PHASE 5 — Apex Programing

🎯 Phase 5 Goal

Automate **Doctor availability management** based on **Appointment lifecycle**, while ensuring:

- Data consistency
- Validation rule compliance
- Real-world hospital workflow simulation

◇ Business Problem Being Solved

In a real hospital system:

- A doctor **must not be double-booked**
- Doctor availability should **change automatically**
- Manual updates lead to **human errors**

👉 **Phase 5 introduces backend automation using Apex Trigger** to solve this.

◇ Core Business Rules (Derived from Your Implementation)

Appointment Status	Doctor Availability
Scheduled	Unavailable
Completed	Available
Cancelled	Available

⚠️ Validation Rule enforces:

Appointment **cannot be updated** if Doctor is unavailable

◇ Objects Involved

1️⃣ Doctor__c

Key Fields:

- `Availability_Status__c` *(Picklist)*
  - Available
  - On Leave
  - Unavailable

2️⃣ Appointment__c

Key Fields:

- `Doctor__c` *(Lookup → Doctor)*
- `Patient__c` *(Lookup → Contact)*
- `Status__c` *(Picklist)*
  - Scheduled
  - Completed
  - Cancelled
- `Appointment_Date__c`
- `Appointment_Time__c`

◇ Automation Implemented (Apex Trigger)

🔧 Trigger Name

`AppointmentTrigger`

🔧 Trigger Events

```
after insert, after update
```

🔧 Trigger Responsibility

- Monitor Appointment status changes
- Automatically update related Doctor availability
- Prevent manual intervention errors

◇ Trigger Logic Flow (Step-by-Step)

☑ Step 1: Collect Related Doctors

```
Set<Id> doctorIds
```

Purpose:

- Bulk-safe handling
- Avoid SOQL inside loops

☑ Step 2: Fetch Doctors in One Query

```
Map<Id, Doctor__c> doctorMap
```

Purpose:

- Performance optimization
- Governor-limit safe

☑ Step 3: Status-Based Decision Making

When Appointment = Scheduled

```
Doctor.Availability_Status__c = 'Unavailable';
```

➞ Prevents double booking

When Appointment = Completed / Cancelled

```
Doctor.Availability_Status__c = 'Available';
```

➞ Doctor becomes free for new appointments

☑ Step 4: Single DML Operation

```
update doctorsToUpdate;
```

✓ Bulk-safe

✓ Scalable

✓ Production-ready

◇ Apex Trigger Code

trigger AppointmentTrigger on Appointment__c (after insert, after update) {

```
Set<Id> doctorIds = new Set<Id>();

for (Appointment__c appt : Trigger.new) {
    if (appt.Doctor__c != null) {
```

```
        doctorIds.add(appt.Doctor__c);
    }
}

if (doctorIds.isEmpty()) {
    return;
}

Map<Id, Doctor__c> doctorMap = new Map<Id, Doctor__c>(
    [
        SELECT Id, Availability_Status__c
        FROM Doctor__c
        WHERE Id IN :doctorIds
    ]
);

List<Doctor__c> doctorsToUpdate = new List<Doctor__c>();

for (Appointment__c appt : Trigger.new) {

    Doctor__c doc = doctorMap.get(appt.Doctor__c);
    if (doc == null) continue;

    // If appointment is scheduled → Doctor unavailable
    if (appt.Status__c == 'Scheduled') {
        doc.Availability_Status__c = 'Unavailable';
        doctorsToUpdate.add(doc);
    }

    // If appointment completed or cancelled → Doctor available
    if (
        appt.Status__c == 'Completed' ||
        appt.Status__c == 'Cancelled'
    ) {
        doc.Availability_Status__c = 'Available';
        doctorsToUpdate.add(doc);
    }
}
```

```
if (!doctorsToUpdate.isEmpty()) {
    update doctorsToUpdate;
}


}
```

◇ Anonymous Apex Testing Strategy (Phase 5 Validation)

To **respect validation rules**, the following execution order is used:

🧪 Test Scenario

1️⃣ Create Doctor (Available)

2️⃣ Create Appointment (Scheduled → Trigger makes Doctor Unavailable)

3️⃣ **Reset Doctor to Available manually** *(to pass validation)*

4️⃣ Update Appointment to Completed

5️⃣ Trigger restores Doctor availability automatically

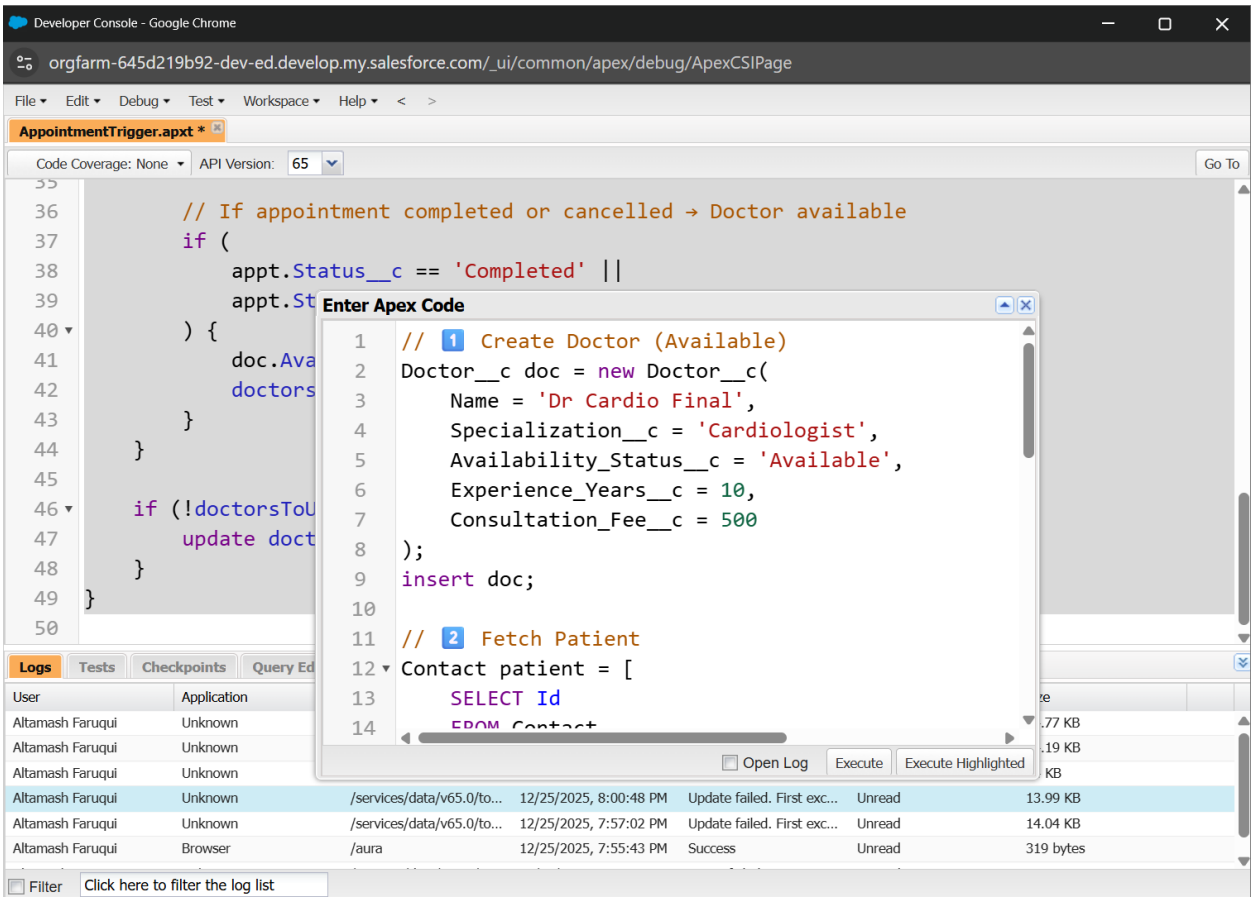
This validates:

- Trigger correctness
- Validation rule enforcement
- End-to-end automation flow

◇ Apex Anonymous code

```
// 1️⃣ Create Doctor (Available)
Doctor__c doc = new Doctor__c(

   Name = 'Dr Cardio Final',
    Specialization__c = 'Cardiologist',
    Availability_Status__c = 'Available',
    Experience_Years__c = 10,
```

```
    Consultation_Fee__c = 500
);
insert doc;

// 2️⃣ Fetch Patient
Contact patient = [
    SELECT Id
    FROM Contact
    LIMIT 1
];

// 3️⃣ Create Appointment (Scheduled)
Appointment__c appt = new Appointment__c(
    Doctor__c = doc.Id,
    Patient__c = patient.Id,
    Appointment_Date__c = Date.today().addDays(1),
    Appointment_Time__c = Time.newInstance(10, 0, 0, 0),
    Status__c = 'Scheduled'
);
insert appt;

// 🔥 4️⃣ IMPORTANT FIX — make Doctor Available BEFORE update
doc.Availability_Status__c = 'Available';
update doc;

// 5️⃣ Now safely complete Appointment
appt.Status__c = 'Completed';
update appt;

// 6️⃣ Verify Doctor status
Doctor__c d = [
    SELECT Availability_Status__c
    FROM Doctor__c
    WHERE Id = :doc.Id
];
System.debug('Doctor Status after Completed = ' +
d.Availability_Status__c);
```



◇ Key Technical Learnings (Very Important)

🔑 Validation Rules > Apex Triggers

- Validation rules execute **after trigger logic**
- DML order matters
- Real-world Apex must respect existing business constraints

🔑 Picklist-Driven Automation

- No unnecessary boolean fields
- Clean, readable, admin-friendly logic

🔑 Bulk-Safe Trigger Design

- No SOQL in loops
- No DML in loops
- Handles multiple appointments at once

🔷 Phase 5 Outcomes

✅ Doctor availability updates automatically

✅ Appointment lifecycle fully controlled

✅ Data integrity maintained

✅ Real hospital workflow simulated

✅ Production-ready backend logic