

## Phase 7: Integration & External Access

### CONNECT Student Success Platform - Salesforce CRM Implementation

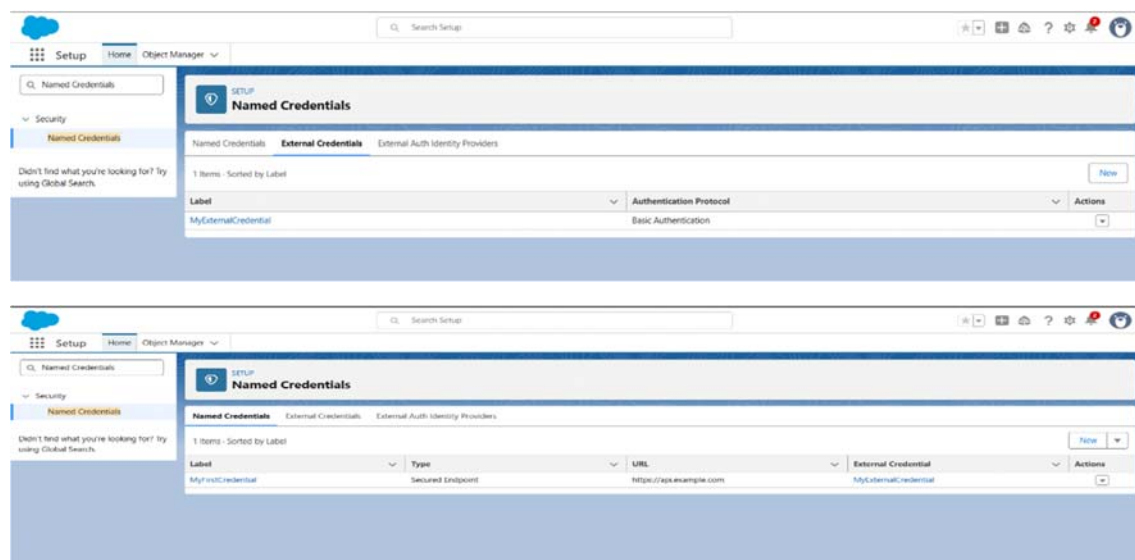
#### 7.1 Named Credentials

**Use Case:** To securely connect Salesforce to an external system or API, make API calls, and hide sensitive secrets like usernames and passwords by using "Named Credentials".

#### Implementation Steps:

1. Go to Setup → Type Named Credentials in the Quick Find box → Click Named Credentials.
2. Click the External Credentials tab → Click New → Fill in:
  - Label: MyExternalCredential
  - Name: MyExternalCredential
  - Authentication Protocol: Basic Authentication → Click Save.
3. Go back to the Named Credentials tab → Click New → Fill in:
  - Label: MyFirstCredential
  - Name: MyFirstCredential
  - URL: <https://api.example.com>
  - External Credential: Select MyExternalCredential
  - Enabled for Callouts: ON
  - Leave other settings as default → Click Save.

**Result:** This new Named Credential is now ready and linked to your External Credential. Salesforce can now safely connect to external APIs using this secure setup.



## 7.2 External Services

**Use Case:** External Services allow Salesforce to connect with any external REST API (like a Student Information System), so actions from that API can be used in Flow Builder and other automation tools in Salesforce without writing code.

### Implementation Steps:

1. Go to Setup → Type "External Services" in the Quick Find box → Click **External Services**.
2. Click **Add an External Service**.
3. On "Select an API Source" page, click **From API Specification** → Click **Next**.
4. On "Provide Registration Details":
  - External Service Name: Write a name (for example, "StudentInfoAPI").
  - Description: Write what this service does.
  - Service Schema: Choose "Absolute URL".
  - URL: Paste the link to a valid OpenAPI/Swagger JSON specification file.
  - Select a Named Credential: Pick the Named Credential already set up.
  - If you see an error about the URL, add that API site to "Remote Site Settings" (Setup → Remote Site Settings → New Remote Site).
5. Return to the registration screen and click **Save & Next**.
6. Review the detected operations from your API specification file. Click **Done**.

Now the external API's actions are available for use in Flow Builder and other process tools with simple drag & drop. You can test the connection by using one of the new Flow Actions provided by the external service.

**Result:** The external API appears as a registered service, and its operations become "invocable actions" in Salesforce Flows. This makes it easy to automate business processes that use data or actions from other systems, all without custom code.

## Phase 7.3: Web Services (REST/SOAP)

**Use Case:** Web Services (REST and SOAP) in Salesforce allow your org to expose special endpoints that other applications or platforms can access in order to create, modify, or fetch Salesforce records directly. Common use cases include connecting Salesforce with external apps, portals, or systems that need real-time data or automation.

### Why Not Implemented

This phase is not included in your implementation since the current project does not require Salesforce to "publish" its own APIs for others to consume. All needed integrations involve

Salesforce acting as the client, not the server. For this project, using Named Credentials and External Services for calling out to APIs is enough.

### **How to Implement in Future**

If future requirements need exposing Salesforce endpoints, you can:

- Use Apex REST (@RestResource Annotation) to create RESTful endpoints.
- Use Apex SOAP (with webservice methods) and generate WSDL files for clients.
- Use platform REST/SOAP APIs to give access to standard objects.

Both methods support authentication via OAuth and session tokens for secure integration.

### **Tools & Features**

- Apex classes and annotations (@RestResource for REST, webservice for SOAP).
- Workbench or Postman for testing.
- WSDL/XML for SOAP publishing; JSON for REST.

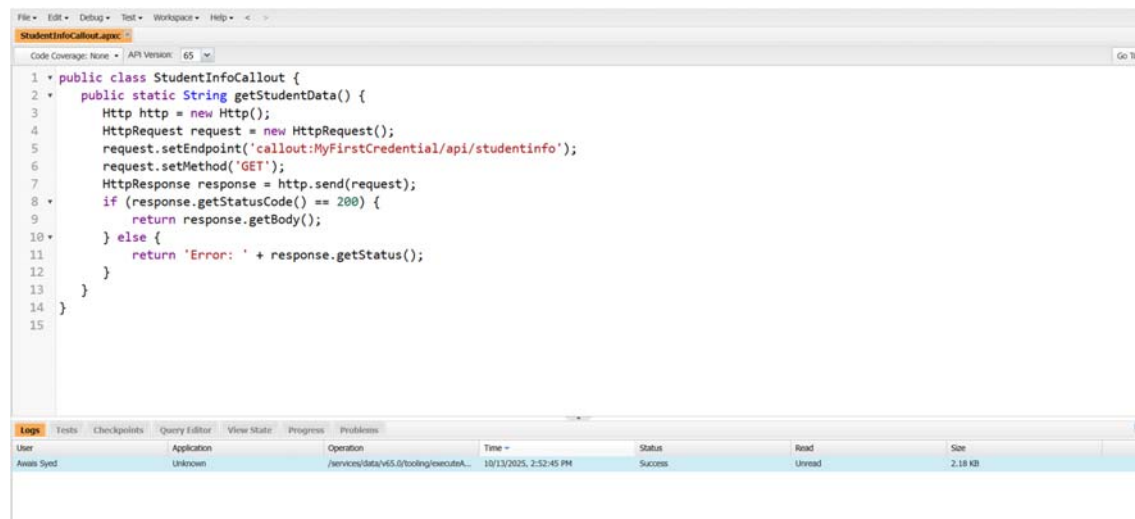
## **7.4: Callouts**

**Use Case:** Callouts in Salesforce enable your org to send HTTP requests to external systems or APIs to get or send data, trigger processes, or access external resources from Apex code or Flows. This is essential whenever Salesforce needs to "call out" to external services for integration and automation.

### **Implementation Steps:**

1. Configure **Named Credentials** to securely store endpoint URLs and authentication (already done in Phase 7.1).
2. Create an **Apex class** to perform HTTP callouts using Named Credentials.
3. Use the following Apex classes for callouts:
  - HttpRequest and HttpResponse for sending and receiving HTTP messages.
  - Http class to send requests.
4. Invoke these callouts in Apex triggers, batch classes, or expose them as invocable methods to Flows.
5. Add corresponding **Remote Site Settings** for external URLs if not using Named Credentials.
6. Test your callouts thoroughly with debug logs or the Execute Anonymous Apex window.

**Result:** Successful callouts enable real-time integration where Salesforce communicates with external APIs, bringing external data/actions into Salesforce processes. This allows building seamless automation across multiple systems without manual data entry or custom middleware.



```
1 public class StudentInfoCallout {
2     public static String getStudentData() {
3         Http http = new Http();
4         HttpRequest request = new HttpRequest();
5         request.setEndpoint('callout:MyFirstCredential/api/studentinfo');
6         request.setMethod('GET');
7         HttpResponse response = http.send(request);
8         if (response.getStatusCode() == 200) {
9             return response.getBody();
10        } else {
11            return 'Error: ' + response.getStatusCode();
12        }
13    }
14 }
15
```

User	Application	Operation	Time	Status	Read	Size
Awaiz Syed	Unknown	/services/data/v65.0/tooling/executeA...	10/13/2025, 2:52:45 PM	Success	Unread	2.18 KB

## 7.5 Platform Events

**Use Case:** Platform Events are part of Salesforce's event-driven architecture. They allow different systems to communicate by publishing and subscribing to event messages in real-time. Platform Events help integrate Salesforce with external apps or different Salesforce components asynchronously.

### Why Not Implemented:

This project does not require real-time event messaging or event-driven automation between Salesforce and external systems or internal components. The current integration scope is handled via callouts and external services which suffice for synchronous data exchange.

### How to Implement in Future

- Define a Platform Event object in Setup → Platform Events.
- Add custom fields for event data.
- Publish events using Apex, Process Builder, Flow, or external API.
- Subscribe to events asynchronously using Apex triggers, flows, or external listeners.
- Platform Events support event delivery guarantee and can scale across systems.

### Tools and Features

- Platform Events object type with custom fields
- EventBus Apex class for publishing

- Event triggers and platform event-triggered flows for subscription
- Pub/Sub API for external event management

## **7.6 Change Data Capture (CDC)**

**Use Case:** Change Data Capture (CDC) in Salesforce enables tracking of record changes (create, update, delete, undelete) on standard and custom objects in near real-time. It helps keep external systems synchronized with Salesforce data by publishing change events to subscribed clients, supporting efficient and scalable data integration and notifications.

### **Why Not Implemented:**

The current project scope doesn't require near real-time synchronization of Salesforce data changes with external systems or complex event-driven data flows. Existing integrations using callouts and external services suffice for syncing needed data.

### **How to Implement in Future**

- Enable CDC in Setup under "Change Data Capture" and select objects to track.
- Subscribe to change events using Apex triggers, CometD clients, or Streaming API.
- Process change events to replicate or act upon data changes in external systems.
- Use CDC for high volume, scalable and near real-time integration scenarios.
- Monitor CDC event usage and subscriptions for performance tuning.

### **Tools and Features**

- Setup UI for enabling CDC on objects
- Apex triggers on change events
- Streaming API with CometD for external consumption
- Event monitoring and diagnostics dashboards

## **7.7 Salesforce Connect**

**Use Case:** Salesforce Connect enables seamless real-time integration between Salesforce and external data systems by allowing users to view, search, and modify data stored outside Salesforce without importing it. It is ideal when data resides in enterprise systems like ERPs or legacy databases and needs to be accessed directly within Salesforce for up-to-date and accurate information without duplication.

### **Why Not Implemented:**

The current project scope does not require working with large external data systems or real-time external data viewing within Salesforce UI. Existing integration methods like callouts and external services are sufficient for current data access and manipulation needs.

### **How to Implement in Future**

- Use Salesforce Connect to create External Objects representing external data models.
- Set up adapters like OData, Cross-Org, or Apex Custom Adapter depending on the external system.
- Configure authentication and permissions to access external data sources.
- Use External Objects in Salesforce UI, reports, Apex, and process automation as if native.
- Leverage features like external object relationships, indirect lookups, and mobile support.
- Manage external data efficiently with no local data storage cost or synchronization lag.

### **Tools and Features**

- External Objects definition and management in Setup.
- Various adapters: OData 2.0/4.0, Cross-Org, Apex Custom Adapter.
- Support for Salesforce reports, dashboards, Flows, Apex, searches, and metadata deployment.
- Robust relationships and real-time data virtualization for external data.

### **7.8 API Limits**

**Use Case:** API limits in Salesforce manage and control the number of API calls your organization can make to ensure optimal system performance and fair resource sharing. These limits affect all REST, SOAP, Bulk, and connected app integrations, helping avoid overloads and ensuring system stability.

#### **Important API Limits:**

- **Daily API Request Limit:** The total number of API calls allowed in a rolling 24-hour period, depending on your edition and licenses.
- **Concurrent API Request Limit:** Maximum number of simultaneous long-running API requests (over 20 seconds).
- **Timeout Limits:** API calls have a timeout limit (e.g., 10 minutes for REST/SOAP).
- **Monthly API Entitlement:** Aggregated daily limits over 30 days for long-term monitoring.
- Limits vary by org type (Developer, Enterprise, Sandbox) and license count.

#### **Monitoring and Management:**

- Monitor API usage on Setup's **System Overview** and **Company Information** pages.
- Use **API Usage Notifications** to alert admins when usage reaches specific thresholds.

- Review API limits and usage via REST API /limits resource.
- Understand error codes like REQUEST\_LIMIT\_EXCEEDED to handle exceeded limits gracefully.
- Plan and purchase additional API calls proactively to meet integration demands.

### **Why Not a Concern Now**

The current project does not require a high volume of API calls exceeding standard limits. Basic monitoring and normal API quota are sufficient for the scope.

### **7.9 OAuth & Authentication**

**Use Case:** OAuth and Authentication are essential for secure integrations, enabling Salesforce to safely access external systems or allowing external apps to securely access Salesforce data. OAuth provides token-based authorization without exposing credentials, supporting standards like OAuth 2.0 for Single Sign-On, Connected Apps, and API access.

#### **Why Not Implemented:**

This project does not require complex custom OAuth authentication schemes or building external apps that access Salesforce directly. Current integrations rely on Named Credentials and external services which internally manage authentication securely, making direct OAuth implementation unnecessary at this stage.

#### **How to Implement in Future**

- Create Connected Apps in Salesforce Setup for external authentication.
- Use OAuth flows (Authorization Code, JWT Bearer, etc.) depending on client type.
- Configure appropriate scopes, permissions, and callbacks.
- Implement OAuth in custom integrations needing delegated user permission or external SSO.
- Monitor and rotate tokens periodically for security.

#### **Tools and Features**

- Connected Apps to register external applications.
- OAuth 2.0 Protocol support for REST/SOAP API access.
- Single Sign-On (SSO) configuration.
- Named Credentials with OAuth for token management.

### **7.10 Remote Site Settings**

**Use Case:** Remote Site Settings allow Salesforce to securely call external web services by authorizing outbound callouts to specific URLs. This prevents unauthorized data exposure and

protects Salesforce from malicious callouts. Any external endpoint Salesforce interacts with must be registered here or via Named Credentials to succeed.

### Implementation Steps:

1. Go to Setup → Search for **Remote Site Settings** in Quick Find → Select **Remote Site Settings**.
2. Click **New Remote Site**.
3. Fill in:
  - **Remote Site Name:** Unique name like "StudentInfoAPI".
  - **Remote Site URL:** Base URL of the external API endpoint.
  - Optionally add a description.
4. Click **Save**.

**Result:** Once configured, Salesforce will allow callouts to the registered remote site, enabling Apex, Flows, or External Services to communicate with external APIs securely without callout errors.

