# MediCare - Patient Appointment & Treatment Assistant

---

**Project Overview**

The MediCare Patient Appointment & Treatment Assistant is a Salesforce CRM solution designed to streamline healthcare appointment management. This application enables healthcare facilities to manage patient records, schedule appointments with doctors, and automatically generate treatment plans upon appointment completion. The system provides an interactive dashboard for creating appointments, viewing upcoming appointments, and updating appointment statuses. By automating treatment plan creation, MediCare reduces administrative overhead and ensures timely follow-up care for patients.

**Key Features:**

- Centralized patient information management
- Interactive appointment scheduling dashboard
- Automated treatment plan generation
- Multi-specialization appointment tracking
- Real-time status updates

---

**Objectives**

The primary objectives of the MediCare CRM are:

1. **Improve Patient Care:** Provide instant access to patient medical history and appointment details for better care decisions.
2. **Automate Workflows:** Automatically generate treatment plans when appointments are completed, eliminating manual tasks.
3. **Enhance Scheduling:** Enable real-time appointment management across multiple specializations.
4. **Ensure Continuity:** Guarantee every completed appointment has a documented treatment plan for follow-up care.
5. **Increase Efficiency:** Streamline workflows and reduce data entry errors through automation.

---

## Phase 1: Problem Understanding & Industry Analysis

**Requirement Gathering**

Healthcare facilities struggle with paper-based appointment systems leading to missed appointments and lost patient records. MediCare addresses these challenges by providing:

- Quick patient lookup and appointment scheduling
- Digital storage of patient medical history
- Automatic treatment plan creation post-appointment
- Real-time appointment status tracking
- Centralized data for reporting and analytics

**Key Requirements:**

- Patient demographic and medical information storage
- Appointment scheduling with doctor and specialization
- Appointment status workflow (Scheduled, Completed, Cancelled, No Show)
- Automated treatment plan generation
- Interactive dashboard for appointment management

---

**Stakeholder Analysis**

**Primary Stakeholders:**

1. **Front Desk Staff:** Create and manage appointments, access patient contact information
2. **Doctors:** View patient symptoms/history, add diagnosis and prescriptions
3. **System Administrators:** Configure automation, manage users, generate reports
4. **Patients:** Benefit from organized scheduling and timely follow-up care

---

**Business Process Mapping**

**Before MediCare:**

- Manual paper-based scheduling → Double bookings and conflicts
- Lost or illegible patient records
- Missed treatment follow-ups

- No systematic tracking or reporting

**After MediCare:**

- Digital patient records with complete history
- Real-time appointment scheduling via dashboard
- Automated treatment plan creation via Flow
- Complete audit trail and reporting capabilities

**Benefits:** Eliminated double bookings, complete digital records, automated follow-ups, real-time visibility

---

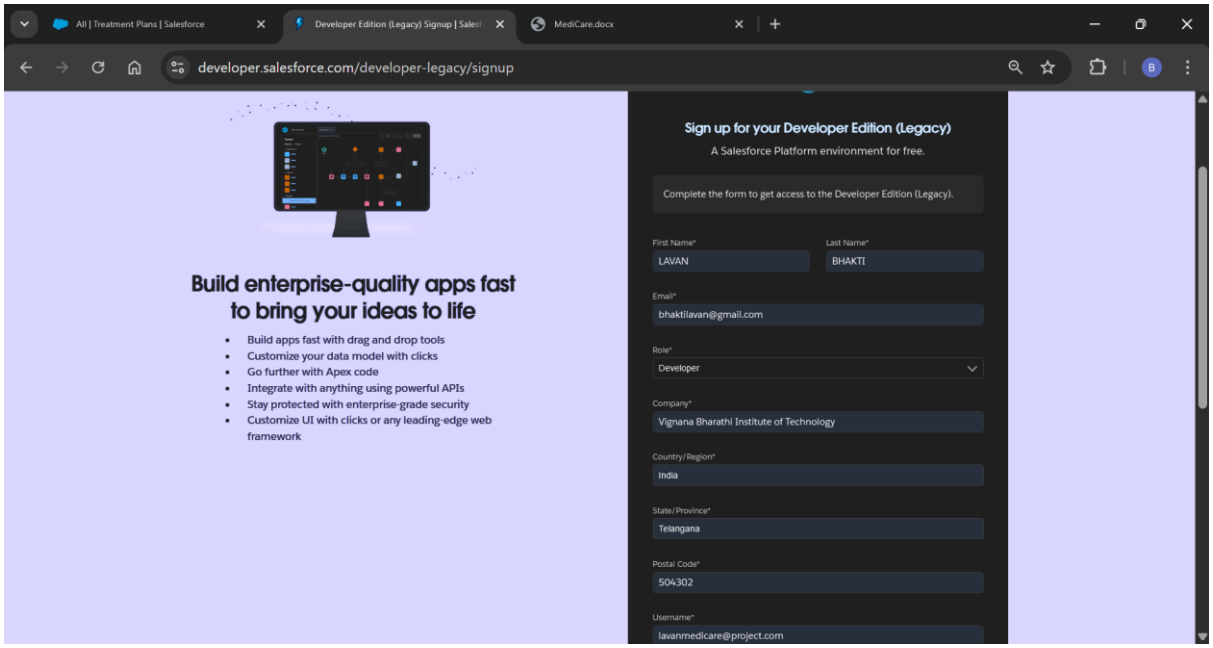
## Phase 2: Org Setup & Configuration

### Salesforce Edition

**Edition Used:** Developer Edition

**Justification:**

- Free access to complete Salesforce platform
- Supports custom objects, Apex, LWC, and Flows
- Suitable for development and demonstration




---

### Company Profile Setup

**Configuration:**

- Organization Name: MediCare Clinic
- Default Language: English
- Default Locale: English (United States)
- Default Time Zone: India Standard Time (IST)
- Currency: INR

---

**User Setup**

**User Configuration:**

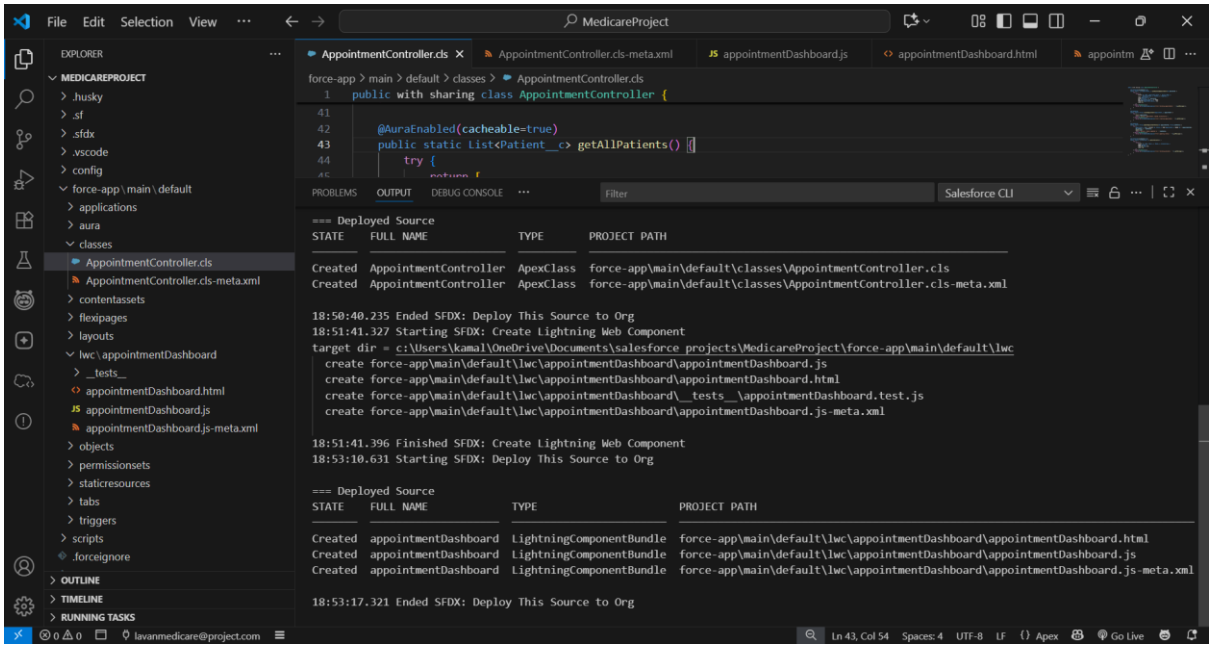
- Name: System Administrator
- License: Salesforce
- Profile: System Administrator
- Purpose: Full org access for development and configuration

---

**Deployment with VS Code**

**Deployment Process:**

1. Installed Salesforce CLI and Salesforce Extensions for VS Code
2. Created SFDX project: MediCareProject
3. Authorized Developer org using SFDX: Authorize an Org
4. Developed Apex class: AppointmentController.cls
5. Developed LWC: appointmentDashboard
6. Deployed using SFDX: Deploy Source to Org
7. Verified components in Salesforce Setup




---

# Phase 3: Data Modeling & Relationships

## Custom Objects

### 1. Patient Object (Patient__c)

**Purpose:** Stores patient demographic and medical information

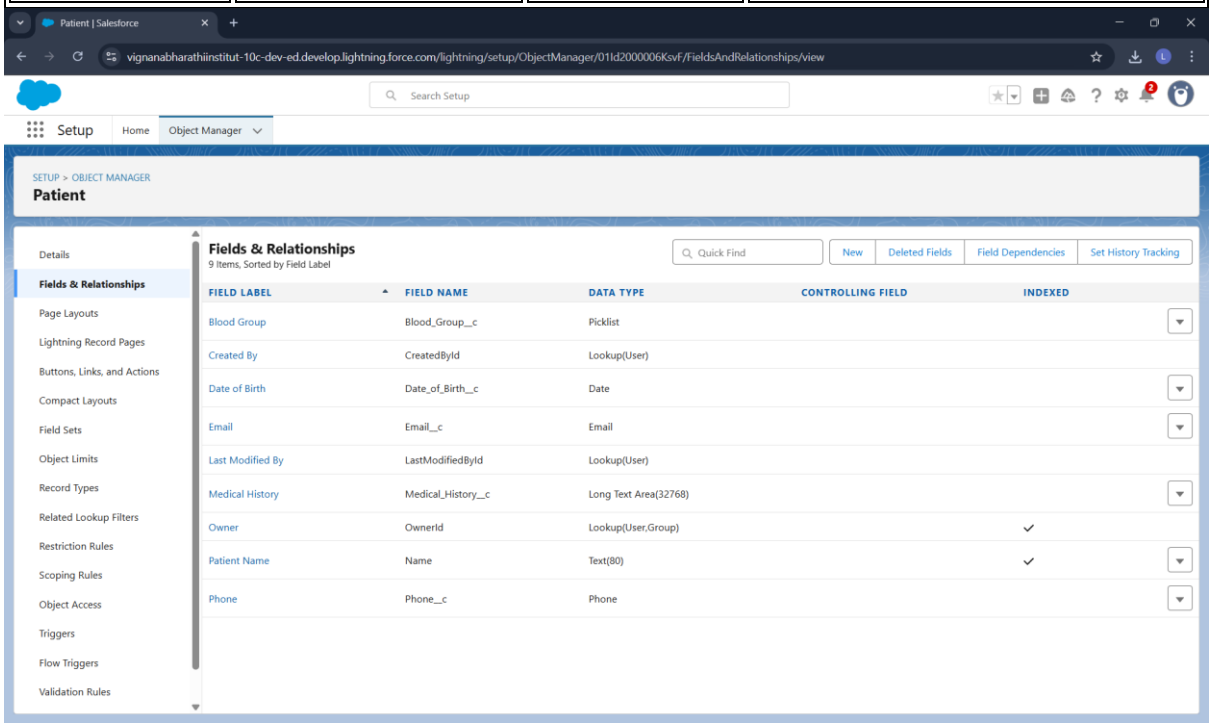
**Configuration:**

- Label: Patient
- Plural Label: Patients
- Record Name: Patient Name (Text)
- Allow Reports: Yes
- Track Field History: Yes

**Fields:**

| Field Label | API Name | Field Type | Values/Length |
|---|---|---|---|
| Patient Name | Name | Text | 80 |
| Email | Email__c | Email | - |
| Phone | Phone__c | Phone | - |
| Date of Birth | Date_of_Birth__c | Date | - |

| Field Label | API Name | Field Type | Values/Length |
|---|---|---|---|
| Blood Group | Blood_Group__c | Picklist | A+, A-, B+, B-, AB+, AB-, O+, O- |
| Medical History | Medical_History__c | Long Text Area | 5 lines |




---

### 2. Appointment Object (Appointment__c)

**Purpose:** Manages appointment scheduling and tracking

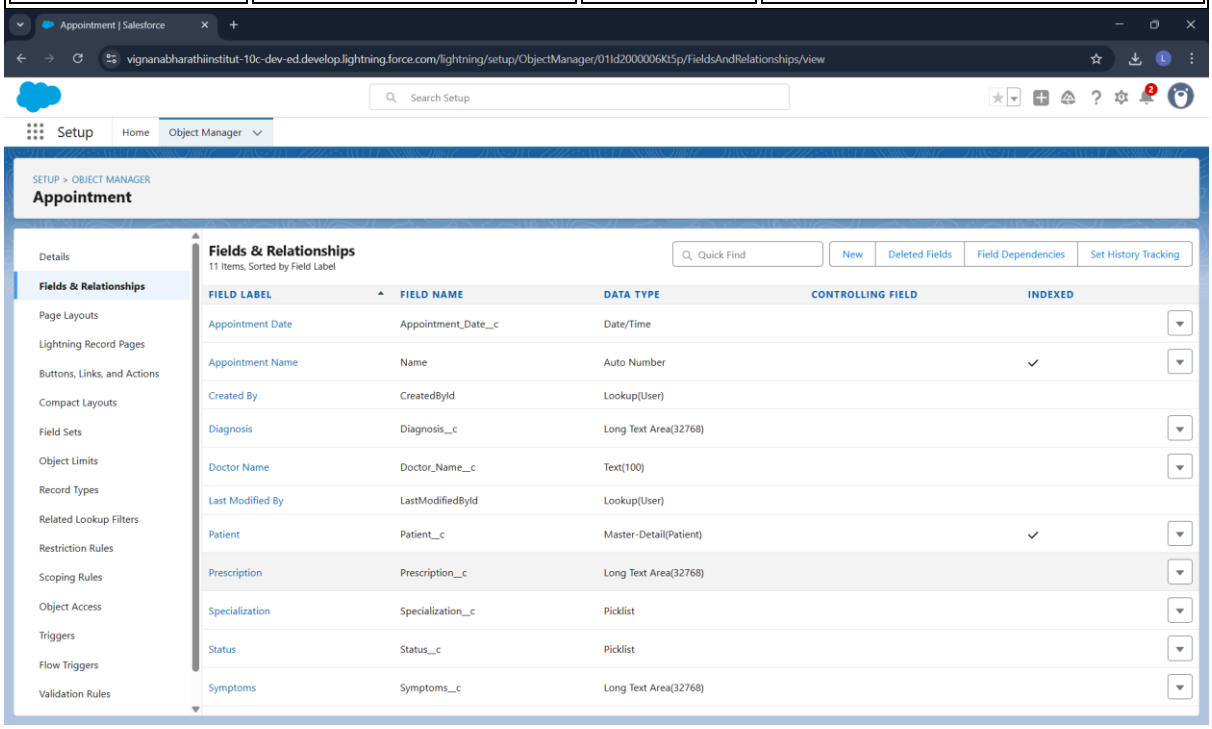
**Configuration:**

- Label: Appointment
- Plural Label: Appointments
- Record Name: Appointment Number (Auto Number: AP-{0000})
- Allow Reports: Yes
- Track Field History: Yes

**Fields:**

| Field Label | API Name | Field Type | Values |
|---|---|---|---|
| Appointment Number | Name | Auto Number | AP-{0000} |

| Field Label | API Name | Field Type | Values |
|---|---|---|---|
| Patient | Patient__c | Master-Detail | To Patient__c |
| Appointment Date | Appointment_Date__c | Date/Time | - |
| Doctor Name | Doctor_Name__c | Text | 100 chars |
| Specialization | Specialization__c | Picklist | General Medicine, Cardiology, Orthopedics, Pediatrics, Dermatology |
| Status | Status__c | Picklist | Scheduled, Completed, Cancelled, No Show |
| Symptoms | Symptoms__c | Long Text Area | 3 lines |
| Diagnosis | Diagnosis__c | Long Text Area | 3 lines |
| Prescription | Prescription__c | Long Text Area | 5 lines |



### 3. Treatment Plan Object (Treatment_Plan__c)

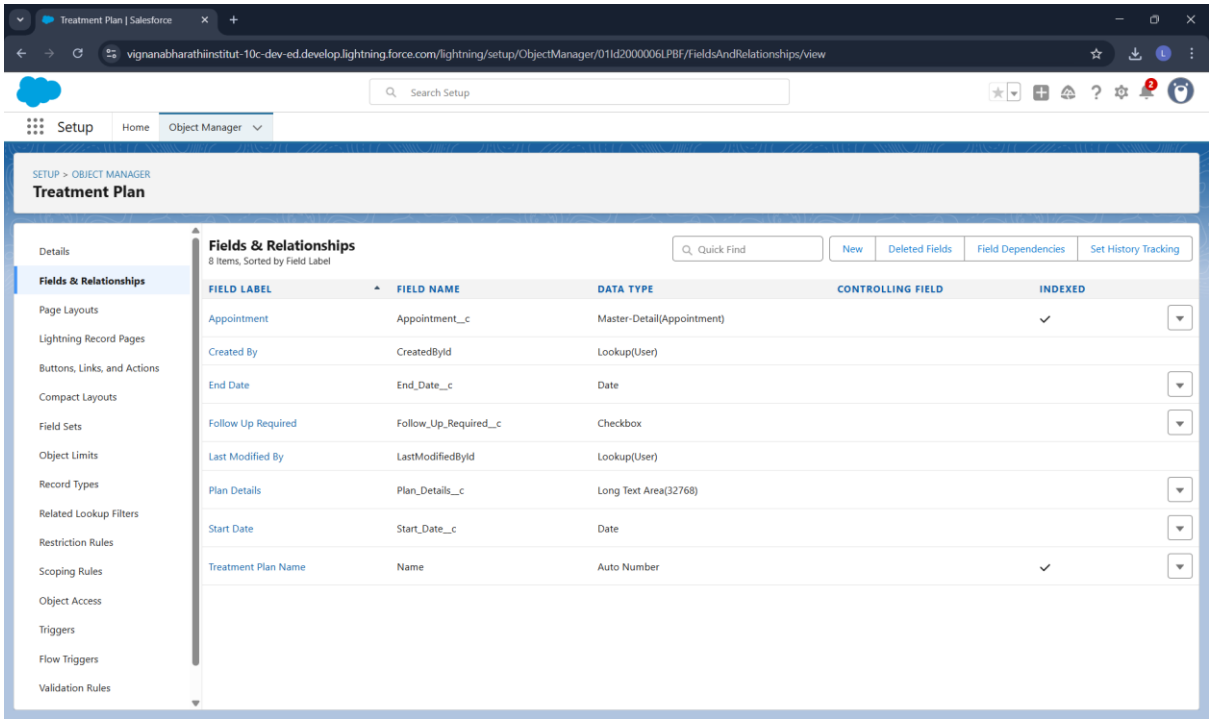
**Purpose:** Documents post-appointment treatment plans

**Configuration:**

- Label: Treatment Plan
- Plural Label: Treatment Plans
- Record Name: Treatment Plan Number (Auto Number: TP-{0000})
- Allow Reports: Yes

**Fields:**

| Field Label | API Name | Field Type | Values |
|---|---|---|---|
| Treatment Plan Number | Name | Auto Number | TP-{0000} |
| Appointment | Appointment__c | Master-Detail | To Appointment__c |
| Plan Details | Plan_Details__c | Long Text Area | 5 lines |
| Start Date | Start_Date__c | Date | - |
| End Date | End_Date__c | Date | - |
| Follow Up Required | Follow_Up_Required__c | Checkbox | - |



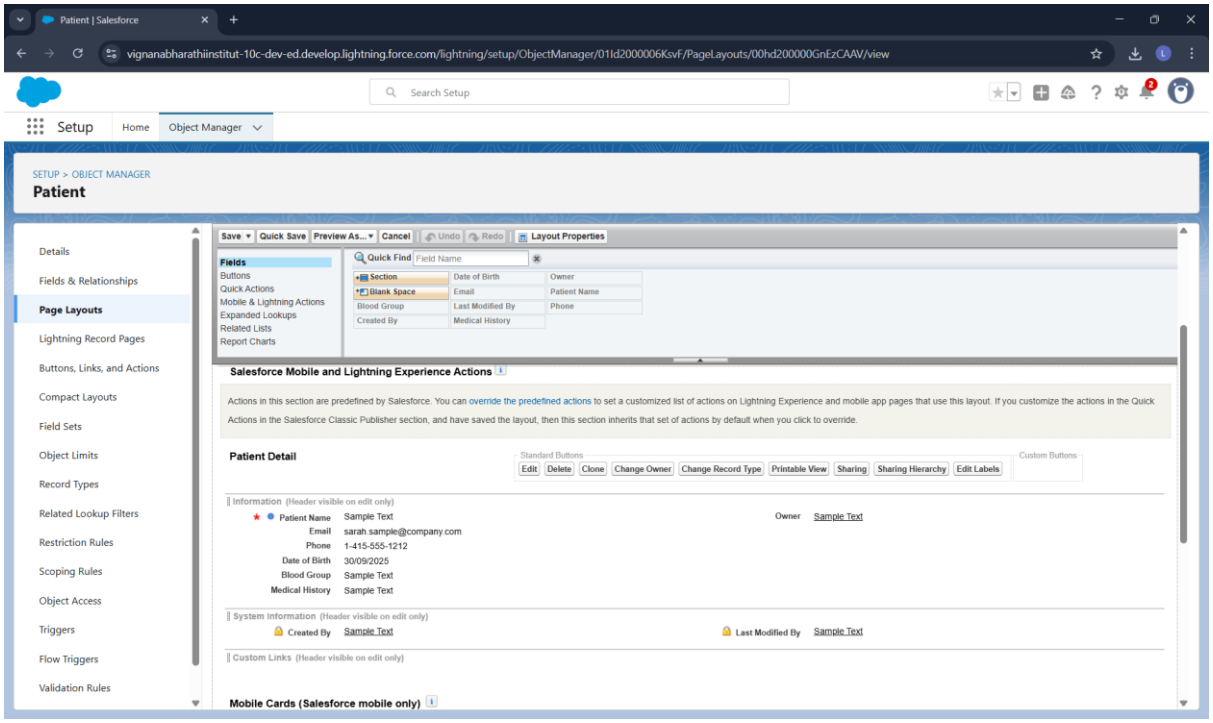

---

## Page Layouts

## Patient Page Layout

## Sections:

- Patient Information: Name, Date of Birth, Blood Group
- Contact Information: Email, Phone
- Medical Details: Medical History
- Related List: Appointments

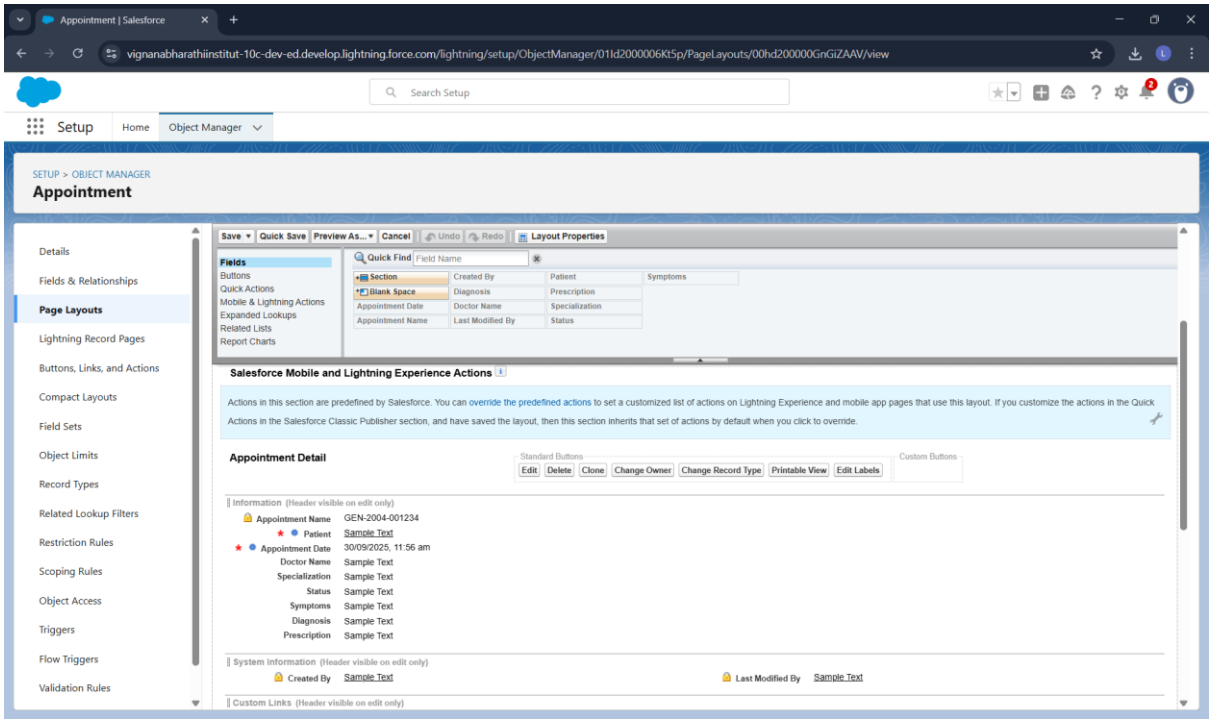



---

## Appointment Page Layout

## Sections:

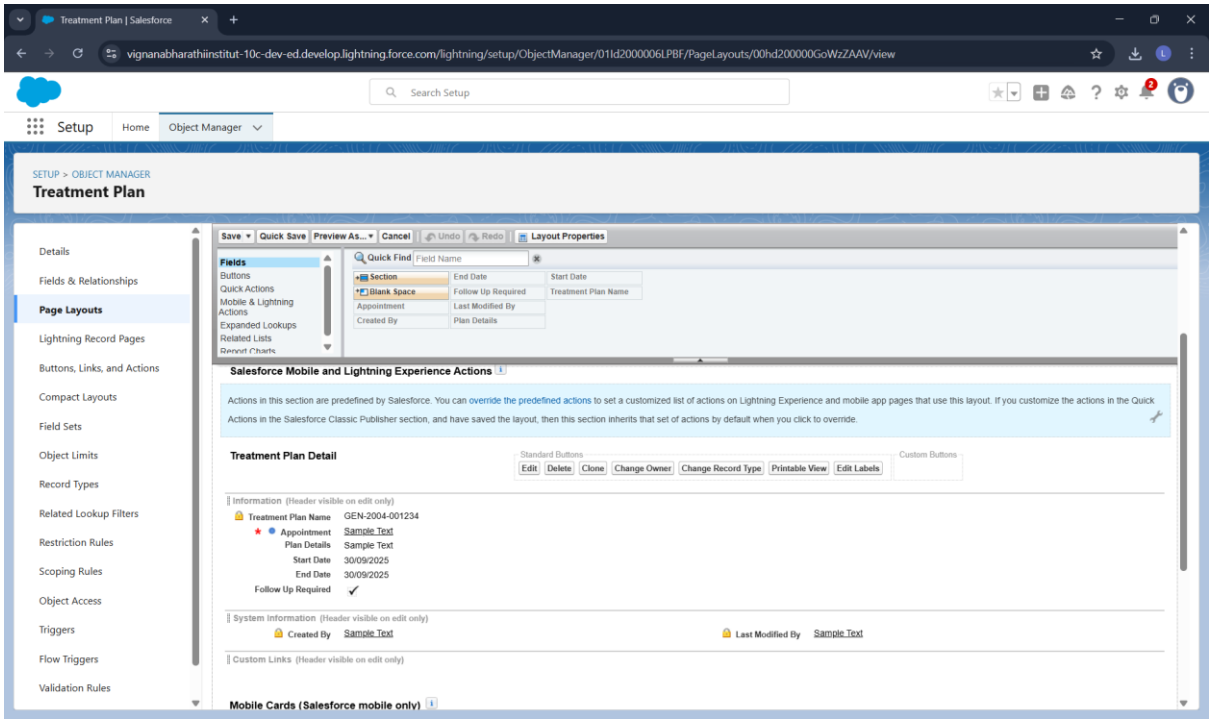
- Appointment Details: Number, Patient, Date, Status
- Doctor Information: Doctor Name, Specialization
- Clinical Information: Symptoms, Diagnosis, Prescription
- Related List: Treatment Plans




## Treatment Plan Page Layout

## Sections:

- Treatment Plan Details: Number, Appointment, Plan Details
- Timeline: Start Date, End Date, Follow Up Required



**Schema Builder**

**Data Model:**

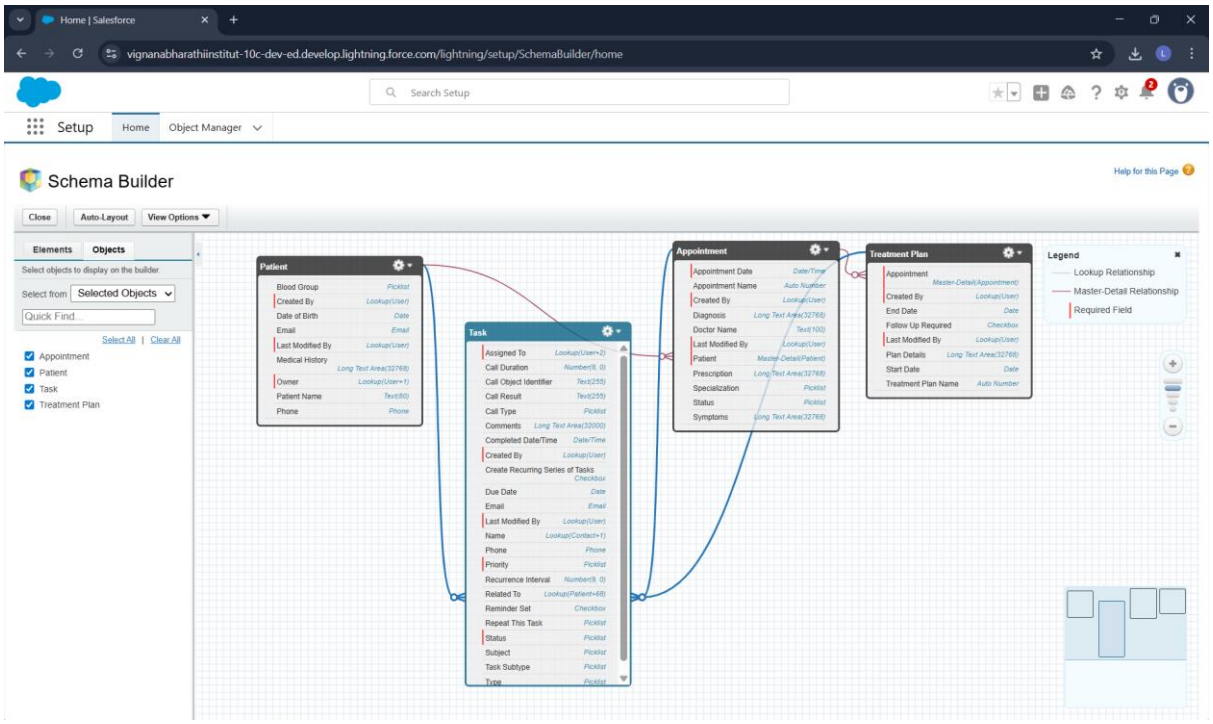
Patient (Parent)

↓ Master-Detail

Appointment (Parent/Child)

↓ Master-Detail

Treatment Plan (Child)




---

**Master-Detail Relationships**

**Patient → Appointment**

- Parent: Patient__c
- Child: Appointment__c
- Field: Patient__c (Master-Detail)
- Cascade Delete: Enabled

**Justification:** Appointments cannot exist without patients. Maintains data integrity.

---

**Appointment → Treatment Plan**

- Parent: Appointment__c
- Child: Treatment_Plan__c
- Field: Appointment__c (Master-Detail)
- Cascade Delete: Enabled

**Justification:** Treatment plans are linked to specific appointments. Ensures data consistency.

---

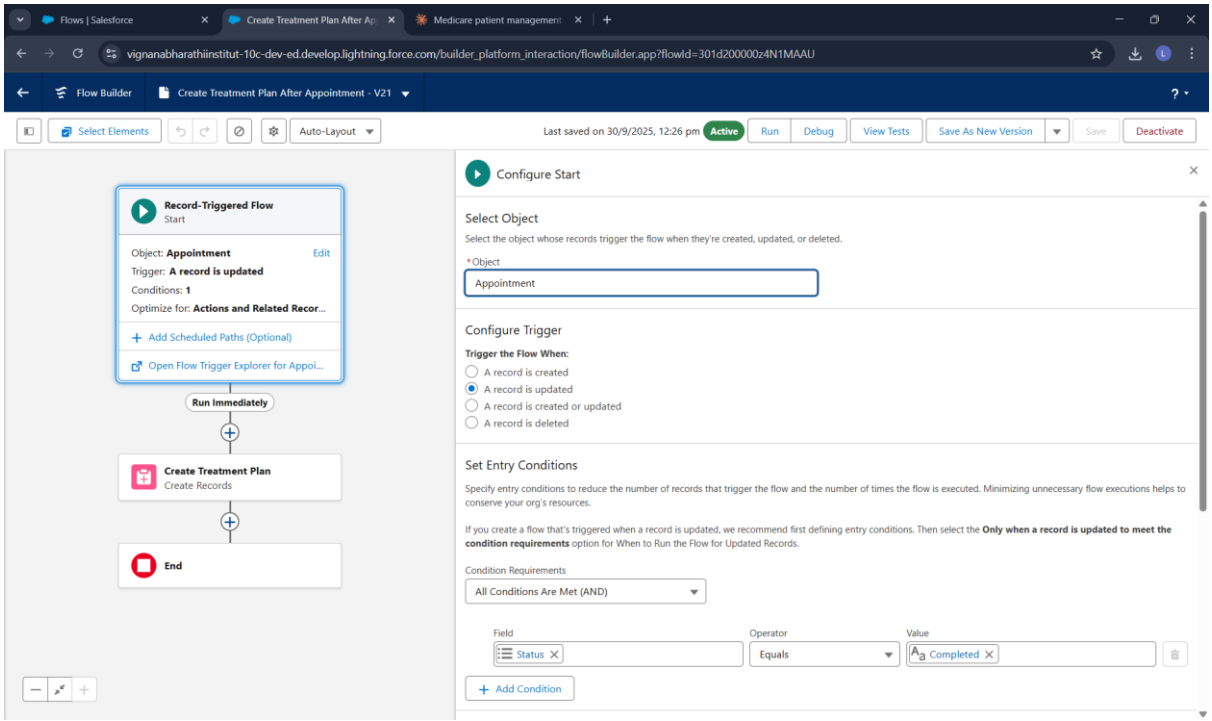
## Phase 4: Process Automation (Admin)

### Record-Triggered Flow

**Flow Name:** Create Treatment Plan After Appointment **Flow Type:** Record-Triggered Flow

### Trigger Configuration:

- Object: Appointment__c
- Trigger: A record is updated
- Entry Condition: Status__c EQUALS Completed
- Optimize For: Actions and Related Records




### Flow Elements:

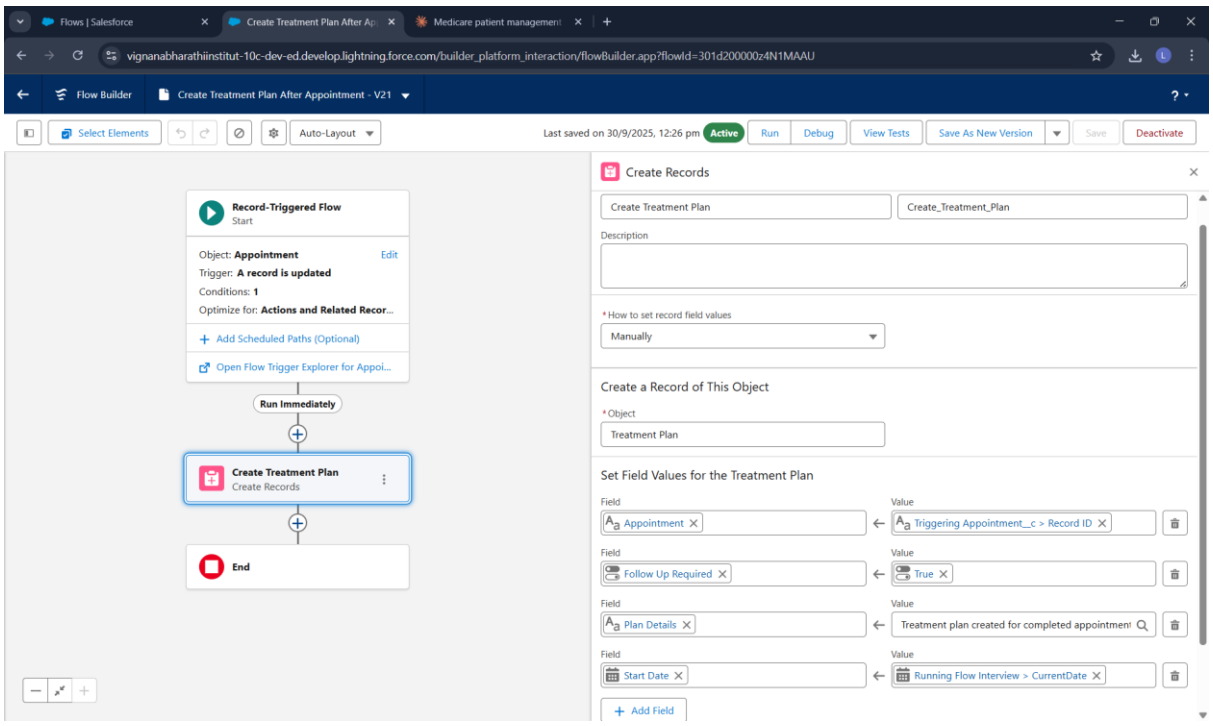
### 1. Start Element:

- Captures updated Appointment record
- Checks if Status = "Completed"

**2. Create Records Element:**

- Label: Create Treatment Plan
- Object: Treatment_Plan__c
- Field Values:
    - Appointment__c = {!$Record.Id}
    - Plan_Details__c = "Treatment plan created for completed appointment"
    - Start_Date__c = {!$Flow.CurrentDate}
    - End_Date__c = {!$Flow.CurrentDate} + 30
    - Follow_Up_Required__c = {!$GlobalConstant.True}

**Business Logic:** When an appointment status changes to "Completed", the flow automatically creates a treatment plan linked to that appointment with a 30-day treatment period and follow-up requirement.




## Phase 5: Apex Programming (Developer)

**Apex Class: AppointmentController**

**Class Name:** AppointmentController **Type:** Public with sharing **Purpose:** Backend controller for appointmentDashboard LWC

**Methods Implemented:**

**1. getUpcomingAppointments**

apex

@AuraEnabled(cacheable=true)

public static List<Appointment__c> getUpcomingAppointments(Id patientId)

- Returns upcoming appointments for selected patient
- Filters by Appointment_Date__c >= TODAY
- Cacheable for performance

---

**2. createAppointment**

apex

@AuraEnabled

public static String createAppointment(Appointment__c appointment)

- Inserts new appointment record
- Returns success message
- Includes error handling

---

**3. updateAppointmentStatus**

apex

@AuraEnabled

public static String updateAppointmentStatus(Id appointmentId, String newStatus)

- Updates appointment status
- Triggers flow when status = "Completed"
- Returns confirmation message

---

**4. getAllPatients**

apex

@AuraEnabled(cacheable=true)

public static List<Patient__c> getAllPatients()

- Returns all patient records for dropdown
- Ordered by Name
- Cacheable for performance

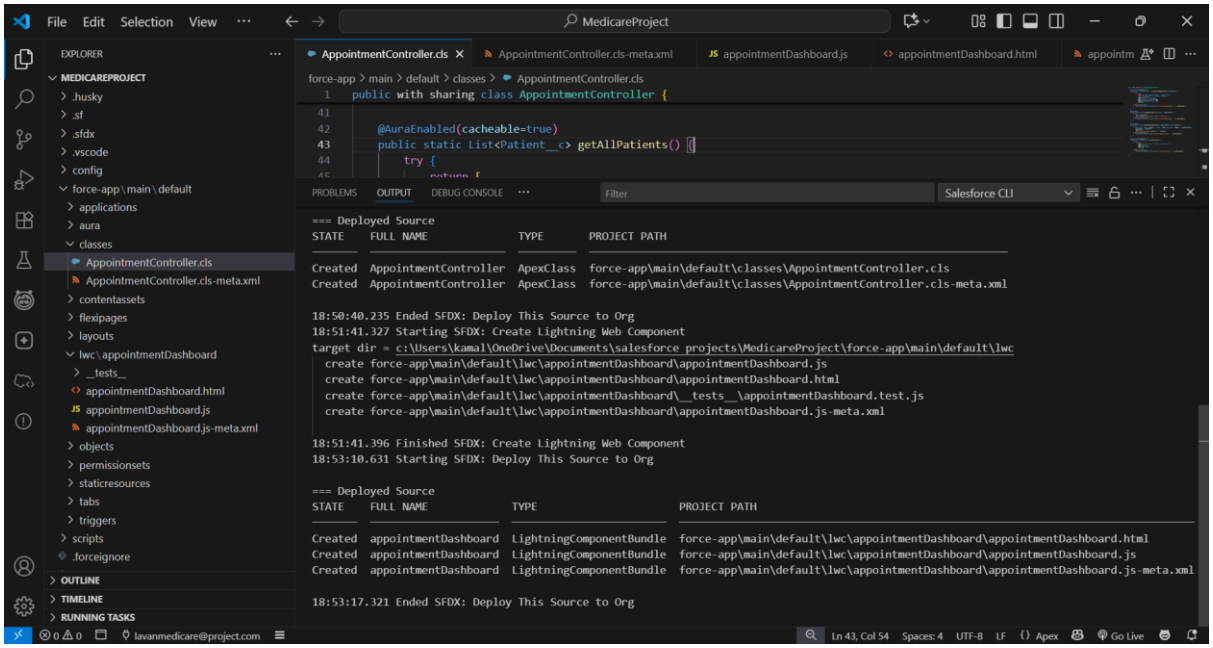
---

**Complete Class:**

apex

```
public with sharing class AppointmentController {

    @AuraEnabled(cacheable=true)
    public static List<Appointment__c> getUpcomingAppointments(Id patientId) {
        try {
            return [SELECT Id, Name, Appointment_Date__c, Doctor_Name__c,
                    Specialization__c, Status__c, Symptoms__c
                    FROM Appointment__c
                    WHERE Patient__c = :patientId AND Appointment_Date__c >= TODAY
                    ORDER BY Appointment_Date__c ASC LIMIT 50];
        } catch (Exception e) {
            throw new AuraHandledException('Error: ' + e.getMessage());
        }
    }

    @AuraEnabled
    public static String createAppointment(Appointment__c appointment) {
        try {
            insert appointment;
            return 'Success: Appointment created successfully!';
        } catch (Exception e) {
```

```
            throw new AuraHandledException('Error: ' + e.getMessage());
        }
    }

    @AuraEnabled
    public static String updateAppointmentStatus(Id appointmentId, String newStatus)
{
        try {
            Appointment__c apt = [SELECT Id, Status__c FROM Appointment__c
                                  WHERE Id = :appointmentId];
            apt.Status__c = newStatus;
            update apt;
            return 'Success: Status updated to ' + newStatus;
        } catch (Exception e) {
            throw new AuraHandledException('Error: ' + e.getMessage());
        }
    }

    @AuraEnabled(cacheable=true)
    public static List<Patient__c> getAllPatients() {
        try {
            return [SELECT Id, Name, Email__c, Phone__c, Blood_Group__c
                    FROM Patient__c ORDER BY Name ASC LIMIT 100];
        } catch (Exception e) {
            throw new AuraHandledException('Error: ' + e.getMessage());
        }
    }
}
```




---

# Phase 6: User Interface Development

### Lightning Web Component: appointmentDashboard

**Component Name:** appointmentDashboard **Purpose:** Interactive dashboard for appointment management

**Files:**

1. appointmentDashboard.js - JavaScript controller
2. appointmentDashboard.html - HTML template
3. appointmentDashboard.js-meta.xml - Metadata

---

### Key Features

### 1. Patient Selection:

- Combobox dropdown with all patients
- Wire service integration with getAllPatients
- Reactive appointment loading on selection

### 2. Create Appointment Form:

- Date/time picker for appointment scheduling
- Doctor name and specialization fields

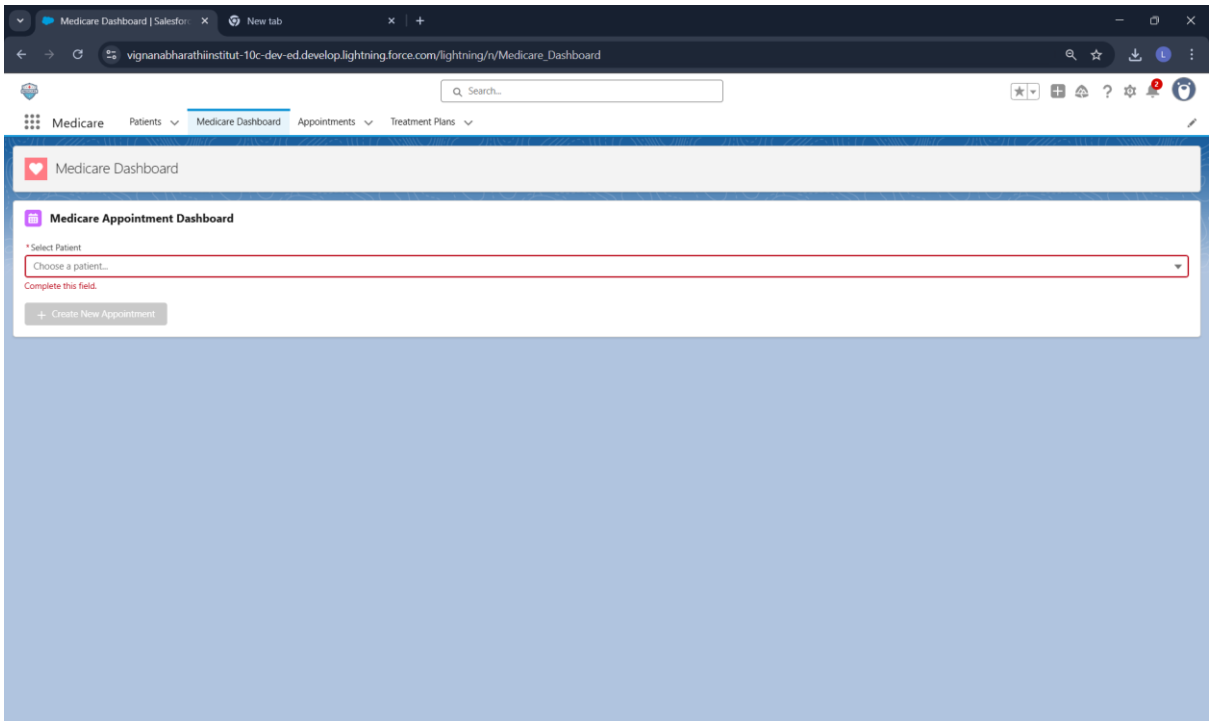
- Symptoms textarea
- Create and Cancel buttons
- Form validation before submission

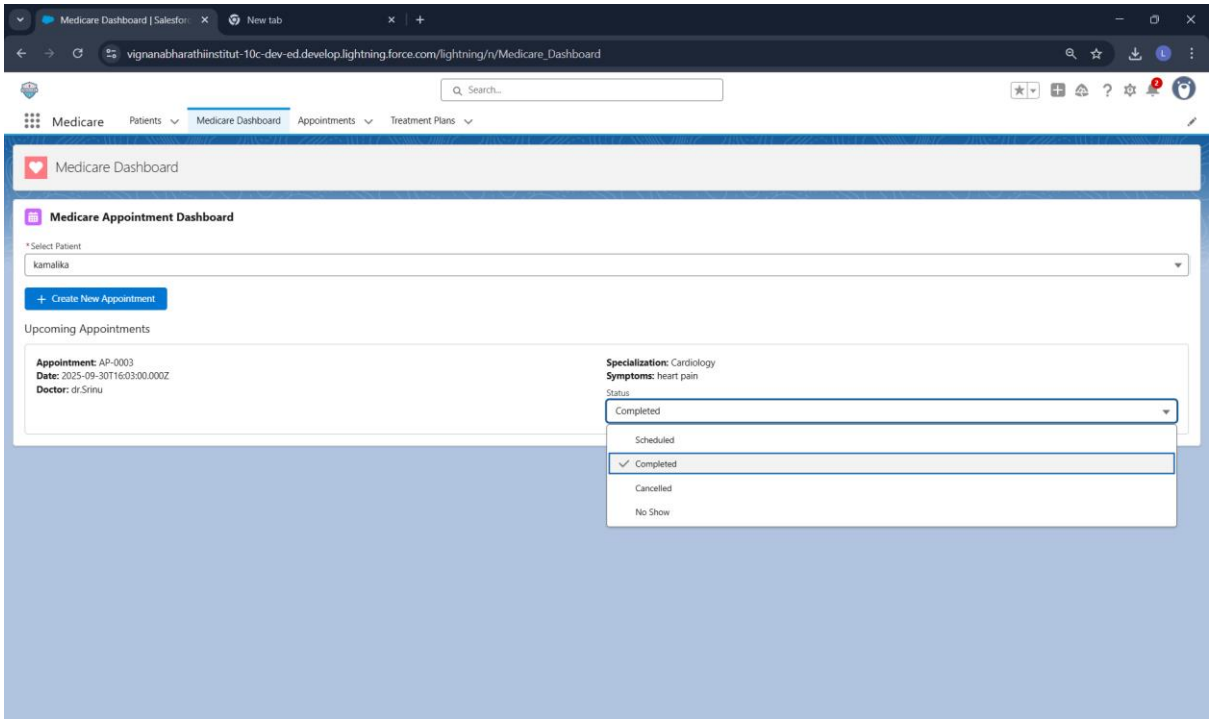
**3. Appointments List:**

- Displays upcoming appointments for selected patient
- Shows appointment details (date, doctor, specialization, symptoms)
- Inline status update combobox
- Responsive grid layout

**4. Status Management:**

- Update appointment status from dashboard
- Triggers flow when changed to "Completed"
- Automatic list refresh after update






---

**Apex Integration**

**Wire Adapters:**

javascript

@wire(getAllPatients)

@wire(getUpcomingAppointments, { patientId: '$selectedPatientId' })

**Imperative Calls:**

javascript

createAppointment({ appointment })

updateAppointmentStatus({ appointmentId, newStatus })

**RefreshApex:** Used to refresh data after DML operations

---

**Component Metadata**

xml

<apiVersion>62.0</apiVersion>

<isExposed>true</isExposed>

<targets>

<target>lightning__AppPage</target>

<target>lightning__HomePage</target>

</targets>

---

**Lightning App Builder**

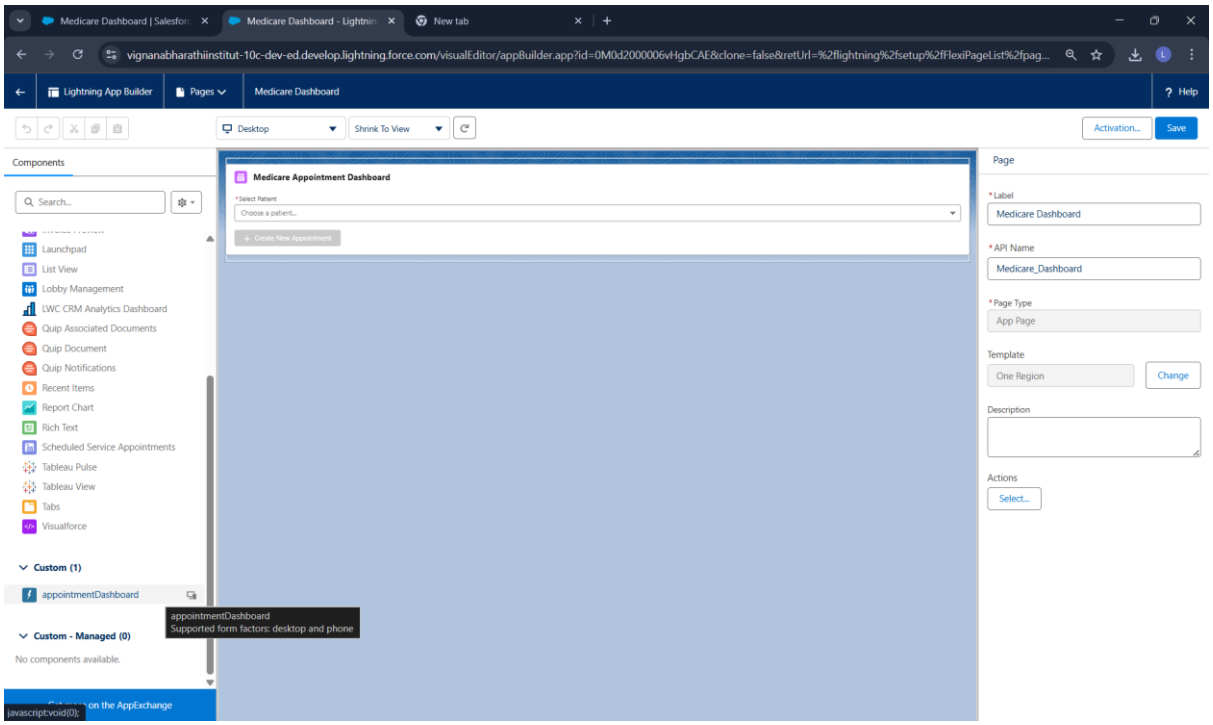
**Page Name:** Medicare Dashboard **Page Type:** App Page **Layout:** One Region

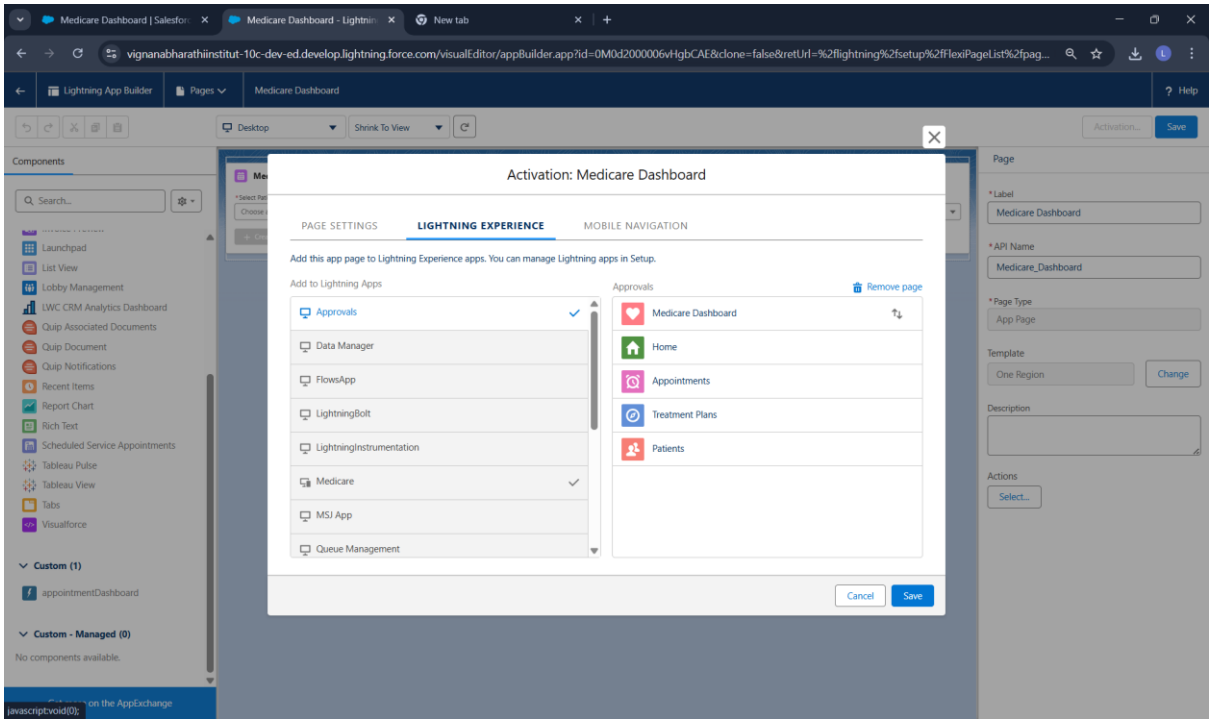
**Component Added:**

- appointmentDashboard (custom LWC)

**Activation:**

- Set as Org Default






---

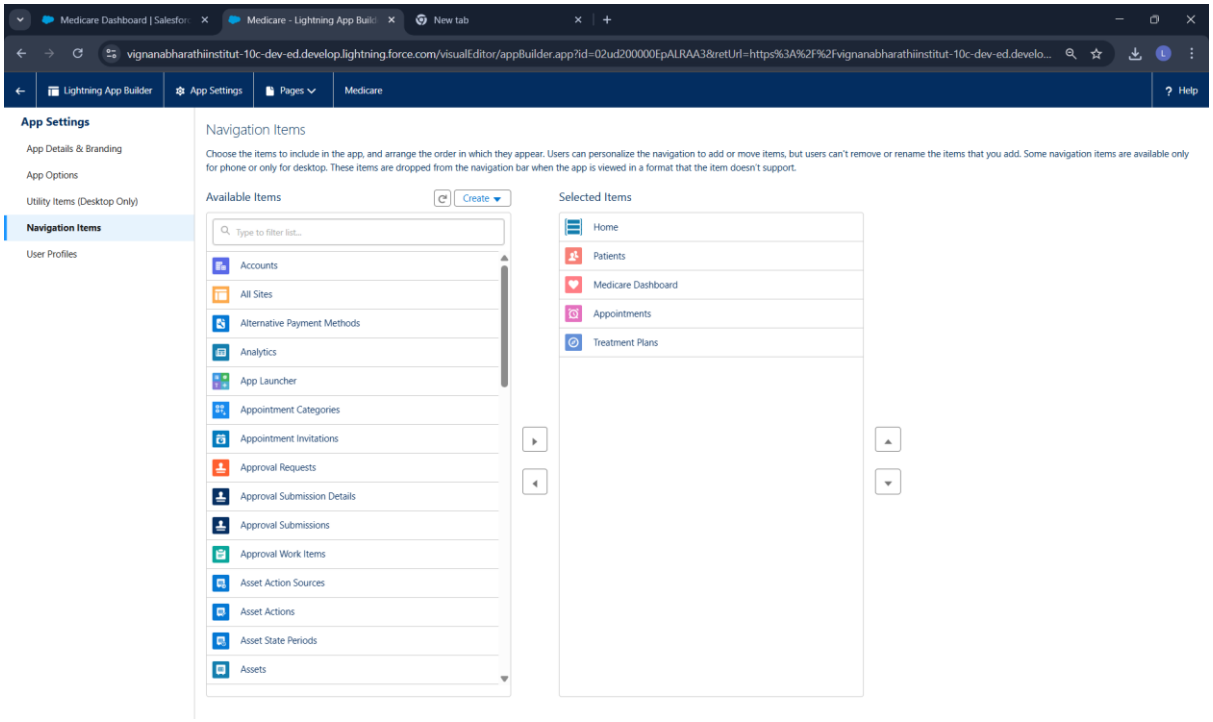
**Custom App: Medicare**

**App Name:** Medicare **Navigation Type:** Standard

**Navigation Items:**

1. Home
2. Patients
3. Appointments
4. Treatment Plans
5. Medicare Dashboard

**Profile Assignment:** System Administrator




---

# Phase 7: Integration & External Access

*Not implemented in current project version*

---

# Phase 8: Data Management & Deployment

### VS Code & SFDX

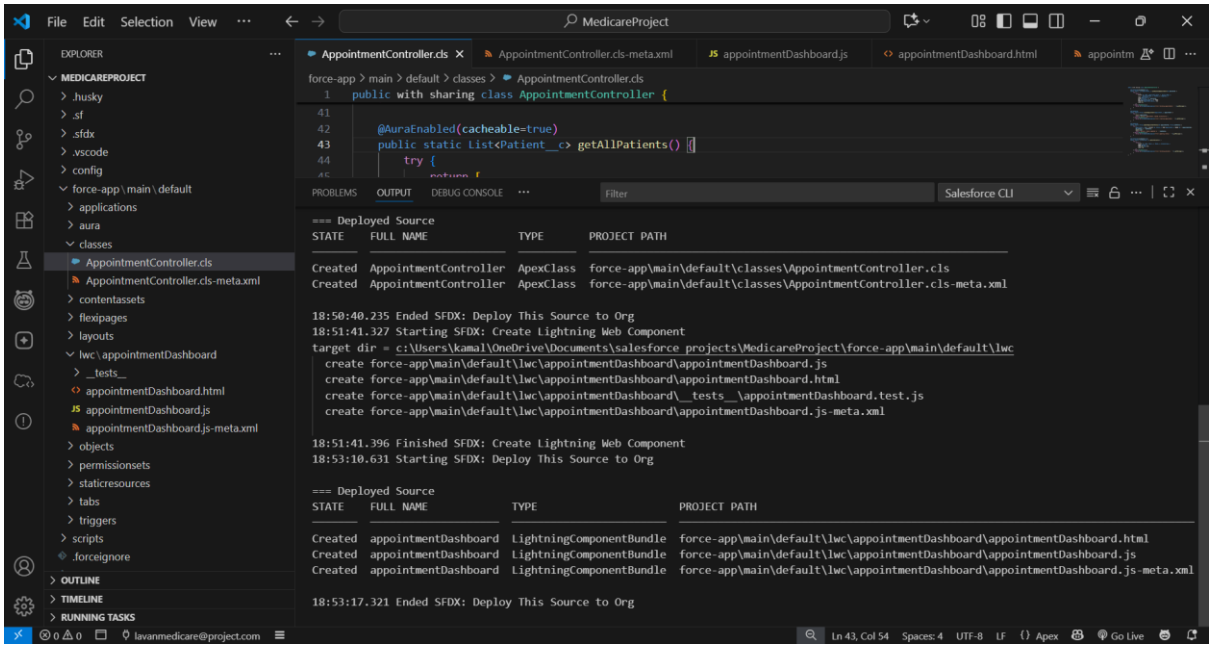
**Deployment Method:** Salesforce CLI with VS Code

**Components Deployed:**

1. Apex Class: AppointmentController.cls
2. LWC: appointmentDashboard

**Deployment Steps:**

1. Created SFDX project in VS Code
2. Authorized Developer org
3. Developed components in VS Code
4. Deployed using SFDX: Deploy Source to Org
5. Verified deployment in Salesforce Setup




---

# Phase 9: Reporting, Dashboards & Security

## Reports

### Appointment Report

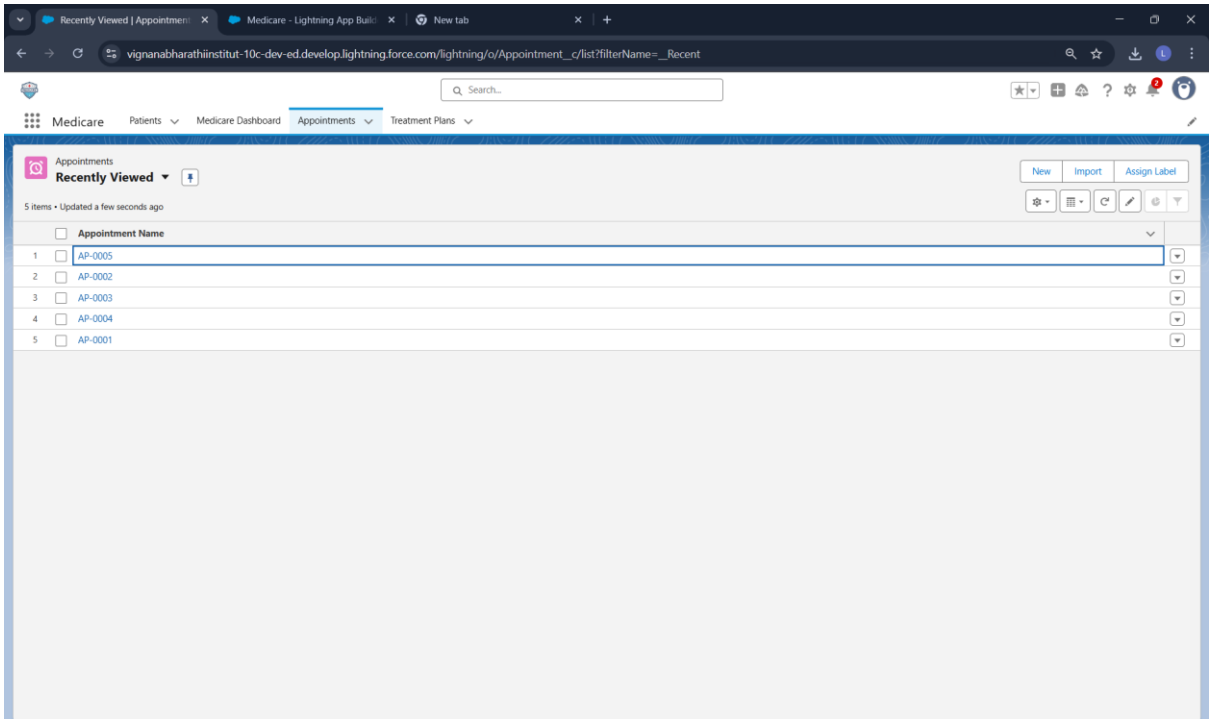
**Report Name:** All Appointments Report **Report Type:** Appointments with Patients **Report Format:** Tabular

**Columns:**

- Patient Name
- Appointment Number
- Appointment Date
- Doctor Name
- Specialization
- Status
- Symptoms

**Filters:**

- Appointment Date >= THIS YEAR



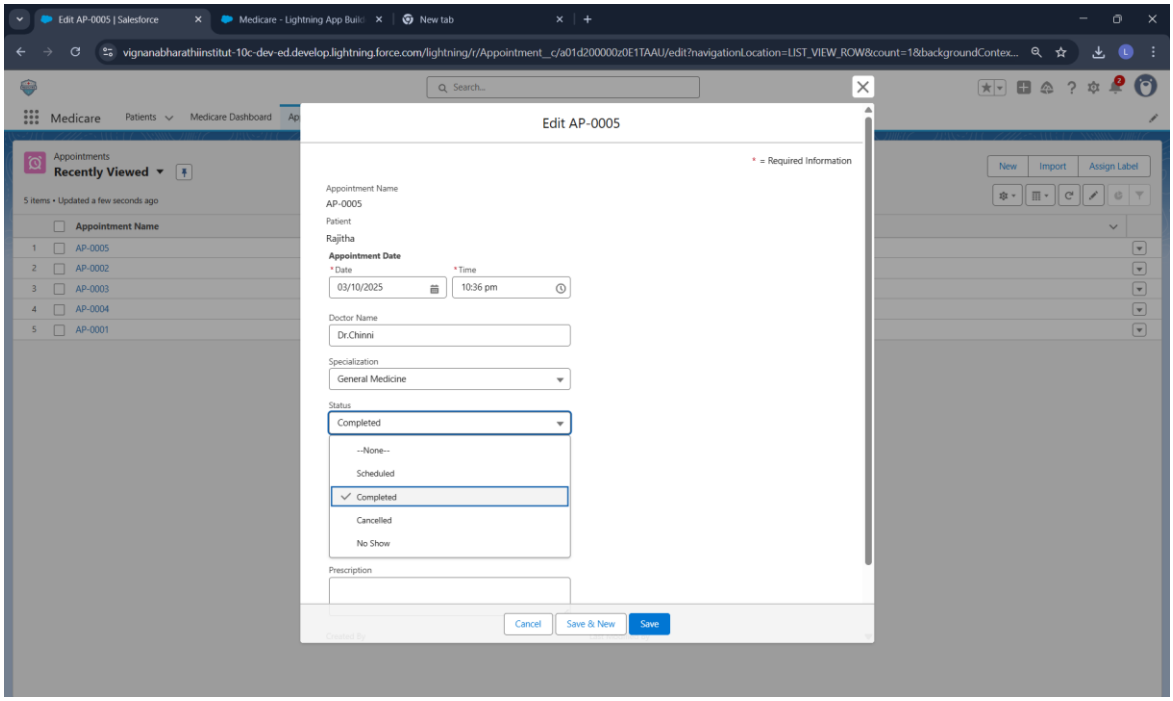

---

## Completed Appointments Report

**Report Name:** Completed Appointments **Report Type:** Appointments **Report Format:** Summary (grouped by Status)

**Filters:**

- Status = Completed




---

**Dashboards**

**Dashboard Name:** MediCare Dashboard **Components:**

1. Appointments by Status (Donut Chart)
2. Appointments by Specialization (Bar Chart)
3. Recent Appointments (Table)

---

**Profiles & Permission Sets**

**Profile Used:** System Administrator

- Full access to all objects
- Create, Read, Edit, Delete permissions
- Access to Apex classes and LWC

---

**Object-Level Security (OWD)**

**Patient Object:**

- Default Access: Private
- Controlled by: Private (Master-Detail controls Appointment access)

**Appointment Object:**

- Controlled by Parent (Patient)

**Treatment Plan Object:**

- Controlled by Parent (Appointment)

---

## Phase 10: Quality Assurance Testing

**Test Case 1: Create Patient Record**

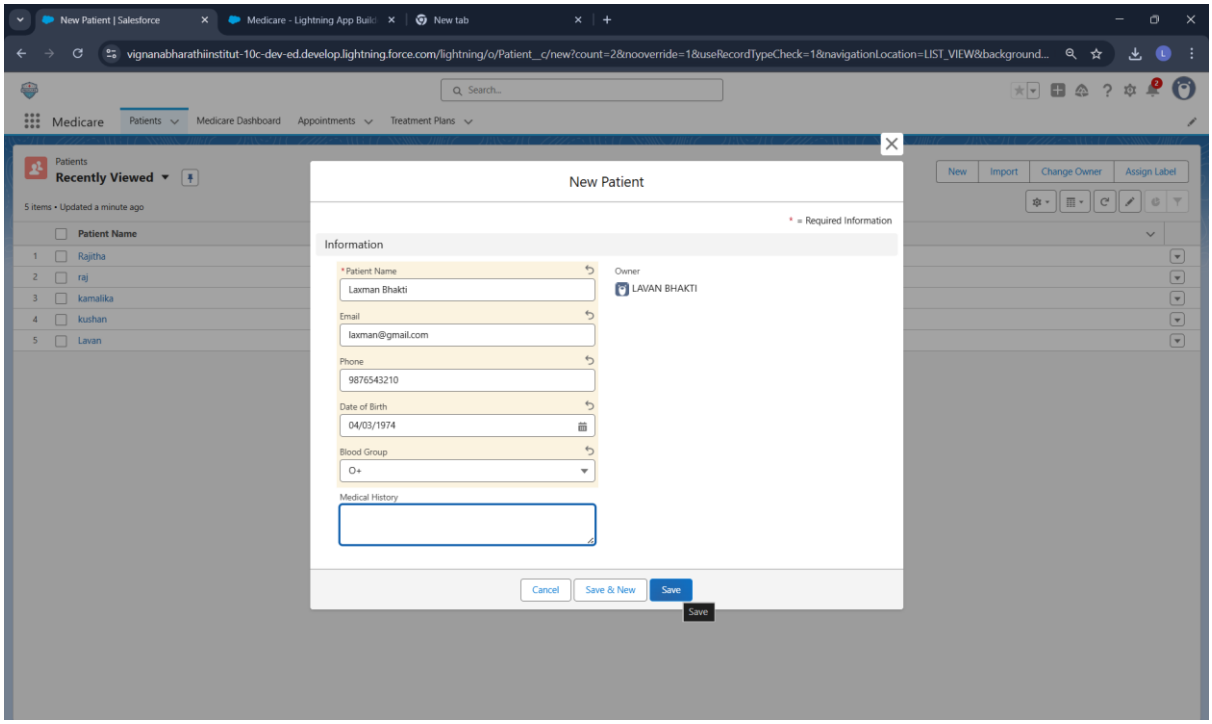
**Use Case:** Create a new patient with complete information

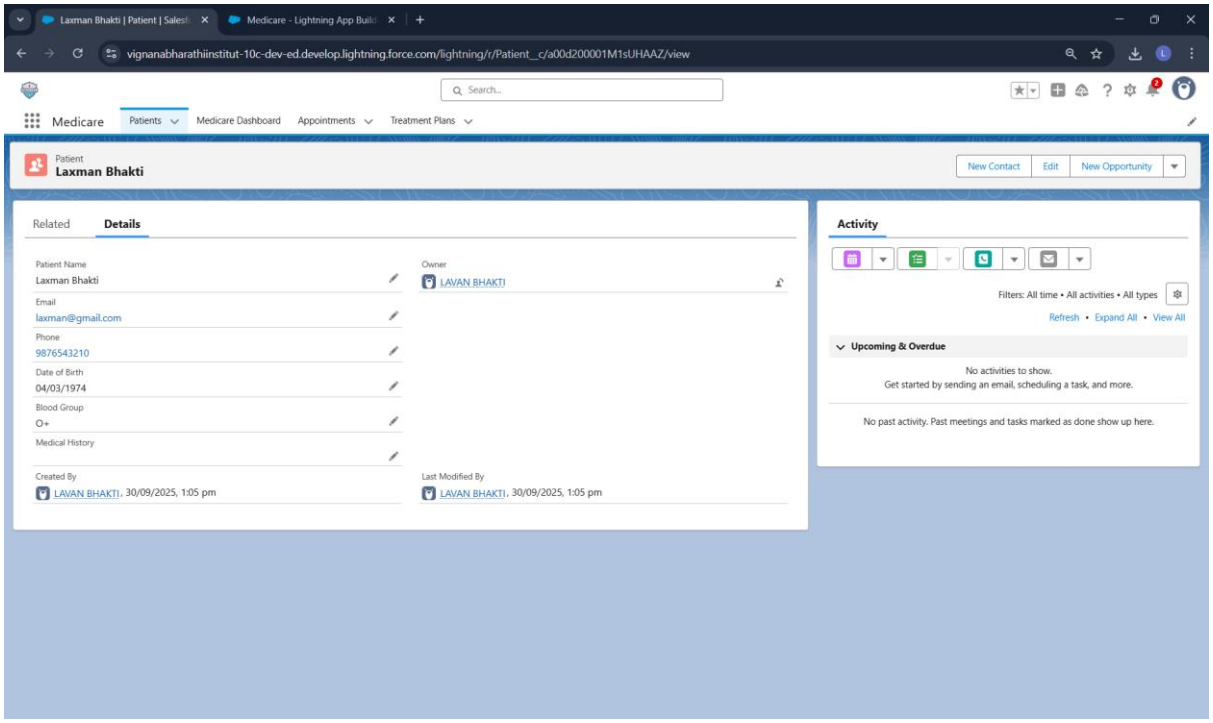
**Test Steps:**

1. Navigate to Patients tab
2. Click New
3. Enter Patient Name: Laxman Bhakti
4. Enter Email: laxman@gmail.com
5. Enter Phone: 9876543210
6. Select Blood Group: O+
7. Enter Medical History: (Optional)
8. Click Save

**Expected Result:** Patient record created successfully with all field values saved

**Actual Result:** Patient record created successfully






---

**Test Case 2: Create Appointment via Dashboard**

**Use Case:** Schedule a new appointment for a patient using LWC dashboard

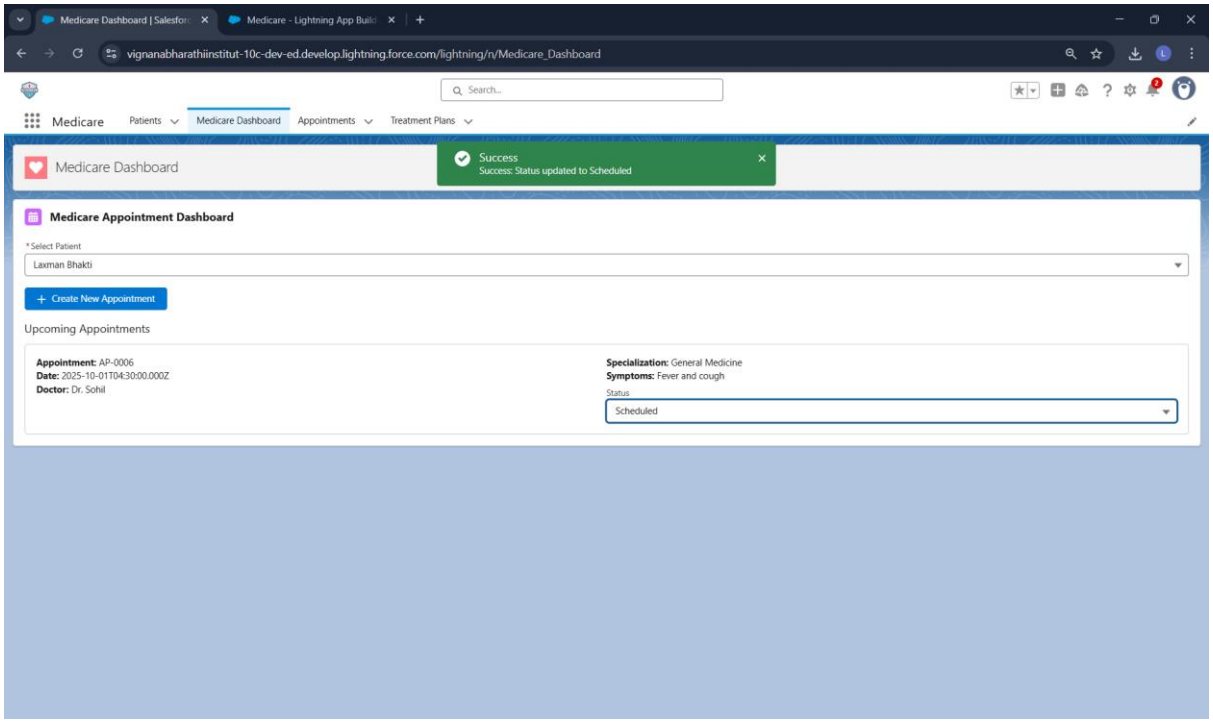
**Test Steps:**

1. Navigate to Medicare Dashboard
2. Select Patient: Laxman Bhakti
3. Click "Create New Appointment"
4. Enter Appointment Date: Tomorrow 10:00 AM
5. Enter Doctor Name: Dr. Sohil
6. Select Specialization: General Medicine
7. Enter Symptoms: Fever and cough
8. Click "Create Appointment"

**Expected Result:**

- Success toast message appears
- New appointment appears in upcoming appointments list
- Appointment Status = Scheduled

**Actual Result:** Appointment created successfully with Status = Scheduled




---

**Test Case 3: Update Appointment Status**

**Use Case:** Change appointment status from dashboard

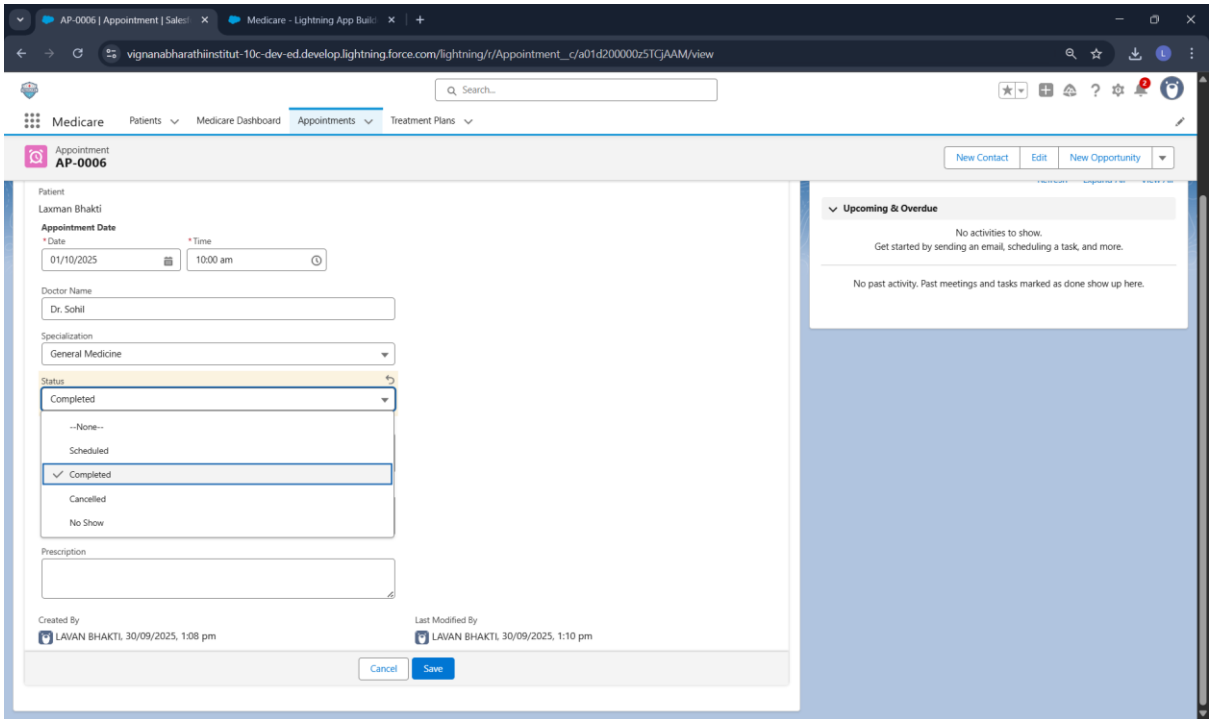
**Test Steps:**

1. Navigate to Medicare Dashboard
2. Select Patient: Laxman Bhakti
3. Locate created appointment
4. Change Status dropdown from "Scheduled" to "Completed"

**Expected Result:**

- Status updates to "Completed"
- Success toast message appears
- List refreshes with updated status

**Actual Result:** Status updated successfully to Completed




---

**Test Case 4: Automatic Treatment Plan Creation (Flow)**

**Use Case:** Verify treatment plan is auto-created when appointment status changes to Completed

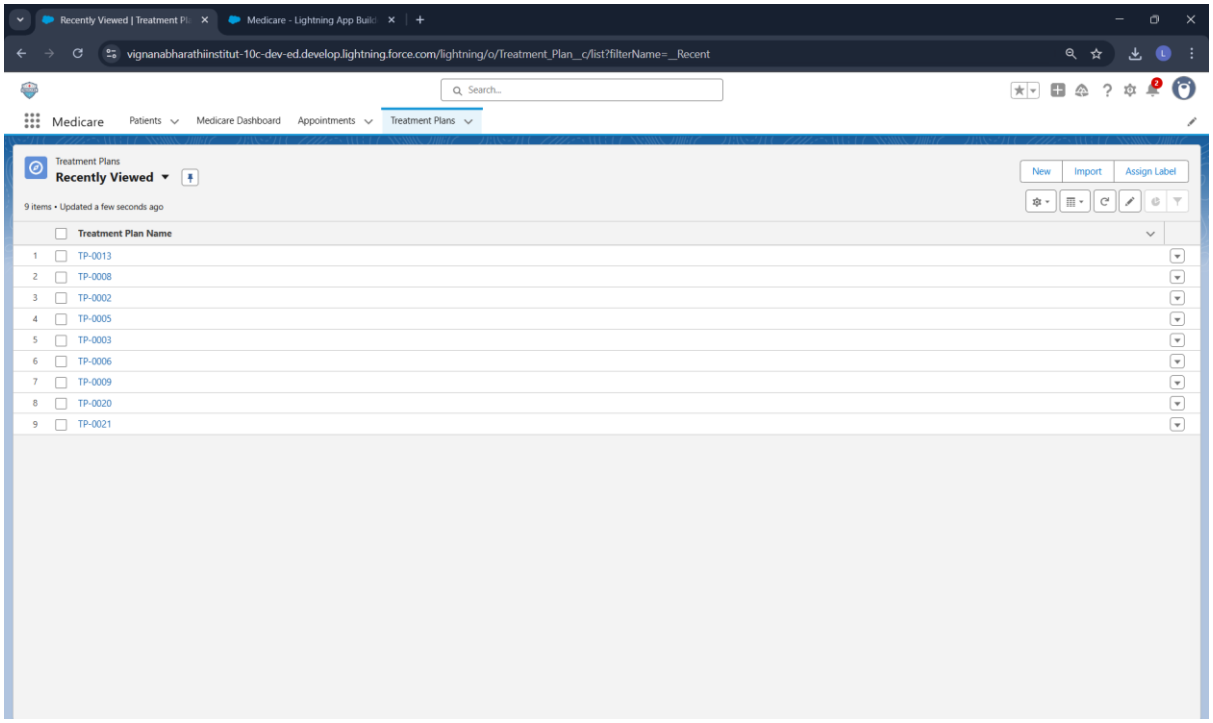
**Test Steps:**

1. Change appointment status to "Completed" (from Test Case 3)
2. Navigate to Treatment Plans tab
3. Verify new treatment plan exists

**Expected Result:**

- Treatment Plan created automatically
- Treatment Plan Number: TP-0001
- Appointment linked correctly
- Plan Details: "Treatment plan created for completed appointment"
- Start Date: Today's date
- End Date: 30 days from today
- Follow Up Required: Checked

**Actual Result:** Treatment plan created automatically with all expected field values




---

**Test Case 5: View Related Records**

**Use Case:** Verify relationships between Patient, Appointment, and Treatment Plan

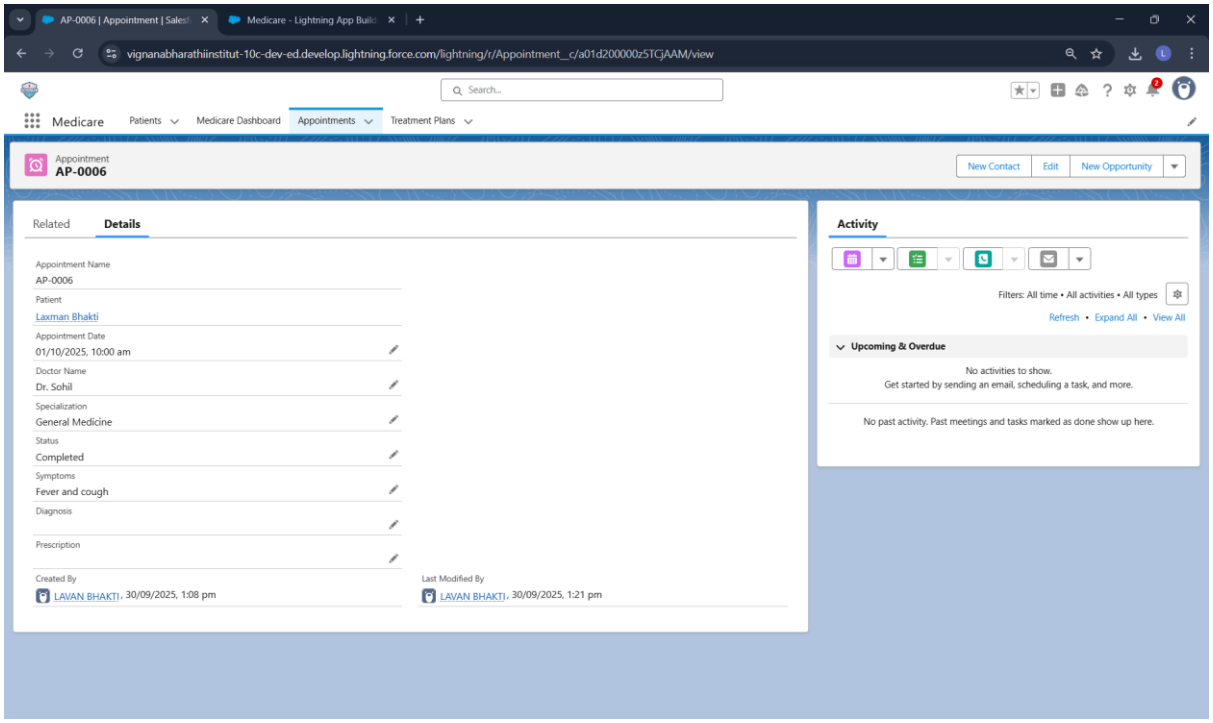
**Test Steps:**

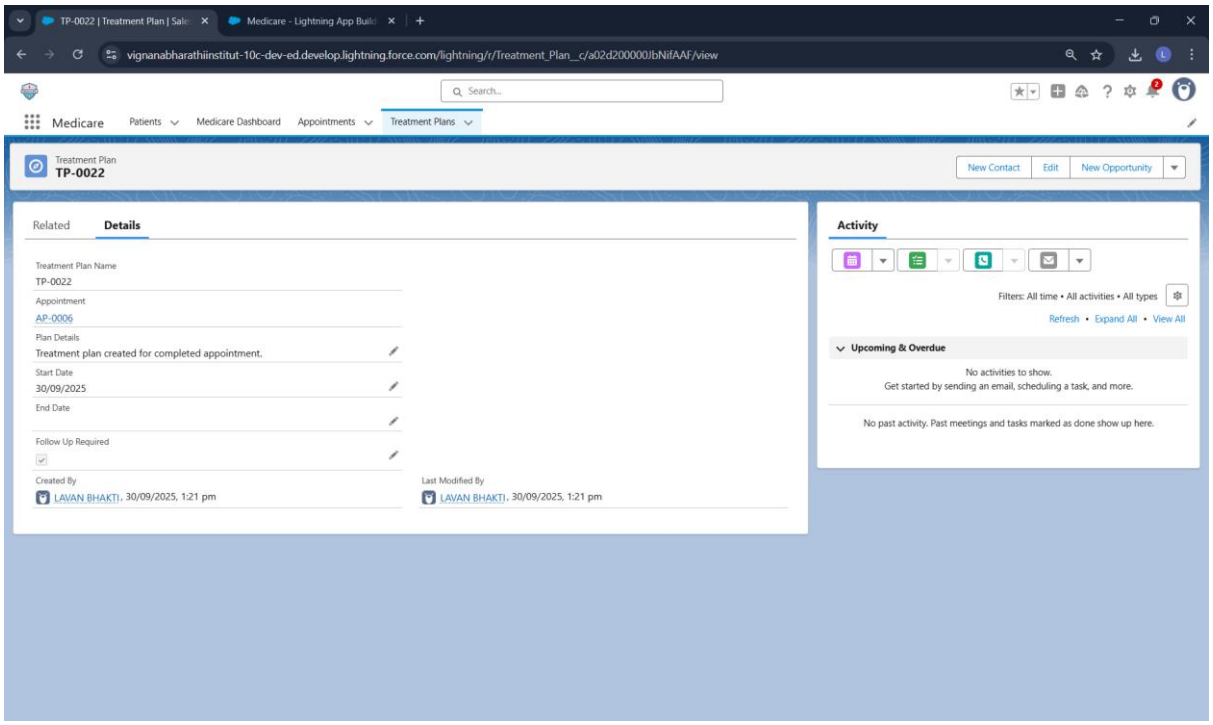
1. Open Patient record: Laxman Bhakti
2. Navigate to Related tab
3. Verify Appointments section shows created appointment
4. Open the Appointment record
5. Navigate to Related tab
6. Verify Treatment Plans section shows created treatment plan

**Expected Result:**

- Patient shows related appointments
- Appointment shows related treatment plans
- Master-Detail relationships working correctly

**Actual Result:** All relations  hips displayed correctly.




---

**Test Case 6: LWC Data Refresh**

**Use Case:** Verify dashboard updates after creating appointment

**Test Steps:**

1. Open Medicare Dashboard

2. Select patient with no appointments
3. Note "No upcoming appointments found" message
4. Create new appointment
5. Verify appointment immediately appears in list without page refresh

**Expected Result:**

- List refreshes automatically using refreshApex
- New appointment visible immediately

**Actual Result:** Dashboard refreshed automatically, new appointment visible

---

**Test Case 7: Form Validation**

**Use Case:** Verify form validation prevents incomplete data entry

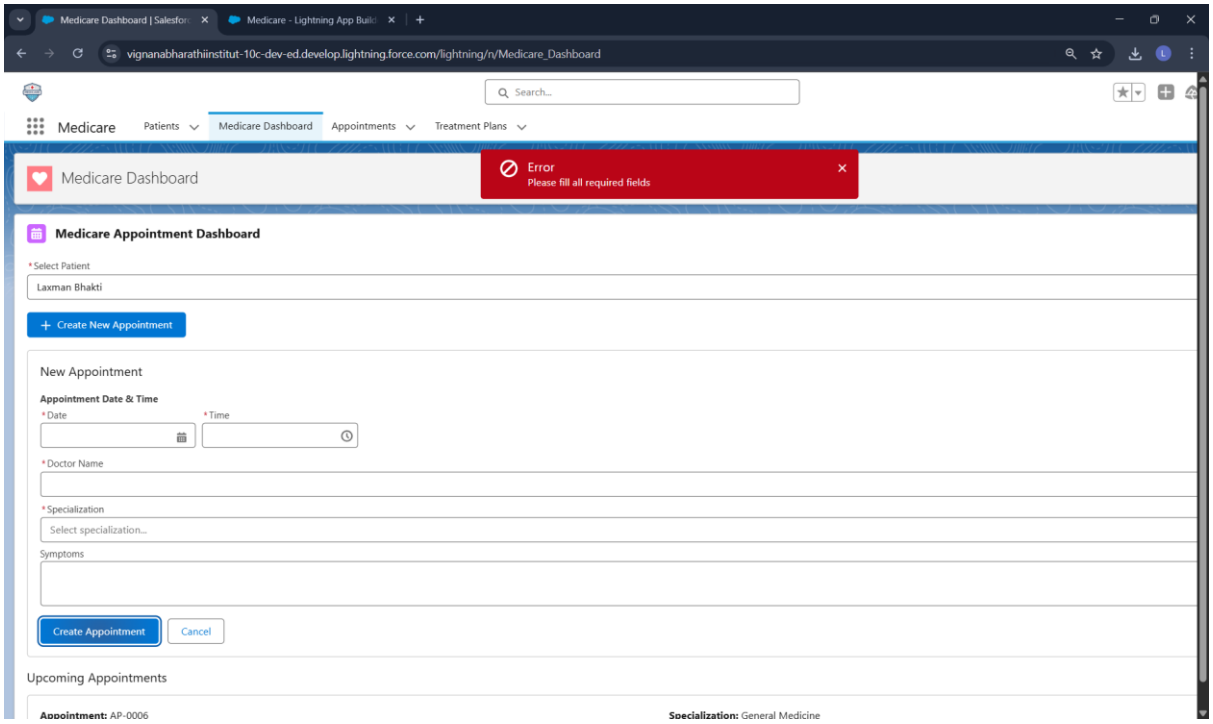
**Test Steps:**

1. Open Medicare Dashboard
2. Click "Create New Appointment" without selecting patient
3. Verify warning message
4. Select patient and click "Create New Appointment"
5. Leave required fields empty and click "Create Appointment"
6. Verify error message

**Expected Result:**

- Cannot create appointment without selecting patient
- Cannot submit form without required fields (Date, Doctor, Specialization)
- User-friendly error messages displayed

**Actual Result:** All validations working correctly




---

**Test Case 8: Flow Debug Testing**

**Use Case:** Test flow using Salesforce Flow Debug

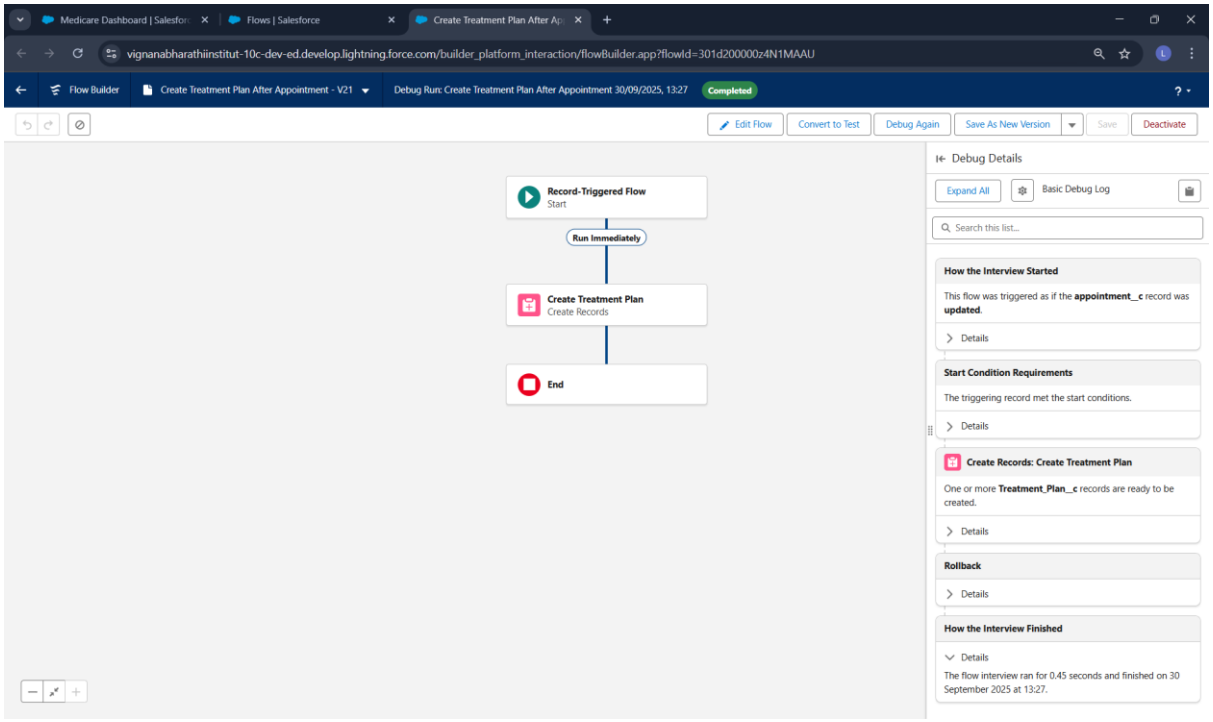
**Test Steps:**

1. Open flow "Create Treatment Plan After Appointment"
2. Click Debug
3. Select test appointment record
4. Set Status = Completed
5. Click Run
6. Review debug results

**Expected Result:**

- Start element triggers (green)
- Entry condition met (Status = Completed)
- Create Records element executes (green)
- Treatment plan record created
- All fields populated correctly

**Actual Result:** Flow executed successfully in debug mode


Medicare Dashboard | Salesforce
Flows | Salesforce
Create Treatment Plan After Ap
vignanabharathiinstitut-10c-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301d200000z4N1MAAU
Flow Builder
Create Treatment Plan After Appointment - V21
Debug Run: Create Treatment Plan After Appointment 30/09/2025, 13:27
Completed
Edit Flow
Convert to Test
Debug Again
Save As New Version
Save
Deactivate
Record-Triggered Flow
Start
Run Immediately
Create Treatment Plan
Create Records
End
Debug Details
Expand All
Basic Debug Log
Search this list...
How the Interview Started
This flow was triggered as if the appointment__c record was updated.
Details
Start Condition Requirements
The triggering record met the start conditions.
Details
Create Records: Create Treatment Plan
One or more Treatment_Plan__c records are ready to be created.
Details
Rollback
Details
How the Interview Finished
Details
The flow interview ran for 0.45 seconds and finished on 30 September 2025 at 13:27.

## Conclusion

The MediCare Patient Appointment & Treatment Assistant successfully demonstrates a complete Salesforce CRM implementation for healthcare appointment management. The project includes:

**Technical Components:**

- 3 Custom Objects (Patient, Appointment, Treatment Plan)
- 1 Apex Class (AppointmentController with 4 methods)
- 1 Lightning Web Component (appointmentDashboard)
- 1 Record-Triggered Flow (auto-creates treatment plans)
- Master-Detail relationships for data integrity
- Lightning App Builder page for UI
- Custom Medicare application

**Business Value:**

- Automated treatment plan creation ensures no patient is lost to follow-up
- Interactive dashboard streamlines appointment scheduling workflow
- Real-time status updates improve operational efficiency
- Complete digital records eliminate paper-based processes
- Reporting capabilities enable data-driven decisions

**Key Achievements:**

- Successfully deployed Apex and LWC using VS Code and SFDX
- Implemented reactive UI using wire adapters and imperative Apex calls
- Created automated workflow using Flow Builder
- Established proper data relationships with Master-Detail
- Comprehensive testing validates all functionality

**Future Enhancements:**

- Email notifications for appointment reminders
- SMS integration for patient communication
- Doctor object with many-to-many relationships
- Advanced reporting and analytics
- Mobile app optimization

- Integration with external lab systems

The MediCare project demonstrates proficiency in Salesforce platform capabilities including declarative automation (Flows), programmatic development (Apex), modern UI development (LWC), data modeling, and deployment tools. All components work together seamlessly to deliver a functional healthcare CRM solution.

---

My Name : Lavan Bhakti

GitHub Repository Link : https://github.com/lavanbhakti/MediCare-Application-project

Linkedin Profile link : https://www.linkedin.com/in/lavan-bhakti-4323a02a5

Salesforce Profile link : https://www.salesforce.com/trailblazer/lavanbhakti

Leetcode link : https://leetcode.com/u/bhaktilavan/

HackerRank Link : https://www.hackerrank.com/profile/bhaktilavan