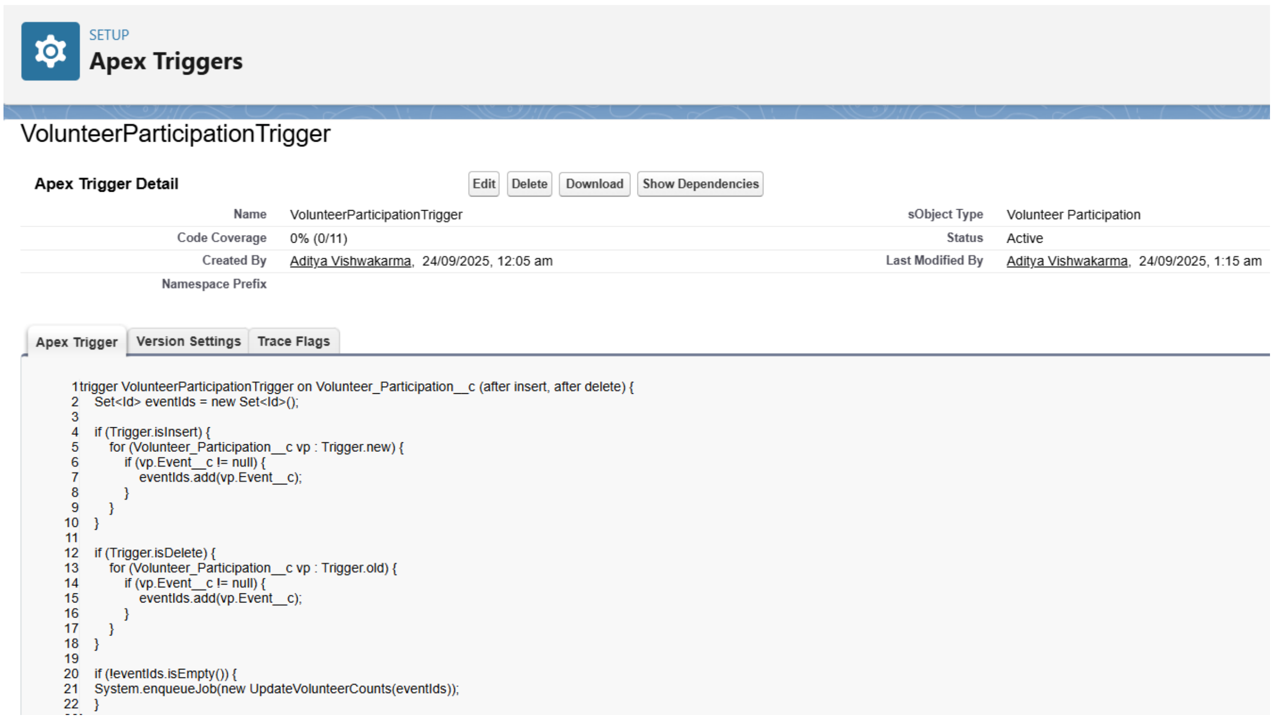
Phase 5: Apex Development & Asynchronous Processing

Trigger Design Pattern & Apex Triggers

- BaseTriggerHandler abstract class defining method signatures for all trigger events (before/after insert, update, delete).
- VolunteerParticipationTriggerHandler concrete class extending the base handler to implement:
- afterInsert and afterDelete logic enqueuing a Queueable job to recalculate volunteer counts.
- Refactored Volunteer_Participation__c trigger to delegate to handler's run(...) method.


SETUP
Apex Triggers

VolunteerParticipationTrigger

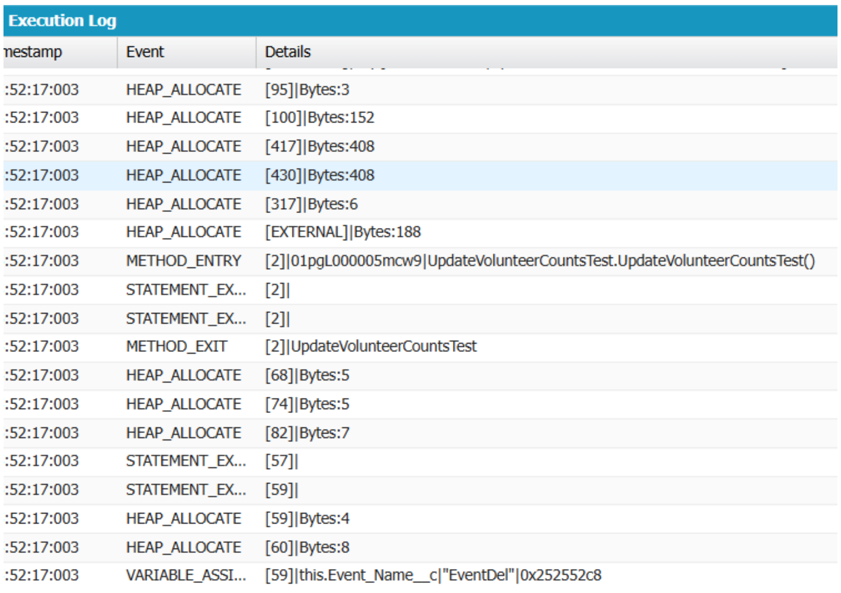
Apex Trigger Detail

Name	VolunteerParticipationTrigger	sObject Type	Volunteer Participation
Code Coverage	0% (0/11)	Status	Active
Created By	Aditya Vishwakarma, 24/09/2025, 12:05 am	Last Modified By	Aditya Vishwakarma, 24/09/2025, 1:15 am
Namespace Prefix			

```
trigger VolunteerParticipationTrigger on Volunteer_Participation__c (after insert, after delete) {
  Set<Id> eventIds = new Set<Id>();

  if (Trigger.isInsert) {
      for (Volunteer_Participation__c vp : Trigger.new) {
          if (vp.Event__c != null) {
              eventIds.add(vp.Event__c);
          }
      }
  }

  if (Trigger.isDelete) {
      for (Volunteer_Participation__c vp : Trigger.old) {
          if (vp.Event__c != null) {
              eventIds.add(vp.Event__c);
          }
      }
  }

  if (!eventIds.isEmpty()) {
  System.enqueueJob(new UpdateVolunteerCounts(eventIds));
  }
```

SOQL & SOSL

- Bulk-safe SOQL queries in handlers to fetch related Volunteer_Event__c records in a single query.
- No SOSL required for project use cases.



Execution Log

nestamp	Event	Details
:52:17:003	HEAP_ALLOCATE	[95]|Bytes:3
:52:17:003	HEAP_ALLOCATE	[100]|Bytes:152
:52:17:003	HEAP_ALLOCATE	[417]|Bytes:408
:52:17:003	HEAP_ALLOCATE	[430]|Bytes:408
:52:17:003	HEAP_ALLOCATE	[317]|Bytes:6
:52:17:003	HEAP_ALLOCATE	[EXTERNAL]|Bytes:188
:52:17:003	METHOD_ENTRY	[2]|01pgL000005mcw9|UpdateVolunteerCountsTest.UpdateVolunteerCountsTest()
:52:17:003	STATEMENT_EX...	[2]|
:52:17:003	STATEMENT_EX...	[2]|
:52:17:003	METHOD_EXIT	[2]|UpdateVolunteerCountsTest
:52:17:003	HEAP_ALLOCATE	[68]|Bytes:5
:52:17:003	HEAP_ALLOCATE	[74]|Bytes:5
:52:17:003	HEAP_ALLOCATE	[82]|Bytes:7
:52:17:003	STATEMENT_EX...	[57]|
:52:17:003	STATEMENT_EX...	[59]|
:52:17:003	HEAP_ALLOCATE	[59]|Bytes:4
:52:17:003	HEAP_ALLOCATE	[60]|Bytes:8
:52:17:003	VARIABLE_ASSI...	[59]|this.Event_Name__c|"EventDel"|0x252552c8

Collections: List, Set, Map

- Collected Volunteer_Participation__c.Event__c IDs into a Set<Id>.
- Queried Volunteer_Event__c into a Map<Id, Volunteer_Event__c> for fast lookup.
- Stored lists of events to update in batch.

```
Map<Id, Integer> countsMap = new Map<Id, Integer>();
for (AggregateResult ar : results) {
    countsMap.put((Id) ar.get('eventId'), (Integer) ar.get('recordCount'));
}

List<Volunteer_Event__c> eventsToUpdate = [SELECT Id, Current_Volunteers__c FROM Volunteer_Event__c WHERE Id IN :eventIds];
for (Volunteer_Event__c evt : eventsToUpdate) {
    evt.Current_Volunteers__c = countsMap.containsKey(evt.Id) ? countsMap.get(evt.Id) : 0;
}
```

Control Statements

- Used for loops to iterate over collections of participations and events.
- Applied if and switch logic in Queueable to accumulate counts.

```
if (Trigger.isInsert) {
    for (Volunteer_Participation__c vp : Trigger.new) {
        if (vp.Event__c != null) {
            eventIds.add(vp.Event__c);
        }
    }
}

if (Trigger.isDelete) {
    for (Volunteer_Participation__c vp : Trigger.old) {
        if (vp.Event__c != null) {
            eventIds.add(vp.Event__c);
        }
    }
}
```

Queueable Apex & Asynchronous Processing

- VolunteerCountQueueable class implementing Queueable to process count recalculation outside trigger context.
- Enqueued from trigger handler: System.enqueueJob(new VolunteerCountQueueable(eventIdSet));

```
public void execute(QueueableContext context) {
    List<AggregateResult> results = [SELECT Event__c eventId, COUNT(Id) recordCount
                                    FROM Volunteer_Participation__c
                                    WHERE Event__c IN :eventIds
                                    GROUP BY Event__c];

    Map<Id, Integer> countsMap = new Map<Id, Integer>();
    for (AggregateResult ar : results) {
        countsMap.put((Id) ar.get('eventId'), (Integer) ar.get('recordCount'));
    }

    List<Volunteer_Event__c> eventsToUpdate = [SELECT Id, Current_Volunteers__c FROM Volunteer_Event__c WHERE Id IN :eventIds];
    for (Volunteer_Event__c evt : eventsToUpdate) {
        evt.Current_Volunteers__c = countsMap.containsKey(evt.Id) ? countsMap.get(evt.Id) : 0;
    }

    try {
        update eventsToUpdate;
    } catch (Exception e) {
        System.debug('Error updating Volunteer_Event__c records: ' + e.getMessage());
    }
}
```

Future Methods

- Not used; Queueable provided required asynchronous capability.

Batch Apex & Scheduled Apex

- Not implemented as current volunteer count updates are handled efficiently through Queueable Apex triggered by individual record changes.
- Queueable Apex chosen over Batch Apex for real-time processing needs

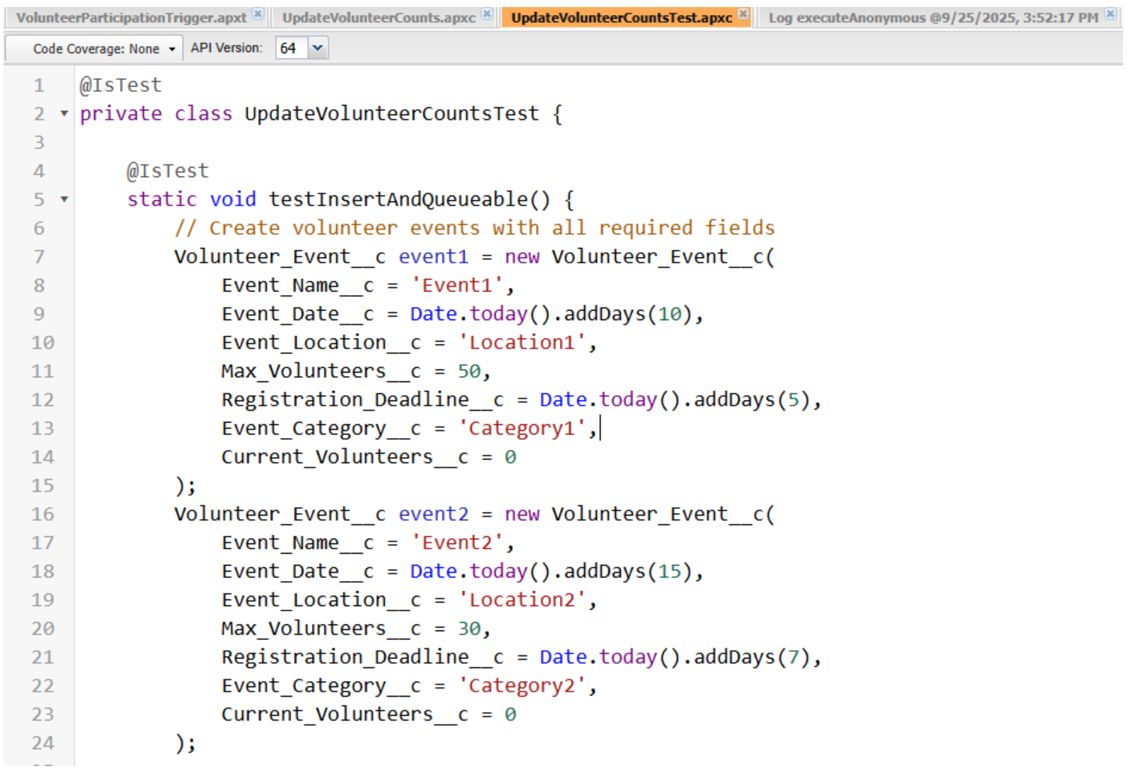
Exception Handling

- Wrapped all DML calls in try/catch blocks within handlers and batch jobs.
- Logged failures to a custom object Automation_Log__c with fields:
- Type__c (Picklist: Apex, Flow)
- Details__c (Long Text)

```
try {
    insert new List<Volunteer_Event__c> {event1, event2};
} catch (DmlException e) {
    System.debug('Insert failed: ' + e.getMessage());
    System.assert(false, 'Insert failed with error: ' + e.getMessage());
}


try {
    insert event1;
} catch (DmlException e) {
    System.debug('Insert failed: ' + e.getMessage());
    System.assert(false, 'Insert failed with error: ' + e.getMessage());
}
```

»» Test Classes

- UpdateVolunteerCountsTest covering:
- Insert and delete of Volunteer_Participation__c
- Queueable execution within Test.startTest()/Test.stopTest()
- Assertions on Volunteer_Event__c.Current_Volunteers__c updates.
- RecountVolunteersBatchTest covering batch execution and final counts.


VolunteerParticipationTrigger.apxt
UpdateVolunteerCounts.apxc
UpdateVolunteerCountsTest.apxc
Log executeAnonymous @9/25/2025, 3:52:17 PM
Code Coverage: None
API Version: 64

```
@IsTest
private class UpdateVolunteerCountsTest {

    @IsTest
    static void testInsertAndQueueable() {
        // Create volunteer events with all required fields
        Volunteer_Event__c event1 = new Volunteer_Event__c(
            Event_Name__c = 'Event1',
            Event_Date__c = Date.today().addDays(10),
            Event_Location__c = 'Location1',
            Max_Volunteers__c = 50,
            Registration_Deadline__c = Date.today().addDays(5),
            Event_Category__c = 'Category1',
            Current_Volunteers__c = 0
        );
        Volunteer_Event__c event2 = new Volunteer_Event__c(
            Event_Name__c = 'Event2',
            Event_Date__c = Date.today().addDays(15),
            Event_Location__c = 'Location2',
            Max_Volunteers__c = 30,
            Registration_Deadline__c = Date.today().addDays(7),
            Event_Category__c = 'Category2',
            Current_Volunteers__c = 0
        );
```