

# Phase 5 Report: Apex Programming (Developer)

## 1. Introduction

This phase focuses on implementing **custom business logic** using **Apex Classes and Triggers**.

While Flows handle standard automation, **Apex allows for advanced logic**, such as:

- Validating attendee registration rules
  - Handling bulk operations efficiently
  - Sending custom notifications or updates
- 

## 2. Purpose of Apex Programming

The main purpose of Apex in this project:

- Automatically check **event capacity** when new attendees register
  - Prevent **overbooking** beyond the defined Event Capacity
  - Maintain data integrity and handle complex business logic that Flow cannot handle
- 

## 3. Classes & Triggers Development

**Apex Trigger Example:**

- Object: **Attendee**
- Trigger Event: **before insert**

**Logic:**

- Count existing attendees for the event
  - Compare with Event Capacity
  - If capacity exceeded, throw error to prevent registration
- 

**Apex Class Example (Helper Class):**

- Class Name: `EventCapacityChecker`
- Method: `checkCapacity(Id eventId)`
- Returns boolean / throws exception if capacity exceeded

## 4. Sample Apex Code

```
trigger AttendeeTrigger on Attendee__c (before insert) {
    Set<Id> eventIds = new Set<Id>();
    for(Attendee__c a : Trigger.new){
        if(a.Event__c != null){
            eventIds.add(a.Event__c);
        }
    }
    Map<Id, Event__c> events = new Map<Id, Event__c>(
        [SELECT Id, Capacity__c,
        (SELECT Id FROM Attendees__r)
        FROM Event__c WHERE Id IN :eventIds]
    );

    for(Attendee__c a : Trigger.new){
        Event__c ev = events.get(a.Event__c);
        if(ev.Attendees__r.size() >= ev.Capacity__c){
            a.addError('Registration failed: Event capacity full.');
        }
    }
}
```

## 5. Testing & Debugging

- Used **Anonymous Apex** or **Test Classes** to simulate registrations
- Verified that **overbooking is blocked**
- Checked **system debug logs** to confirm logic execution

The screenshot shows the Salesforce Setup interface. The left sidebar contains a navigation menu with categories like Email, Custom Code, Environments, and Jobs. Under Custom Code, 'Apex Classes' is selected. The main content area is titled 'Apex Classes' and includes a description of Apex Code. A green status box indicates 'Percent of Apex Used: 0%'. Below this, there are links for 'Estimate your organization's code coverage' and 'Compile all classes'. A table with columns for Name, Namespace Prefix, Api Version, Status, Size Without Comments, Last Modified By, and Has Trace Flags is shown, with a note 'No records to display.' Below this table, there is a section for 'Dynamic Apex Classes' with a similar table structure and a 'No records to display.' message. The bottom of the page shows a URL: 'https://ss6-dev-ed.develop.lightning.force.com/lightning/setup/ApexClasses/no...'.

---

## 6. Conclusion

- Apex programming ensures **advanced validations** that Flows cannot handle alone
- Maintains **data integrity** and enforces event capacity rules
- Lays foundation for further developer-level customization in the project