Phase 5: Apex Programming (Developer Components)

5.1 Trigger Development

5.1.1 Appointment Trigger

Purpose: Handle appointment-related business logic, including conflict validation, setting default details, sending confirmation emails, and managing status changes.

Trigger: `AppointmentTrigger`

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

    if (Trigger.isBefore) {
        if (Trigger.isInsert || Trigger.isUpdate) {

AppointmentTriggerHandler.validateAppointmentConflicts(Trigger.new);
            AppointmentTriggerHandler.setAppointmentDetails(Trigger.new);
        }
    }

    if (Trigger.isAfter) {
        if (Trigger.isInsert) {

AppointmentTriggerHandler.sendAppointmentConfirmation(Trigger.new);
        }

        if (Trigger.isUpdate) {
            AppointmentTriggerHandler.handleStatusChanges(Trigger.new,
Trigger.oldMap);
        }
    }
}
```

Handler Class: `AppointmentTriggerHandler`

Purpose: Encapsulate business logic for the trigger to maintain modularity and scalability.

Key Methods:

1. **Prevent double booking of doctors**

```
public static void validateAppointmentConflicts(List<Appointment__c>
newAppts) {
    Set<Id> doctorIds = new Set<Id>();
    Map<String, List<Appointment__c>> doctorDateTimeMap = new Map<String, 
List<Appointment__c>>();

    // Collect doctor IDs and map appointment date-time
    for (Appointment__c appt : newAppts) {
        if (appt.Doctor__c != null && appt.Appointment_Date__c != null && 
appt.Appointment_Time__c != null) {
            doctorIds.add(appt.Doctor__c);
            String key = appt.Doctor__c + '_' + 
String.valueOf(appt.Appointment_Date__c) + '_' + 
String.valueOf(appt.Appointment_Time__c);
            if (!doctorDateTimeMap.containsKey(key)) 
doctorDateTimeMap.put(key, new List<Appointment__c>());
            doctorDateTimeMap.get(key).add(appt);
        }
    }

    // Query existing appointments
    List<Appointment__c> existingAppts = [
        SELECT Id, Doctor__c, Appointment_Date__c, Appointment_Time__c, 
Status__c
        FROM Appointment__c
        WHERE Doctor__c IN :doctorIds
        AND Status__c IN ('Scheduled', 'Confirmed', 'In Progress')
        AND Id NOT IN :Trigger.newMap.keySet()
    ];

    // Check for conflicts
    for (Appointment__c existingAppt : existingAppts) {
        String key = existingAppt.Doctor__c + '_' + 
String.valueOf(existingAppt.Appointment_Date__c) + '_' + 
String.valueOf(existingAppt.Appointment_Time__c);
        if (doctorDateTimeMap.containsKey(key)) {
            for (Appointment__c conflictingAppt : doctorDateTimeMap.get(key)) 
{
                conflictingAppt.addError('This doctor already has an 
appointment at the selected time.');
            }
        }
    }
}
```

2. **Set appointment details (consultation fee, duration, etc.)**

```
public static void setAppointmentDetails(List<Appointment__c> appointments) {
    Set<Id> doctorIds = new Set<Id>();
    for (Appointment__c appt : appointments) {
        if (appt.Doctor__c != null) doctorIds.add(appt.Doctor__c);
    }

    Map<Id, Doctor__c> doctorMap = new Map<Id, Doctor__c>([
        SELECT Id, Consultation_Fee__c, Specialization__c
        FROM Doctor__c
        WHERE Id IN :doctorIds
    ]);
    for (Appointment__c appt : appointments) {
        if (doctorMap.containsKey(appt.Doctor__c)) {
            if (appt.Duration_Minutes__c == null) appt.Duration_Minutes__c =
30; // default duration
        }
    }
}
```

3. **Send confirmation emails to patients**

```
public static void sendAppointmentConfirmation(List<Appointment__c>
appointments) {
    List<Messaging.SingleEmailMessage> emails = new
List<Messaging.SingleEmailMessage>();

    for (Appointment__c appt : appointments) {
        if (appt.Patient__c != null && appt.Status__c == 'Scheduled') {
            Messaging.SingleEmailMessage email = new
Messaging.SingleEmailMessage();
            email.setTargetObjectId(appt.Patient__c);
            email.setSubject('Appointment Confirmation - ' + appt.Name);
            email.setPlainTextBody('Your appointment has been scheduled
successfully.');
            email.setSaveAsActivity(true);
            emails.add(email);
        }
    }

    if (!emails.isEmpty()) Messaging.sendEmail(emails);
}
```

4. **Handle appointment status changes**

- Create Medical Case and Bill records automatically when appointment status changes.

```
public static void handleStatusChanges(List<Appointment__c> newAppts, Map<Id,
Appointment__c> oldMap) {
    List<Medical_Case__c> casesToCreate = new List<Medical_Case__c>();
    List<Bill_Information__c> billsToCreate = new
List<Bill_Information__c>();

    for (Appointment__c appt : newAppts) {
        Appointment__c oldAppt = oldMap.get(appt.Id);
        // Example logic: create medical case or bill when status changes
from Scheduled -> Completed
        if(oldAppt.Status__c != 'Completed' && appt.Status__c ==
'Completed'){
            casesToCreate.add(new Medical_Case__c(
                Patient__c = appt.Patient__c,
                Doctor__c = appt.Doctor__c,
                Related_Appointment__c = appt.Id,
                Case_Status__c = 'Open',
                Case_Date__c = Date.today()
            ));
            billsToCreate.add(new Bill_Information__c(
                Patient__c = appt.Patient__c,
                Related_Appointment__c = appt.Id,
                Doctor__c = appt.Doctor__c,
                Consultation_Fee__c = appt.Doctor__r.Consultation_Fee__c,
                Bill_Date__c = Date.today(),
                Payment_Status__c = 'Pending'
            ));
        }
    }

    if(!casesToCreate.isEmpty()) insert casesToCreate;
    if(!billsToCreate.isEmpty()) insert billsToCreate;
}
```

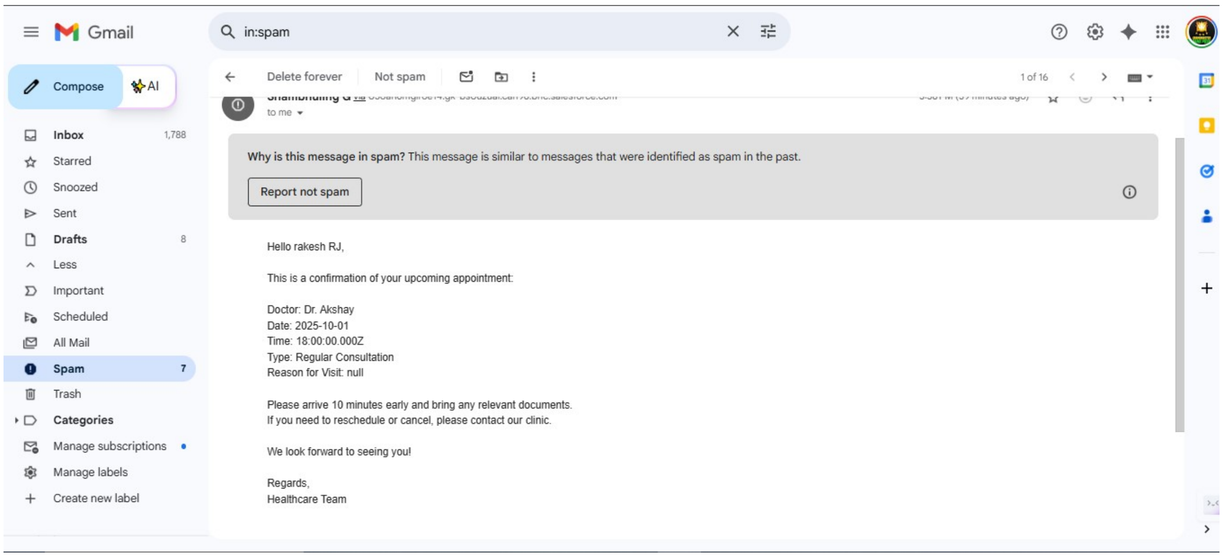
5.2 Summary of Apex Components

- **Triggers:** Handle appointment logic before and after insert/update.
- **Handler Classes:** Encapsulate logic for modularity and maintainability.
- **Key Features Implemented:**
    - Prevent double booking of doctors
    - Set default appointment details (duration, fees)
    - Send confirmation emails automatically
    - Handle status changes and create related Medical Case and Bill records

✅**Benefits:**

- Reduces manual errors
- Automates communication with patients
- Ensures accurate billing and record keeping
- Scalable and easy to maintain

Apex code Automation Screen Shoot



Why is this message in spam? This message is similar to messages that were identified as spam in the past.

Hello rakesh RJ,

This is a confirmation of your upcoming appointment:

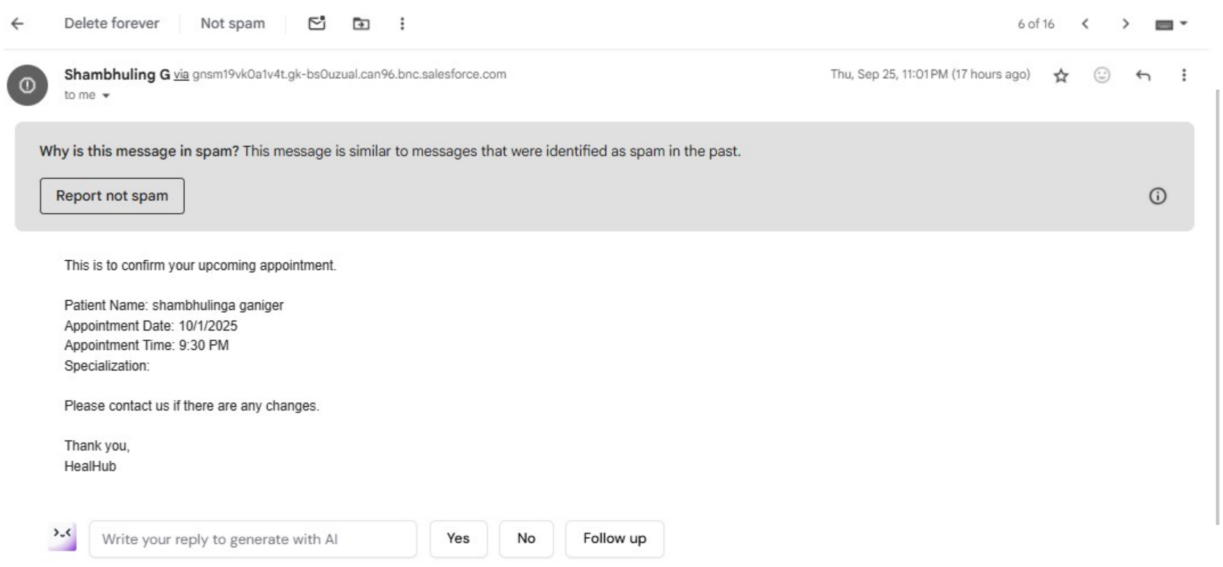
Doctor: Dr. Akshay
Date: 2025-10-01
Time: 18:00:00.000Z
Type: Regular Consultation
Reason for Visit: null

Please arrive 10 minutes early and bring any relevant documents.
If you need to reschedule or cancel, please contact our clinic.

We look forward to seeing you!

Regards,
Healthcare Team

Email Notification Trigger Flow



Shambhuling G via gnsm19vk0a1v4t.gk-bs0uzual.can96.bnc.salesforce.com
to me
Thu, Sep 25, 11:01 PM (17 hours ago)

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

This is to confirm your upcoming appointment.

Patient Name: shambhulinga ganiger
Appointment Date: 10/1/2025
Appointment Time: 9:30 PM
Specialization:

Please contact us if there are any changes.

Thank you,
HealHub