

## Phase 5 Report – Apex Trigger Implementation

### 1. Objective of Phase 5

The objective of Phase 5 is to automate and enforce business rules for the Contact Training Tracker using Apex triggers and handler classes. This implementation focuses on three key objects within the system:

- Training Session (Training\_Session\_\_c)
- Attendance (Attendance\_\_c)
- Certificate (Certificate\_\_c)

The outcome is improved data accuracy, streamlined automation (like attendance tracking and certification), and enhanced operational efficiency in managing training activities.

**SETUP**  
**Apex Classes**

**Apex Classes** [Help for this Page](#)

Apex Code is an object oriented programming language that allows developers to develop on-demand business applications on the Lightning Platform.

**Percent of Apex Used: 0.02%**  
You are currently using 1,085 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

[Estimate your organization's code coverage](#)

[Compile all classes](#)

**View:** [All](#) [Create New View](#)

Action	Name	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
<a href="#">Edit</a>   <a href="#">Del</a>   <a href="#">Security</a>	AttendanceHandler		64.0	Active	35	Vansh Balli, 9/26/2025, 6:20 AM	<input type="checkbox"/>
<a href="#">Edit</a>   <a href="#">Del</a>   <a href="#">Security</a>	CertificateHandler		64.0	Active	36	Vansh Balli, 9/26/2025, 6:22 AM	<input type="checkbox"/>
<a href="#">Edit</a>   <a href="#">Del</a>   <a href="#">Security</a>	TestContactTrainingTracker		64.0	Active	44	Vansh Balli, 9/26/2025, 6:23 AM	<input type="checkbox"/>
<a href="#">Edit</a>   <a href="#">Del</a>   <a href="#">Security</a>	TrainingSessionHandler		64.0	Active	970	Vansh Balli, 9/26/2025, 6:16 AM	<input type="checkbox"/>

**Dynamic Apex Classes**

Dynamic Apex extends your programming reach by interacting with Lightning Platform components.

**View:** [All](#) [Create New View](#)

Action	Class Name	Namespace Prefix	Api Version	Created By	Last Modified By
--------	------------	------------------	-------------	------------	------------------

### 2. Trigger Design Pattern

A structured approach was applied for trigger development to ensure maintainability and scalability:

- Only one trigger per object to simplify management and reduce errors.
- Business logic is separated into handler classes instead of writing it directly in triggers.
- Centralized handler logic improves reusability, readability, testing, and debugging efficiency.

SETUP

Apex Triggers

Apex Triggers

[Help for this Page](#)

This page allows you to view and modify all the triggers in your organization. To create a new trigger, navigate to the appropriate sObject triggers page.

Percent of Apex Used: 0.03%

You are currently using 2,093 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

Compile all triggers

View: All [Create New View](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Other All

Action	Name ↑	Namespace Prefix	sObject Type	Developer Console		Size Without Comments	Last Modified By	Has Trace Flags
				Api Version	Status			
<a href="#">Edit</a>   <a href="#">Del</a>	AttendanceTrigger		Attendance	64.0	Active	63	Vansh Ball: 9/26/2025, 6:26 AM	<input type="checkbox"/>
<a href="#">Edit</a>   <a href="#">Del</a>	CertificateTrigger		Certificate	64.0	Active	65	Vansh Ball: 9/26/2025, 6:28 AM	<input type="checkbox"/>
<a href="#">Edit</a>   <a href="#">Del</a>	TrainingSessionTrigger		Training_Session	64.0	Active	880	Vansh Ball: 9/26/2025, 6:26 AM	<input type="checkbox"/>

### 3. Implementation Steps

- Defined and created trigger files for each main object with contexts such as before insert, before update, after insert, and after delete.
- Developed handler classes to manage business rules, including:
  - Validation (e.g., preventing duplicate attendance records).
  - Pre-deletion logic (e.g., blocking deletion of sessions if certificates exist).
  - Post-insertion automation (e.g., auto-creating attendance records when trainees enroll).
  - Post-update automation (e.g., issuing certificates once a session is marked “Completed”).
- Created and organized metadata files for each Apex class for smooth deployment and Salesforce compliance.

### 4. Deployment Steps

- Organized triggers and handler classes within the Salesforce project structure using VS Code and Salesforce CLI.
- Deployed to Salesforce and verified trigger activation in Object Manager under the respective objects.
- Performed testing with sample data to validate that rules and automation execute correctly.

### 5. Expansion for Other Objects

The same design pattern can be extended to additional objects in the future, such as:

- Trainer (User) – automation for trainer notifications.
- Feedback (Feedback\_\_c) – auto-creation of follow-up tasks when negative feedback is submitted.

This ensures consistent automation, scalability, and easy addition of future business logic.

## **6. Benefits of This Approach**

- Clean and concise code with separation of concerns.
- Handler methods are reusable, reducing future development time.
- Easier and more efficient testing and debugging.
- Flexible system architecture to accommodate evolving training requirements.