrescriptionIssued\_\_e to notify pharmacy system in near real-time.
* **Change Data Capture (CDC)**
  + Use to push Appointment and Bill changes to external billing system.
* **Salesforce Connect**
  + Use to surface external billing DB records as External Objects.
* **API Limits & Error Handling**
  + Implement exponential backoff and bulkify callouts if needed.
* **OAuth & Authentication**
  + Patient portal: OAuth-connected community or Experience Cloud site for patient login/registration.
* **Remote Site Settings**
  + For dev org: add endpoints if not using Named Credentials (prefer Named Credentials).

**Phase 8: Data Management & Deployment**

**Goal:** Seed test data, handle migrations, ensure data quality.

* **Data Import Wizard**
  + Use for Patients & Doctors small volume uploads with templates.
* **Data Loader**
  + Bulk import Appointments, Prescriptions, Bills in CSV. Pay attention to date/time formats and Sandbox vs Production IDs.
* **Duplicate Rules**
  + Create matching rules for Patients by phone/email to prevent duplicates. Add merge UI/process.
* **Data Export & Backup**
  + Weekly export using Data Export or scheduled Data Loader jobs.
* **Change Sets / VS Code & SFDX / ANT**
  + For CI/CD: store metadata in Git; use sfdx force:source:deploy or use CI tool to run deployments.
  + Example SFDX commands:
  + sfdx force:source:convert -d deploy
  + sfdx force:mdapi:deploy -d deploy -u PROD -w 10
* **Unmanaged vs Managed Packages**
  + Use Unmanaged Package for portfolio sharing (code visible); Managed Packages if you package for distribution.

**Phase 9: Reporting, Dashboards & Security Review**

**Goal:** Provide actionable insights and secure PHI.

* **Reports**
  + Doctor Utilization (Appointments per day/week) — Summary report grouped by Doctor.
  + Patient Visits per Month — Matrix or Summary.
  + Revenue by Treatment Type — Chart with filters by date and doctor.
* **Report Types**
  + Custom report types: Appointment with Prescription & Bill.
* **Dashboards**
  + Manager Dashboard: KPI tiles (Today’s revenue, total patients), charts (Revenue by Type), list components (Top 10 doctors by visits).
  + Doctor Dashboard: Upcoming appointments list (filter: owner or Doctor\_\_c = current user).
* **Dynamic Dashboards**
  + Use for role-based viewing (Doctor sees only own metrics).
* **Sharing Settings**
  + Confirm OWD and sharing rules. Use manual sharing or Apex-managed sharing for exceptional cases.
* **Field-Level Security**
  + Hide sensitive fields (Medical\_History\_\_c, Insurance details) from Receptionist profile.
* **Session Settings**
  + Session timeout 30 minutes.
* **Login IP Ranges**
  + Restrict receptionists to hospital network IP ranges.
* **Audit Trail**
  + Enable field history tracking on Appointment, Patient, Bill. Keep logs for compliance.
* **Security Review Checklist**
  + CRUD/FLS validation in Apex and LWC, remote callout validation, avoid hardcoded credentials, ensure proper CSP in Experience Cloud.

**Phase 10: Final Presentation & Demo Day**

**Goal:** Demonstrate end-to-end value and hand off documentation.

* **Pitch Presentation (slides)**
  + Slide structure: Problem → Impact (metrics) → Solution architecture → Demo screenshots/video → Business benefits → Next steps.
* **Demo Walkthrough (scripted flow)**
  + Login as Receptionist → Book appointment (show Screen Flow) → show prevention of overlap (attempt conflicting booking).
  + Book a Surgery > ₹50,000 → demonstrate approval queue, manager approval UI, approval email.
  + Login as Doctor → view schedule and open Prescription related list → issue prescription → platform event notifies pharmacy (show log).
  + Billing flow: generate Bill, mark Payment, show Revenue dashboard.
  + Show Manager dashboard with KPIs & daily summary email.
* **Handoff Documentation**
  + System Design Doc: ERD, Apex classes, Flows, LWC components, integration endpoints.
  + User Guide: step-by-step for Receptionist, Doctor, Manager, Patient portal.
  + Admin Guide: Deployment steps, code rollout, scheduled jobs, backups.
  + Troubleshooting guide & runbook (how to re-run failed jobs, where logs are).
* **Portfolio / LinkedIn**
  + Add feature list, screenshots, short clip (1–2 min) demo, and link to GitHub (unmanaged package or metadata repository). Include key metrics achieved (e.g., reduced booking conflicts by X% in tests).