Phase 5: Apex Programming (Developer)

Goal: Add advanced logic and automation for Leave Tracking App.

1. Classes & Objects

- Create a **LeaveService** class to handle reusable logic like leave validation, balance calculation, and notifications.
- Example: LeaveService.calculateRemainingLeaves(UserId)

2. Apex Triggers

- On **LeaveRequest Insert** → Prevent overlapping leave requests for the same employee.
- On **LeaveRequest Update** → Ensure only Pending requests can be edited.

3. Trigger Design Pattern

- Implement a **LeaveRequestTriggerHandler** class.
- Trigger file only calls the handler methods → keeps code clean and testable.

4. SOQL & SOSL

Example SOQL:

```
SELECT Id, Status__c, From_Date__c, To_Date__c
FROM LeaveRequest__c
WHERE User__c = :UserInfo.getUserId()
AND Status__c = 'Approved'
```

- Use to fetch approved leaves and check for overlaps.

5. Collections: List, Set, Map

- Use **Set<Date>** to store applied leave dates and prevent duplicates.
- Use **Map<Id, LeaveRequest__c>** to handle bulk triggers efficiently.

6. Control Statements

Example:

```
 IF (newLeave.From_Date__c <= existingLeave.To_Date__c
    && newLeave.To_Date__c >= existingLeave.From_Date__c) {
    throw new LeaveException('Leave dates overlap with an existing request.');
}
```

7. Batch Apex

- Nightly job to **mark expired leave requests** (where `To_Date__c < TODAY` and Status = Pending → Status = Expired).

```
 submitHandler(event){ event.preventDefault();
 const fields={...event.detail.fields}; fields.Status__c='Pending';
 // fields.User__c=this.currentUserId;
if (new Date(fields.From_Date__c)>new Date(fields.To_Date__c))
{
this.ShowToast('From date should be grater than to date','Error','error');
 }
 else if(new Date()>new Date(fields.From_Date__c)){
 this.ShowToast('From date should not be less than Today','Error','error');
 }
else{
 this.refs.leaveRequestFrom.submit(fields);
}
}
```

8. Future Methods

- Call external **HR/Payroll API** asynchronously when a leave request is approved (to sync leave balances with payroll system).

```
@AuraEnabled
  public static void sendNotification(String email, String subject, String body) {
  Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
  mail.setToAddresses(new String[] { email });
  mail.setSubject(subject);
  mail.setPlainTextBody(body);
  Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
}
```

9. Exception Handling

- Wrap logic in **try-catch** blocks.

Example:

```
 try {
    LeaveService.validateLeave(newLeave);
} catch (Exception e) {
    throw new AuraHandledException('Error: ' + e.getMessage());
}
```

10. Test Classes

- Create sample employees and leave requests.
- Insert overlapping leave request → ensure trigger throws error.
- Approve leave → ensure balance reduces correctly.
- Coverage goal: **> 85%**.

Phase 5 Outcome:

- Robust Apex code with reusable classes and triggers.
- Asynchronous processing ensures performance.
- Proper exception handling + test coverage makes the system reliable and deployment-ready.