Phase 5: Apex Programming (Developer)

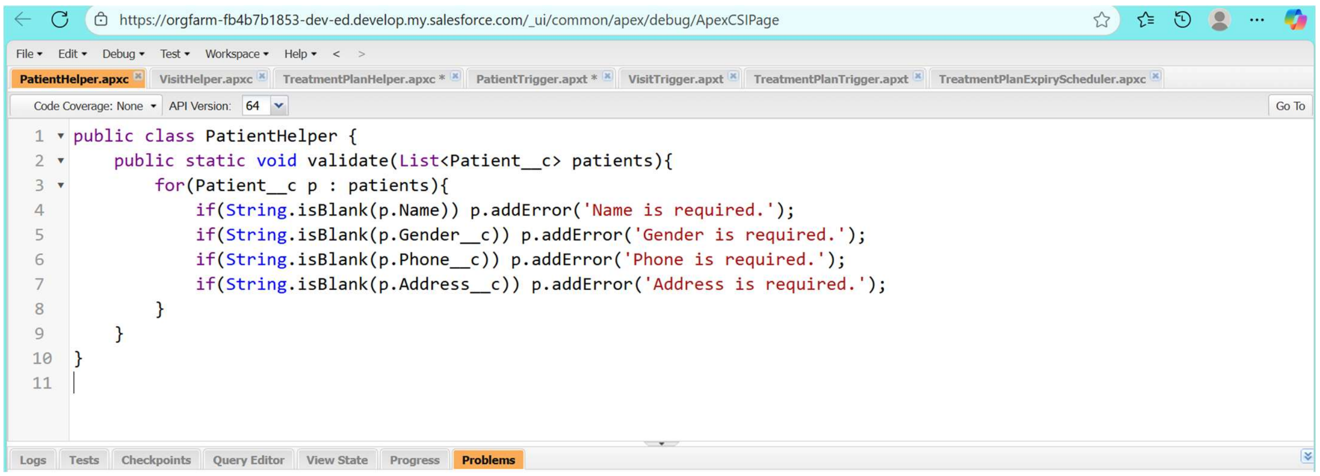
Overview

In Phase 5, we implemented Apex Programming to extend automation beyond declarative tools.
Apex was used for custom logic to manage patient records, treatment plans, and scheduled visits, ensuring real-time updates, data consistency, and scalability.

1. Apex Classes & Objects

Implemented Classes:

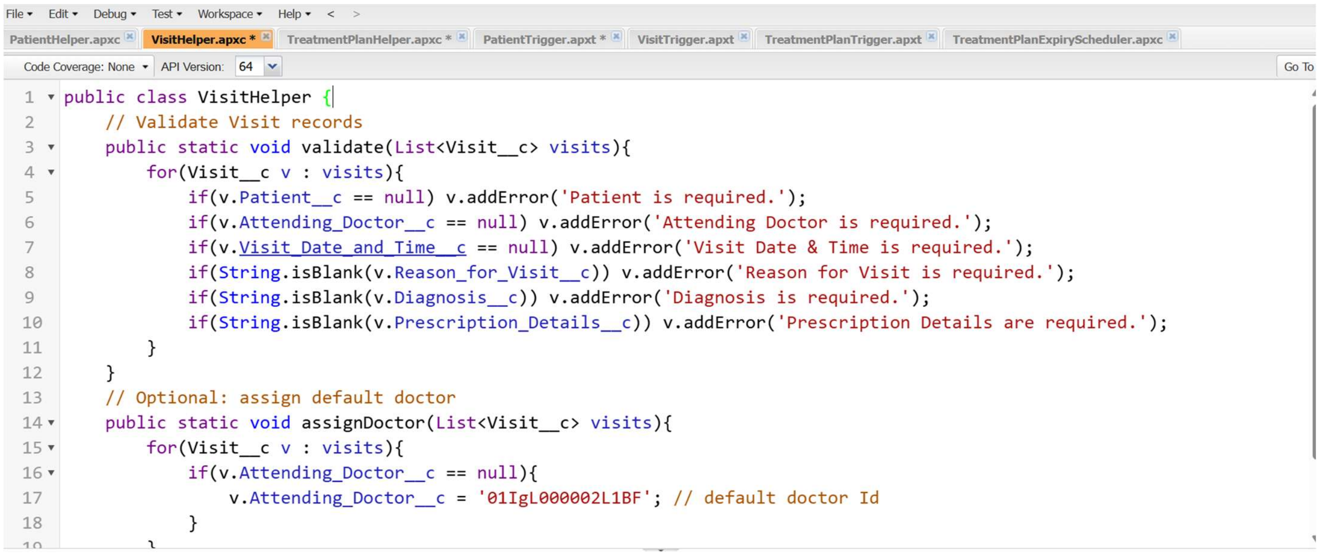
- **PatientHelper.cls**
    - Contains core logic for validating patient details.
    - Prevents recursion using a static flag.
    - Verifies required fields (Name, Date of Birth, Gender, Email).
    - Updates related treatment plans when patient data changes.


https://orgfarm-fb4b7b1853-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
PatientHelper.apxc
VisitHelper.apxc
TreatmentPlanHelper.apxc *
PatientTrigger.apxt *
VisitTrigger.apxt
TreatmentPlanTrigger.apxt
TreatmentPlanExpiryScheduler.apxc
Code Coverage: None
API Version: 64
Go To
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems

```
public class PatientHelper {
    public static void validate(List<Patient__c> patients){
        for(Patient__c p : patients){
            if(String.isBlank(p.Name)) p.addError('Name is required.');
            if(String.isBlank(p.Gender__c)) p.addError('Gender is required.');
            if(String.isBlank(p.Phone__c)) p.addError('Phone is required.');
            if(String.isBlank(p.Address__c)) p.addError('Address is required.');
        }
    }
}
```

- **TreatmentPlanHandler.cls**
    - Processes treatment plan records.
    - Calculates treatment duration and sets status (Planned → Active → Completed).
    - Ensures patient association and required approvals before activation.

- **VisitScheduler.cls**
    - Handles scheduling and reminders for patient visits.
    - Calculates next-visit dates and sends notification updates to related treatment plans.

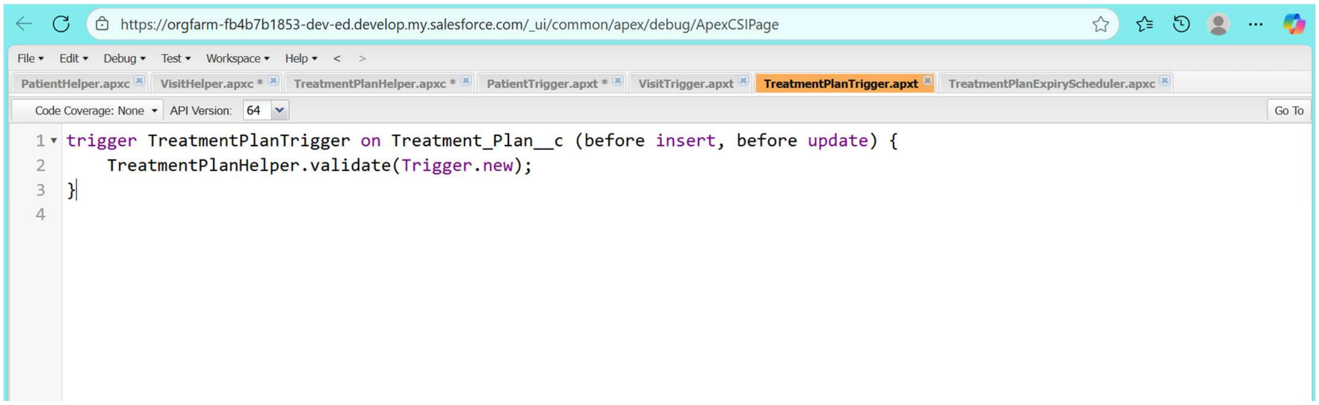

File Edit Debug Test Workspace Help < >
PatientHelper.apxc
VisitHelper.apxc *
TreatmentPlanHelper.apxc *
PatientTrigger.apxt *
VisitTrigger.apxt
TreatmentPlanTrigger.apxt
TreatmentPlanExpiryScheduler.apxc
Code Coverage: None
API Version: 64
Go To

```
public class VisitHelper {
    // Validate Visit records
    public static void validate(List<Visit__c> visits){
        for(Visit__c v : visits){
            if(v.Patient__c == null) v.addError('Patient is required.');
            if(v.Attending_Doctor__c == null) v.addError('Attending Doctor is required.');
            if(v.Visit_Date_and_Time__c == null) v.addError('Visit Date & Time is required.');
            if(String.isBlank(v.Reason_for_Visit__c)) v.addError('Reason for Visit is required.');
            if(String.isBlank(v.Diagnosis__c)) v.addError('Diagnosis is required.');
            if(String.isBlank(v.Prescription_Details__c)) v.addError('Prescription Details are required.');
        }
    }
    // Optional: assign default doctor
    public static void assignDoctor(List<Visit__c> visits){
        for(Visit__c v : visits){
            if(v.Attending_Doctor__c == null){
                v.Attending_Doctor__c = '01IgL000002L1BF'; // default doctor Id
            }
```

2. Apex Triggers (after insert, after update)

Implemented Triggers:

- **PatientTrigger.trigger**
  - Fires on **Patient__c** after insert and update.
  - Calls PatientHelper.processPatients() to validate fields and update related treatment plans.
- **TreatmentPlanTrigger.trigger**
  - Fires on **Treatment_Plan__c** after insert and update.
  - Calls TreatmentPlanHandler.processPlans() to manage status changes and enforce business rules.
- **VisitTrigger.trigger**
  - Fires on **Visit__c** after insert and update.
  - Calls VisitScheduler.processVisits() to manage visit scheduling and reminders.


https://orgfarm-fb4b7b1853-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
PatientHelper.apxc
VisitHelper.apxc *
TreatmentPlanHelper.apxc *
PatientTrigger.apxt *
VisitTrigger.apxt
TreatmentPlanTrigger.apxt
TreatmentPlanExpiryScheduler.apxc
Code Coverage: None
API Version: 64
Go To

```
trigger TreatmentPlanTrigger on Treatment_Plan__c (before insert, before update) {
    TreatmentPlanHelper.validate(Trigger.new);
}
```

3. Trigger Design Pattern

- **One Trigger per Object** principle followed.
- Logic is delegated to handler classes, keeping triggers lean.

- Added a **static Boolean flag** in each handler to prevent recursive updates.

4. SOQL

Used SOQL queries to fetch related records:

- Fetch Patients with required fields.
- Fetch Treatment Plans filtered by status and patient ID.
- Fetch Visits scheduled for specific dates.

Example:

```
List<Treatment_Plan__c> plans = [
   SELECT Id, Status__c, Patient__c
   FROM Treatment_Plan__c
   WHERE Status__c IN :statuses
];
```

5. Collections: List, Set, Map

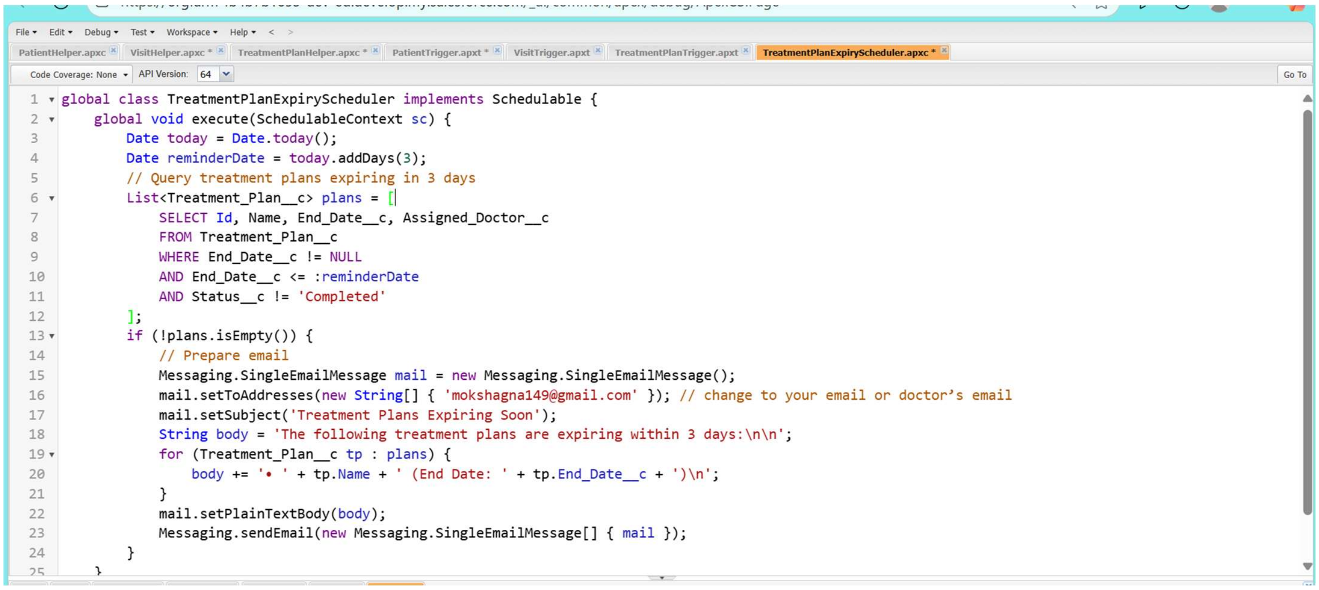
- **List:** Stored treatment plans and visits for bulk updates.
- **Set:** Collected unique patient IDs to avoid duplicates.
- **Map:** Grouped treatment plans by patient for quick lookups *(Map<Id, List<Treatment_Plan__c>>)*.

6. Control Statements

- **If-Else:** Checked required fields and plan status transitions.
- **For Loops:** Iterated through patient and plan records to update statuses.
- **Break Statements:** Stopped loops once necessary updates were applied.

7. Scheduled Apex

- **TreatmentPlanExpiryScheduler.cls** implements **Schedulable**.
- Runs **daily** to:
  - Find treatment plans nearing their **end date**.
  - Send **automated email alerts** to the assigned doctor/nurse.
  - Notify patients about upcoming plan expiry.
  - Update status if the treatment plan is overdue.


File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
PatientHelper.apxc
VisitHelper.apxc *
TreatmentPlanHelper.apxc *
PatientTrigger.apxt *
VisitTrigger.apxt
TreatmentPlanTrigger.apxt
TreatmentPlanExpiryScheduler.apxc *
Code Coverage: None ▾
API Version: 64
Go To

```
global class TreatmentPlanExpiryScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        Date today = Date.today();
        Date reminderDate = today.addDays(3);
        // Query treatment plans expiring in 3 days
        List<Treatment_Plan__c> plans = [
            SELECT Id, Name, End_Date__c, Assigned_Doctor__c
            FROM Treatment_Plan__c
            WHERE End_Date__c != NULL
            AND End_Date__c <= :reminderDate
            AND Status__c != 'Completed'
        ];
        if (!plans.isEmpty()) {
            // Prepare email
            Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
            mail.setToAddresses(new String[] { 'mokshagna149@gmail.com' }); // change to your email or doctor’s email
            mail.setSubject('Treatment Plans Expiring Soon');
            String body = 'The following treatment plans are expiring within 3 days:\n\n';
            for (Treatment_Plan__c tp : plans) {
                body += '• ' + tp.Name + ' (End Date: ' + tp.End_Date__c + ')\n';
            }
            mail.setPlainTextBody(body);
            Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
        }
    }
```

Conclusion

Phase 5 introduced **Apex-driven automation** into the Patient Records and Treatment Tracking system. With handler classes, triggers, SOQL queries, collections, and a scheduled job, the system now:

- Validates patient details in real time.
- Automatically updates treatment plan statuses.
- Schedules and reminds patient visits daily.
- Maintains data consistency and scalability beyond point-and-click automation.