Phase 5: Apex Programming (Developer)

1.Apex Classes & Objects

1. PetUtility.cls: To check number of available pets to adopt

PetUtility.apxc | TriggerHandler.apxc | PetTriggerHandler.apxc * | AdoptionRequestTriggerHandler.apxc | AdoptionRequestTrigger.apxt * | PetTrigger.apxt * | AdoptionNotificationSer

Code Coverage: None | API Version: 64

```
public with sharing class PetUtility {
    // Get all available pets
    public static List<Pet__c> getAvailablePets(){
        return [SELECT Id, Name, Breed__c, Availability__c FROM Pet__c WHERE Availability__c = 'available'];
    }

    // Mark a pet as adopted
    public static void markAsAdopted(Id petId){
        Pet__c pet = [SELECT Id, Availability__c FROM Pet__c WHERE Id = :petId];
        pet.Availability__c = 'adopted';
        update pet;
    }
}
```

2. AdopterService.cls : To Count how many pets adopter has adopted

AdopterService.apxc

Code Coverage: None | API Version: 64

```
public with sharing class AdopterService {
    // Count how many pets adopter has adopted
    public static Integer getTotalPetsAdopted(Id adopterId){
        return [SELECT COUNT() FROM Adoption_Request__c WHERE Adopter__c = :adopterId AND Status__c = 'Approved'];
    }
}
```

3. AdoptionService.cls : This class created to get discount for potential adopters

AdoptionService.apxc

Code Coverage: None | API Version: 64

```
public with sharing class AdoptionService {
    // Calculate discounted fee (10% off for returning adopters)
    public static Decimal calculateDiscountedFee(Decimal fee, Boolean isReturningAdopter){
        if(isReturningAdopter){
            return fee * 0.9;
        }
        return fee;
    }
}
```

4. VolunteerService.cls : To check Number of hours staff worked

DService.apxc | VolunteerService.apxc

Code Coverage: None | API Version: 64

```
public with sharing class VolunteerService {
    // Get total volunteer hours logged for one volunteer
    public static Integer getVolunteerHours(Id volunteerId){
        AggregateResult result = [
            SELECT SUM(Hours_Logged__c) totalHours
            FROM Volunteer_Shift__c
            WHERE Volunteer_c__c = :volunteerId
        ];

        // Return 0 if no hours found
        if(result != null && result.get('totalHours') != null){
            return (Integer)result.get('totalHours');
        }
        return 0;
    }
}
```

2.Apex Triggers (before/after insert/update/delete)

1. PetTrigger.trigger

Purpose: To handle automation and business logic for Pet records

File | Edit | Debug | Test | Workspace | Help

PetTrigger.apxt

Code Coverage: None | API Version: 64

```
trigger PetTrigger on Pet__c (
    before insert, before update, before delete,
    after insert, after update, after delete, after undelete
) {
    PetTriggerHandler handler = new PetTriggerHandler();

    if(Trigger.isBefore && Trigger.isInsert) {
        handler.beforeInsert(Trigger.new);
    } else if(Trigger.isBefore && Trigger.isUpdate) {
        handler.beforeUpdate(Trigger.new, Trigger.old, Trigger.newMap, Trigger.oldMap);
    } else if(Trigger.isAfter && Trigger.isInsert) {
        handler.afterInsert(Trigger.new, Trigger.newMap);
    } else if(Trigger.isAfter && Trigger.isUpdate) {
        handler.afterUpdate(Trigger.new, Trigger.old, Trigger.newMap, Trigger.oldMap);
    }
}
```

2. AdoptionRequestTrigger.trigger

Purpose: To manage the complete adoption request lifecycle

File Edit Debug Test Workspace Help

AdoptionRequestTrigger.apxt

Code Coverage: None API Version: 64

```
trigger AdoptionRequestTrigger on Adoption_Request__c (
    before insert, before update, before delete,
    after insert, after update, after delete, after undelete
) {
    // Instantiate the trigger handler
    AdoptionRequestTriggerHandler handler = new AdoptionRequestTriggerHandler();

    // Before Insert Context
    if(Trigger.isBefore && Trigger.isInsert) {
        handler.beforeInsert(Trigger.new);
    }

    // Before Update Context
    else if(Trigger.isBefore && Trigger.isUpdate) {
        handler.beforeUpdate(Trigger.new, Trigger.old, Trigger.newMap, Trigger.oldMap);
    }

    // Before Delete Context
    else if(Trigger.isBefore && Trigger.isDelete) {
```

Logs Tests Checkpoints Query Editor View State Progress Problems

SOQL & SOSL - Implement Search Functionality

File Edit Debug Test Workspace Help

PetSearchService.apxc *

Code Coverage: None API Version: 64

```
public with sharing class PetSearchService {

    // SOQL with complex queries
    public static Map<String, List<Pet__c>> getPetsByCategory() {
        Map<String, List<Pet__c>> petsBySpecies = new Map<String, List<Pet__c>>();

        // Bulk SOQL query
        List<Pet__c> allPets = [SELECT Id, Name, Species__c, Breed__c, Age__c, Availability__c
                                FROM Pet__c
                                WHERE Availability__c = 'available'
                                ORDER BY Species__c, Name];

        // Using Map to categorize
        for(Pet__c pet : allPets) {
            if(!petsBySpecies.containsKey(pet.Species__c)) {
                petsBySpecies.put(pet.Species__c, new List<Pet__c>());
            }
            petsBySpecies.get(pet.Species__c).add(pet);
        }

        return petsBySpecies;
```

What it does:

- **SOQL**: Query Salesforce database to get pet records
- **WHERE clause**: Filter only available pets
- **IN clause**: Get multiple pets at once (bulk query)

Collections (List, Set, Map)

```
public class AdoptionRequestTriggerHandler extends TriggerHandler {

    // Before Insert
    public override void beforeInsert(List<SObject> newList) {
        List<Adoption_Request__c> newRequests = (List<Adoption_Request__c>) newList;
        validateAdoptionRequests(newRequests);
        setDefaultValues(newRequests);
    }

    // Before Update
    public override void beforeUpdate(List<SObject> newList, List<SObject> oldList, Map<Id, SObject> newMap, Map<Id, SObject> oldMap) {
        List<Adoption_Request__c> newRequests = (List<Adoption_Request__c>) newList;
        Map<Id, Adoption_Request__c> oldRequests = (Map<Id, Adoption_Request__c>) oldMap;
        validateStatusChanges(newRequests, oldRequests);
    }

    // After Insert
    public override void afterInsert(List<SObject> newList, Map<Id, SObject> newMap) {
        List<Adoption_Request__c> newRequests = (List<Adoption_Request__c>) newList;
        handleNewAdoptionRequests(newRequests);
    }
```

- **List**: Maintains order of adoption requests
- **Set**: Ensures no duplicate pet IDs
- **Map**: Quick lookup of pets by ID (instead of looping through lists)

Control Statements

```
public with sharing class AdoptionFeeCalculator {

    public static Decimal calculateAdoptionFee(Pet__c pet, Boolean isReturningAdopter) {
        Decimal baseFee = 0;

        // Switch statement for species-based pricing
        switch on pet.Species__c {
            when 'Dog' {
                baseFee = 250;
                // If-else for age-based discount
                if(pet.Age__c > 7) {
                    baseFee = 150; // Senior pet discount
                }
            }
            when 'Cat' {
                baseFee = 150;
                if(pet.Age__c > 7) {
                    baseFee = 100;
                }
            }
            when 'Bird' {
```

What it does:

- **Conditional logic:** Validate business rules
- **Looping:** Process multiple records efficiently

Batch Apex (Data Maintenance)

```
public class MonthlyAdoptionReportBatch implements Database.Batchable<SObject>, Database.Stateful {

    public Integer totalAdoptions = 0;
    public Integer totalDonations = 0;
    public Decimal totalRevenue = 0;

    public Database.QueryLocator start(Database.BatchableContext bc) {
        // Get adoption requests from last month
        Date lastMonth = Date.today().addMonths(-1);
        return Database.getQueryLocator([
            SELECT Id, Adoption_Fee__c, Request_Date__c, Status__c
            FROM Adoption_Request__c
            WHERE Request_Date__c = LAST_MONTH
            AND Status__c = 'Approved'
        ]);
    }

    public void execute(Database.BatchableContext bc, List<Adoption_Request__c> scope) {
        List<Adoption_Request__c> successfulAdoptions = new List<Adoption_Request__c>();

        for(Adoption_Request__c request : scope) {
```

Queueable Apex

```
public class DonationReceiptQueueable implements Queueable {
    private List<Id> donationIds;

    public DonationReceiptQueueable(List<Id> donationIds) {
        this.donationIds = donationIds;
    }

    public void execute(QueueableContext context) {
        // Process donations and send receipts asynchronously
        List<Donation__c> donations = [
            SELECT Id, Donation_Amount__c, Adopter_c__r.Email__c, Donation_Date__c
            FROM Donation__c
            WHERE Id IN :donationIds AND Status__c = 'approved'
        ];

        sendDonationReceipts(donations);

        // Chain next job if needed
        if(!donations.isEmpty()) {
            System.enqueueJob(new ThankYouEmailQueueable(donationIds));
        }
```

- **Donation receipt chains:** Send receipt → Thank you email → Follow-up
- **Adoption process chains:** Approval → Notification → Welcome kit
- **Volunteer shift reminders**

Scheduled Apex

File Edit Debug Test Workspace Help < >

MonthlyAdoptionReportBatch.apxc * | DonationReceiptQueueable.apxc * | PetUtilityTest.apxc * | WeeklyAdoptionReportScheduler.apxc *

Code Coverage: None API Version: 64

```apex
public class WeeklyAdoptionReportScheduler implements Schedulable {
    public void execute(SchedulableContext sc) {
        // Run batch job for weekly reports
        WeeklyAdoptionReportBatch batch = new WeeklyAdoptionReportBatch();
        Database.executeBatch(batch, 50);
    }
}
```

Scheduled Apex Needed For:

- **Daily shelter cleanup tasks**
- **Weekly adoption statistics**
- **Monthly volunteer hour calculations**

Exception Handling

PetSearchService.apxc * | PetUtility.apxc *

Code Coverage: None API Version: 64

```apex
    public static void markAsAdopted(Id petId){
        try {
            List<Pet__c> pets = [SELECT Id, Availability__c FROM Pet__c
                                 WHERE Id = :petId WITH SECURITY_ENFORCED];

            if(pets.isEmpty()) {
                throw new IllegalArgumentException('Pet not found with ID: ' + petId);
            }

            Pet__c pet = pets[0];
            pet.Availability__c = 'adopted';
            update pet;

        } catch (DmlException e) {
            System.debug('DML Error: ' + e.getMessage());
            throw e;
        } catch (Exception e) {
            System.debug('Unexpected error: ' + e.getMessage());
            throw e;
        }
    }
}
```

Exception Handling Needed For:

- **Prevent system crashes** when data is missing
- **Graceful error messages** for users
- **Error logging** for debugging

Test Classes

File Edit Debug Test Workspace Help

MonthlyAdoptionReportBatch.apxc * | DonationReceiptQueueable.apxc * | PetUtilityTest.apxc *

Code Coverage: None | API Version: 64

```
@isTest
private class PetUtilityTest {
    @testSetup
    static void setupTestData() {
        // Create test pets
        List<Pet__c> testPets = new List<Pet__c>{
            new Pet__c(Name = 'Test Dog', Species__c = 'Dog', Availability__c = 'available'),
            new Pet__c(Name = 'Test Cat', Species__c = 'Cat', Availability__c = 'adopted')
        };
        insert testPets;
    }

    @isTest
    static void testGetAvailablePets() {
        Test.startTest();
        List<Pet__c> availablePets = PetUtility.getAvailablePets();
        Test.stopTest();

        System.assertEquals(1, availablePets.size(), 'Should return only available pets');
        System.assertEquals('Test Dog', availablePets[0].Name);
    }
```

Deployment to production

- **Code quality assurance**
- **Preventing bugs in live system**