Lead Management & Task Tracking System
Phase 7: Integration & External Access

1. Introduction

This phase focuses on connecting Salesforce to external systems and enabling secure external access. Integration ensures real-time updates, automated data sync, and event-driven notifications for leads and tasks.

Objectives:

- Connect Salesforce to external APIs securely.
- Automate lead and task synchronization.
- Enable real-time notifications using platform events and Change Data Capture (CDC).

2. Named Credentials

Purpose: Store authentication credentials securely for external services without hardcoding in Apex.

Setup Steps:

1. Go to Setup → Named Credentials → New Named Credential.
2. Fill in the details:
    - Label & Name: LeadServiceAPI
    - URL: https://api.externalcrm.com/
    - Identity Type: Named Principal
    - Authentication Protocol: OAuth 2.0 / Password Authentication
3. Save the credential.

Example Use (HTTP Callout in Apex):

```
HttpRequest req = new HttpRequest();

req.setEndpoint('callout:LeadServiceAPI/leads');

req.setMethod('POST');

req.setHeader('Content-Type', 'application/json');
```

```
req.setBody(JSON.serialize(newLead));

Http http = new Http();

HTTPResponse res = http.send(req);

System.debug('Response: ' + res.getBody());
```

Benefits:

- Secure and maintainable authentication.
- Easy to update credentials without changing code.

Deliverable: Named Credentials configured for external lead sync.

3. External Services

Purpose: Register external REST APIs declaratively to use in Flows or Apex.

Setup Steps:

1. Go to Setup → External Services → New External Service.
2. Upload the OpenAPI specification JSON/YAML from the external API.
3. Assign a Named Credential (created above) for authentication.
4. Generate Flow actions automatically from the external service.

Example Use:

- Call a lead scoring API to enrich new leads automatically in Salesforce Flow.

Benefits:

- Declarative integration without coding.
- Faster, maintainable API integration.

Deliverable: External Service registered and integrated into a Flow.

4. Web Services (REST/SOAP)

Purpose: Expose Salesforce data externally or consume external APIs.

Setup Steps (Exposing Salesforce REST API):

1. Go to Setup → Object Manager → Lead → Buttons, Links, and Actions → New REST Resource.

2. Define a custom Apex REST class:

```
@RestResource(urlMapping='/LeadsAPI/*')
global with sharing class LeadsRestController {
    @HttpGet
    global static Lead getLeadById() {
        RestRequest req = RestContext.request;
        String leadId = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
        return [SELECT Id, Name, Company, Status FROM Lead WHERE Id =
:leadId];
    }

    @HttpPost
    global static String createLead(String name, String company, String status) {
        Lead l = new Lead(Name=name, Company=company, Status=status);
        insert l;
        return l.Id;
    }
}
```

Setup Steps (Consuming External SOAP API):

1. Go to Setup → Apex Classes → Generate from WSDL.
2. Upload the WSDL file from external service.
3. Use the generated Apex class to call SOAP web service.

Benefits:

- Enables bi-directional communication between Salesforce and other systems.

Deliverable: REST API endpoints exposed for leads; SOAP client integrated for external task updates.

5. Callouts

Purpose: Make HTTP requests to external APIs from Apex.

Setup Steps:

1. Create Named Credential (see above).
2. Write Apex class for callout.

Example HTTP GET Callout:

```
HttpRequest req = new HttpRequest();

req.setEndpoint('callout:LeadServiceAPI/leads');

req.setMethod('GET');

Http http = new Http();

HTTPResponse res = http.send(req);

List<Lead> leadList = (List<Lead>) JSON.deserialize(res.getBody(),
List<Lead>.class);
```

Deliverable: Apex classes for HTTP callouts to external CRM/lead scoring service.

6. Platform Events

Purpose: Real-time event-driven updates.

Setup Steps:

1. Go to Setup → Platform Events → New Platform Event.
2. Name it LeadAssigned__e with fields: LeadId, AssignedTo, Priority.
3. Create Apex trigger to handle events:

```
trigger LeadAssignedTrigger on LeadAssigned__e (after insert) {
   for(LeadAssigned__e event : Trigger.New){
      System.debug('Lead Assigned: ' + event.LeadId + ' to ' + event.AssignedTo);
   }
}
```

Deliverable: Platform events created for lead assignment notifications.

7. Change Data Capture (CDC)

Purpose: Automatically track and respond to record changes.

Setup Steps:

1. Go to Setup → Change Data Capture → Select Objects (e.g., Lead, Task).
2. Subscribe using CometD, Apex triggers, or Flows.

Example Apex Trigger for CDC:

```
trigger LeadChangeCapture on LeadChangeEvent (after insert) {
    for (LeadChangeEvent event : Trigger.New) {
        System.debug('Lead updated: ' + event.ChangeEventHeader.recordIds);
    }
}
```

Benefits:

- Real-time tracking of lead and task changes.
- Integration-ready for external systems.

Deliverable: CDC enabled for Leads and Tasks with subscribers implemented.