Phase 5: Apex Programming (Developer)

1. Introduction

This stage focuses on using Apex to expand Salesforce beyond its built-in tools. The language is used to build custom logic, automation, and data processing that are too advanced for declarative solutions. In the *Hospital Appointment & Health Tracker System*, Apex helps manage doctor schedules, avoid appointment clashes, generate patient health records, and run background tasks like reminders.

---

2. Preparation Steps

Before coding, a few things must be set up:

Enable Developer Console and connect the Salesforce org with VS Code and SFDX.

Decide clear naming conventions for classes, triggers, and test classes.

Load sample data (Doctors, Patients, Appointments, Health Records).

Configure Remote Site Settings in case external systems need to be integrated.

---

3. Apex Components

Classes: Reusable services for managing hospital processes (appointments, health records, notifications).

Triggers: Logic that runs automatically before or after database events (e.g., stop double-booking, create health records after a visit).

Trigger Framework: Organizing triggers into handlers to keep code clean and scalable.

SOQL & SOSL: Efficient querying for data like upcoming patient visits or searching doctors by specialty.

Collections: Lists, Sets, and Maps for handling bulk data (e.g., mapping patients to their appointments).

Control Logic: If-else statements to enforce business rules, such as creating health records only when an appointment is completed.

Batch Apex: Jobs for handling large datasets (e.g., nightly reminders, weekly archiving).

Queueable Apex: Asynchronous tasks like sending bulk notifications or syncing lab data.

Scheduled Apex: Automated recurring jobs, such as daily patient reminders or weekly hospital summaries.

Future Methods: Lightweight asynchronous operations, like updating patient portals or logging wearable device data.

Exception Handling: Error management to prevent failures (e.g., showing “Doctor not available” instead of crashing).

 Test Classes: Automated testing to verify business logic and ensure at least 75% coverage for deployment.

A. Classes & Objects

Purpose: Define reusable logic for hospital operations.

Examples:

AppointmentService.cls → Manages booking, cancellation, and availability.

HealthRecordService.cls → Handles creation of health records after completed appointments.

NotificationService.cls → Sends reminders and notifications via email or push.

B. Apex Triggers

Purpose: Execute logic automatically before/after database events.

Examples:

Before Insert Trigger on Appointment__c → Prevent double booking for the same doctor at the same time.

After Update Trigger on Appointment__c → If Status = "Completed", auto-create related HealthRecord__c.

Before Delete Trigger on Patient__c → Prevent deletion if patient has active appointments.

Ensures **data integrity and automation beyond declarative tools**.

C. Trigger Design Pattern

Purpose: Maintain **clean, scalable code**.

Use a **Trigger Handler Framework** → one trigger per object, delegating logic to handler classes.

Example: AppointmentTrigger → calls methods in AppointmentHandler.

Keeps triggers **organized and reusable**.

D. SOQL & SOSL

Purpose: Query Salesforce data efficiently.

Use Cases:

Get all upcoming appointments for a patient.

Search doctors by specialization and availability.

Enables **dynamic lookups** for patients and doctors.

E. Collections (List, Set, Map)

Purpose: Store and process bulk data.

Use Cases:

Use **List** to fetch all appointments scheduled for tomorrow.

Use **Set** to avoid duplicate doctor IDs when checking availability.

Use **Map** to relate Patient IDs → Appointments for batch processing.

F. Control Statements

Used for conditional logic in hospital workflows.

Example:

```
if(appointment.Status__c == 'Completed') {
  HealthRecordService.createRecord(appointment.Patient__c);
}
```

G. Batch Apex

Purpose: Handle **large data volumes**.

Use Cases:

Nightly batch job to send reminders for next-day appointments.

Weekly batch job to archive old health records.

H. Queueable Apex

Purpose: Manage **asynchronous operations**.

Use Cases:

Push patient reports to external lab systems.

Queue background notification jobs when many appointments are created at once.

I. Scheduled Apex

Purpose: Automate recurring tasks.

Use Cases:

Run daily job at 7 AM to send SMS/email reminders to patients.

Run weekly job to summarize hospital activity for Admin.

J. Future Methods

Purpose: Execute lightweight tasks asynchronously.

Use Cases:

Send prescription updates to patient portal after appointment completion.

Log wearable device data (e.g., heart rate, steps) asynchronously.

K. Exception Handling

Purpose: Prevent system failures and provide user-friendly messages.

Example:

If doctor's schedule is full → show error "Doctor not available, please choose another time."

Catch API failures when fetching external lab results.

L. Test Classes

Purpose: Validate Apex logic and ensure minimum **75% code coverage**.

Examples:

TestAppointmentService.cls → Tests double-booking prevention.

TestHealthRecordService.cls → Tests auto-creation of health records.

TestNotificationService.cls → Tests reminders and notifications.

Ensures **reliable and deployable code**.

---

4. Benefits of Apex Programming

Handles complex scenarios beyond declarative automation.

Ensures patient safety by preventing scheduling conflicts.

Supports large-scale hospital operations via batch processing.

Improves patient experience with automated reminders and instant notifications.

Prepares platform for integrations with external health systems.

Phase 5 Deliverable:

Created Apex Classes & Triggers.

Implemented Trigger Handler design pattern.

Configured Batch, Queueable, and Scheduled Apex jobs.

Added Exception Handling and Test Classes.

Achieved >75% Code Coverage for deployment.

This forms the foundation for **Phase 6: User Interface Development**