Project-Title – “Government Schemes Management System”

Phase 5 :Apex Programming

Classes & objects

1. Helper Class for Business Logic

Purpose:

- Encapsulates the main business logic separately from triggers to maintain clean code.
- Automatically updates the status of applications based on the End_Date__c.

How it works:

- Checks each application in the list.
- If the end date is past today, sets status to Expired; otherwise, sets it to Active.

Benefits:

- Promotes code reusability (can be called from triggers, batch jobs, etc.).
- Easier to maintain and update business rules in one place

File Edit Debug Test Workspace Help < >

CustomNotificationType@1:51 PM | SchemeApplicationHelper.apxc | SchemeApplicationTrigger.apxt | Log executeAnonymous @9/25/2025, 2:42:13 PM

Code Coverage: None | API Version: 64

```
public class SchemeApplicationHelper {
    public static void setSubmissionStatus(List<Scheme_Application__c> apps) {
        Set<Id> schemeIds = new Set<Id>();
        for (Scheme_Application__c app : apps) {
            if (app.Scheme__c != null) schemeIds.add(app.Scheme__c);
        }

        Map<Id, Date> schemeEndDates = new Map<Id, Date>();
        if (!schemeIds.isEmpty()) {
            for (Scheme__c s : [
                SELECT Id, End_Date__c
                FROM Scheme__c
                WHERE Id IN :schemeIds
            ]) {
                schemeEndDates.put(s.Id, s.End_Date__c);
            }
        }

        for (Scheme_Application__c app : apps) {
            Date schemeEnd = schemeEndDates.get(app.Scheme__c);

            // Default End Date if missing
            if (schemeEnd == null) schemeEnd = Date.today().addDays(30);

            if (app.Submission_Date__c != null) {
                if (app.Submission_Date__c > schemeEnd) {
                    app.Status__c = 'Rejected'; // or 'Application Expired'
                } else {
                    app.Status__c = 'Submitted';
                }
            } else {
                app.Status__c = 'Submitted';
            }
        }
    }
}
```

Anonymous code Window

```
// Create test scheme with required fields
Scheme__c scheme = new Scheme__c(
    Name = 'Scholarship Scheme',
    Start_Date__c = Date.today().addDays(-10),
    End_Date__c = Date.today().addDays(-1), // yesterday
    Active__c = true,
    Max_Amount__c = 10000 // REQUIRED field
);
insert scheme;

// Create a citizen contact
Contact c = new Contact(LastName='Applicant One', Email='test@example.com');
insert c;

// Create an application with submission after deadline
Scheme_Application__c lateApp = new Scheme_Application__c(
    Citizen__c = c.Id,
    Scheme__c = scheme.Id,
    Submission_Date__c = Date.today(), // today > scheme end date
    Requested_Amount__c = 5000,
    Status__c = 'Submitted' // just to avoid blank required picklist issues
);
insert lateApp;

// Query back to check status
Scheme_Application__c check = [
    SELECT Id, Status__c
    FROM Scheme_Application__c
    WHERE Id = :lateApp.Id
];
System.debug('Application Status = ' + check.Status__c);
```

Output:

SELECT Id, Name, Status__c, Submission_Date__c, Citizen__r.Name FROM Scheme_Application__c WHERE Status__c = 'Submitted'

Query Results - Total Rows: 3

Id	Name	Status__c	Submission_Date__c	Citizen__r.Name
a01dM000033rM7YQAU	HealthAppTest1	Submitted	2025-09-23	Citizen One
a01dM000033zQAbQAM	Education Application - 001	Submitted	2025-09-24	Citizen One
a01dM000034CezeQAC	a01dM000034Ceze	Submitted	2025-09-25	Applicant 8

```
SELECT Id, Name, Status__c, Submission_Date__c, Citizen__r.Name
FROM Scheme_Application__c
WHERE Status__c = 'Submitted'
```

2. Trigger Design Pattern

Purpose:

- Handles record-level automation for Scheme_Application__c during insert or update.
- Uses a trigger handler to separate logic from the trigger itself.

How it works:

- beforeInsert and beforeUpdate methods set default status and check end dates.
- Reuses logic for both insert and update events to avoid duplication.

Benefits:

- Ensures data consistency automatically whenever a record is created or updated.
- Follows best practices by using a handler class instead of putting all logic in the trigger.

```
public class SchemeApplicationHandler {
    private static Boolean isRunning = false;

    public static void beforeInsert(List<Scheme_Application__c> newApps) {
        if(isRunning) return;
        isRunning = true;

        // Default Citizen_Email if blank
        for(Scheme_Application__c app : newApps){
            if(String.isBlank(app.Citizen_Email__c) && app.Citizen__c != null){
                Contact c = [SELECT Email FROM Contact WHERE Id = :app.Citizen__c LIMIT 1];
                app.Citizen_Email__c = c.Email;
            }
        }

        // Set submission status
        SchemeApplicationHelper.setSubmissionStatus(newApps);

        // Send notification to officers
        sendOfficerNotification(newApps);

        isRunning = false;
    }

    public static void beforeUpdate(List<Scheme_Application__c> updatedApps, Map<Id, Scheme_Application__c> oldMap) {
        if(isRunning) return;
        isRunning = true;

        // Prevent edits to rejected/expired applications
        for(Scheme_Application__c app : updatedApps){
            Scheme_Application__c oldApp = oldMap.get(app.Id);
            if(oldApp.Status__c == 'Rejected' || oldApp.Status__c == 'Application Expired'){
                app.addError('Cannot modify an application that is already Rejected or Expired.');
            }
        }

        // Update status if submission date changed
        SchemeApplicationHelper.setSubmissionStatus(updatedApps);

        isRunning = false;
    }

    private static void sendOfficerNotification(List<Scheme_Application__c> apps){
        List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();
        List<User> officers = [SELECT Id, Email FROM User WHERE Profile.Name = 'Scheme Officer' AND IsActive = true];

        for(User officer : officers){
            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
            email.setToAddresses(new List<String>{officer.Email});
            email.setSubject('New Scheme Application Submitted');
            email.setPlainTextBody('A new application has been submitted. Please review it in the portal.');
            emails.add(email);
        }

        if(!emails.isEmpty()){
            Messaging.sendEmail(emails);
        }
```

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

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

ionType@5:35 PM | SchemeApplicationHelper.apxc | NotifyOfficerQueueable.apxc | ExpiredApplicationBatch.apxc | Log executeAnonymous @9/25/2025, 5:53:02 PM | Sche

Code Coverage: None ▾ API Version: 64

```
trigger SchemeApplicationTrigger on Scheme_Application__c (before insert, before update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert) {
            SchemeApplicationHandler.beforeInsert(Trigger.new);
        }
        if (Trigger.isUpdate) {
            SchemeApplicationHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
        }
    }
}
```

Output:

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

SchemeApplicationHelper.apxc | SchemeStatusQueueable.apxc | **Log executeAnonymous @9/25/2025, 5:20:10 PM**

Execution Log

Timestamp	Event	Details
17:20:10:001	USER_INFO	[EXTERNAL]|005dM00000OKT3F|achchitha_govtschemes@rvrjc.com|(GMT+05:30) India Standard Time
17:20:10:001	EXECUTION_ST...	
17:20:10:001	CODE_UNIT_ST...	[EXTERNAL]|execute_anonymous_apex
17:20:10:001	VARIABLE_SCO...	[1]|appIds|List<Id>|true|false
17:20:10:001	HEAP_ALLOCATE	[95]|Bytes:3
17:20:10:001	HEAP_ALLOCATE	[100]|Bytes:152
17:20:10:001	HEAP_ALLOCATE	[417]|Bytes:408
17:20:10:001	HEAP_ALLOCATE	[430]|Bytes:408
17:20:10:001	HEAP_ALLOCATE	[317]|Bytes:6
17:20:10:002	HEAP_ALLOCATE	[EXTERNAL]|Bytes:3
17:20:10:002	STATEMENT_EX...	[1]|
17:20:10:002	STATEMENT_EX...	[1]|
17:20:10:002	HEAP_ALLOCATE	[1]|Bytes:4
17:20:10:002	HEAP_ALLOCATE	[1]|Bytes:15
17:20:10:002	HEAP_ALLOCATE	[1]|Bytes:4
17:20:10:002	HEAP_ALLOCATE	[1]|Bytes:15
17:20:10:002	HEAP_ALLOCATE	[1]|Bytes:4
17:20:10:002	HEAP_ALLOCATE	[EXTERNAL]|Bytes:12
17:20:10:002	VARIABLE_ASSI...	[1]|appIds|["a01dM0000345wKVQAY","a01dM000034Ca2IQAS"]|0x74317544
17:20:10:002	STATEMENT_EX...	[2]|
17:20:10:013	HEAP_ALLOCATE	[2]|Bytes:3
17:20:10:013	HEAP_ALLOCATE	[2]|Bytes:2
17:20:10:013	METHOD_ENTRY	[1]|01pdM00000M1wEz|SchemeStatusQueueable.SchemeStatusQueueable()

3. SOQL Query Example

Purpose:

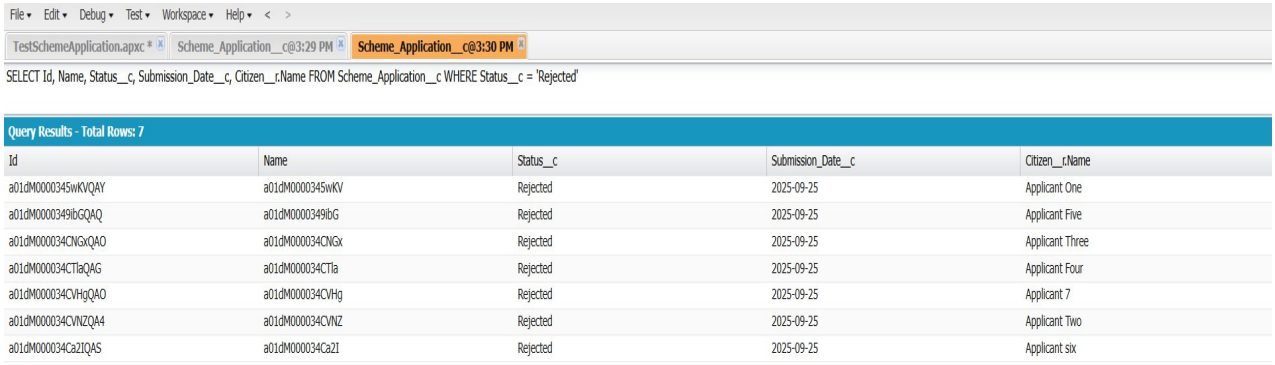
- Retrieve records from Salesforce based on specific criteria (e.g., active applications).

Example:

- Queries Scheme_Application__c records where Status__c = 'Active'.
- Can be used in reports, dashboards, or automation logic.

Benefits:

- Efficient way to filter and process records.
- Works well with Apex logic for further processing.



Id	Name	Status__c	Submission_Date__c	Citizen__r.Name
a01dM0000345wKVQAY	a01dM0000345wKV	Rejected	2025-09-25	Applicant One
a01dM0000349IbGQAQ	a01dM0000349IbG	Rejected	2025-09-25	Applicant Five
a01dM000034CNGxQAO	a01dM000034CNGx	Rejected	2025-09-25	Applicant Three
a01dM000034CTlaQAG	a01dM000034CTla	Rejected	2025-09-25	Applicant Four
a01dM000034CVHgQAO	a01dM000034CVHg	Rejected	2025-09-25	Applicant 7
a01dM000034CVNZQA4	a01dM000034CVNZ	Rejected	2025-09-25	Applicant Two
a01dM000034Ca2IQAS	a01dM000034Ca2I	Rejected	2025-09-25	Applicant six

4. SOSL Example

Purpose:

- Search across multiple objects simultaneously (e.g., Scheme_Application__c + Contact).

How it works:

- Searches for records containing a keyword in any field.
- Returns results in lists by object type.

Benefits:

- Useful for global search functionality in apps or portals.
- Reduces multiple queries by combining search in one SOSL call.

File Edit Debug Test Workspace Help

CustomNotificationType@5:35 PM | SchemeApplicationHelper.apxc | NotifyOfficerQueueable.apxc | ExpiredApplicationBatch.apxc | Log executeAnon

Code Coverage: None API Version: 64

```
List<List<SObject>> searchResults = [
    FIND 'Test*' IN ALL FIELDS
    RETURNING Scheme_Application__c(Name, Status__c), Contact(Name, Email)
];
List<Scheme_Application__c> apps = (List<Scheme_Application__c>) searchResults[0];
List<Contact> contacts = (List<Contact>) searchResults[1];

for(Scheme_Application__c app : apps){
    System.debug('Found App: ' + app.Name + ' Status: ' + app.Status__c);
}
```

5. Collections

Purpose:

- Efficiently manage sets of records in Apex code.

Types:

- List: Ordered collection of records (e.g., query results).

- Set: Stores unique values (avoids duplicates).
- Map: Links keys to values (e.g., Applicant ID → Application record).

Benefits:

- Improves performance and organization of data.
- Enables easy iteration, searching, and mapping of records.

6. Control Statements

Purpose:

- Implements logic decisions and iterations in Apex.

Examples in project:

- if-else: Checks end dates to set Status__c.
- for loops: Iterates through records from queries or user input.

Benefits:

- Ensures business rules are applied correctly.
- Supports automated decision-making in Apex.

7.Asyncronous Processing

Use of Batch Apex and Queueable Apex in Government Schemes Application Project

In the Smart Subscription Tracker project for managing government schemes, citizens can submit applications for various schemes, and officers need to verify them. Each application has an end date (End_Date__c), after which the application should automatically be marked as Expired. Handling this manually or in real-time for a large number of applications is inefficient and prone to errors. This is where Asynchronous Apex comes into play.

1. Batch Apex in the Project

Purpose in Project:

- To automatically update the status of all expired applications in bulk.
- Ensures that the system always reflects the correct status of every application, even if there are thousands of records.

How it works in my project:

- A Batch Apex class (ExpiredApplicationsBatch) queries all Scheme_Application__c records where End_Date__c < TODAY and Status__c != 'Expired'.
- The class then processes these records in chunks and updates their Status__c field to Expired.
- This batch job can be scheduled to run daily, so administrators do not need to manually update statuses.

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

SchemeApplicationHelper.apxc | SchemeStatusQueueable.apxc | Log executeAnonymous @9/25/2025, 5:20:10 PM | SchemeStatusBatch.apxc | Log executeAnonym

Code Coverage: None ▾ API Version: 64

```
public class SchemeStatusBatch implements Database.Batchable<SObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator(
            'SELECT Id, Submission_Date__c, Scheme__c, Status__c FROM Scheme_Application__c'
        );
    }

    public void execute(Database.BatchableContext bc, List<Scheme_Application__c> apps) {
        // ✅ Reuse your helper
        SchemeApplicationHelper.setSubmissionStatus(apps);
        update apps;

        System.debug('✅ Batch processed: ' + apps.size() + ' applications');
    }

    public void finish(Database.BatchableContext bc) {
        System.debug('✅ Batch Finished Processing All Applications');
    }
}
```

Output :

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

025, 5:53:02 PM | SchemeApplicationTrigger.apxt | SchemeSearchService.apxc * | SchemeApplicationHandler.apxc | Sche

Execution Log

Timestamp	Event	Details
19:02:56:000	CODE_UNIT_ST...	[EXTERNAL]|01pdM00000M1lQw|SchemeStatusBatch
19:02:56:021	STATEMENT_EX...	[15]|
19:02:56:021	VARIABLE_ASSI...	[31]|this|{}|0x5a5c7a7a
19:02:56:021	VARIABLE_ASSI...	[31]|jobId|"707dM00000o9HP2QAM"
19:02:56:021	VARIABLE_ASSI...	[31]|childJobId|null
19:02:56:022	METHOD_ENTRY	[1]|01pdM00000M1lQw|SchemeStatusBatch.SchemeStatusBatch()
19:02:56:022	STATEMENT_EX...	[1]|
19:02:56:022	STATEMENT_EX...	[1]|
19:02:56:022	METHOD_EXIT	[1]|SchemeStatusBatch
19:02:56:022	VARIABLE_ASSI...	[2]|this|{}|0x61f5a41f
19:02:56:022	VARIABLE_ASSI...	[2]|bc|{"jobId":"707dM00000o9HP2QAM"}|0x5a5c7a7a
19:02:56:022	STATEMENT_EX...	[2]|
19:02:56:022	STATEMENT_EX...	[3]|
19:02:56:026	SOQL_EXECUTE...	[3]|Aggregations:0|SELECT Id, Submission_Date__c, Scheme__c, Status__c FROM Scheme_Application__c
19:02:56:063	SOQL_EXECUTE...	[3]|Rows:12
19:02:56:096	CODE_UNIT_FI...	SchemeStatusBatch

2. Queueable Apex in the Project

Purpose in Project:

- To handle **smaller, immediate updates** of application statuses without running a full batch.
- Useful when a smaller set of records needs to be processed, such as applications submitted or updated recently.

How it works in your project:

- A Queueable Apex class (UpdateExpiredApplicationsQueue) is used to update expired applications in the background.
- This job can be triggered manually or automatically whenever a record is updated to check if the End_Date__c has passed.
- It allows the system to **process updates asynchronously**, so citizen submissions are not delayed.

Benefits for Government Schemes:

- **Fast Response:** Immediate status updates for smaller record sets.
- **Chaining Jobs:** Can perform sequential operations, e.g., update status → notify officer → send email to citizen.
- **Resource Management:** Reduces the load on the system compared to real-time updates for all records.

File Edit Debug Test Workspace Help < >

SchemeApplicationHelper.apxc | SchemeStatusQueueable.apxc | Log executeAnonymous @9/25/2025, 5:20:10 PM

Code Coverage: None | API Version: 64

```
public class SchemeStatusQueueable implements Queueable {
    private List<Id> appIds;

    public SchemeStatusQueueable(List<Id> appIds) {
        this.appIds = appIds;
    }

    public void execute(QueueableContext context) {
        List<Scheme_Application__c> apps = [
            SELECT Id, Submission_Date__c, Scheme__c, Status__c
            FROM Scheme_Application__c
            WHERE Id IN :appIds
        ];

        // ✅ Reuse your helper
        SchemeApplicationHelper.setSubmissionStatus(apps);
        update apps;

        System.debug('✅ Queueable processed apps: ' + apps);
    }
}
```

Output:

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

SchemeStatusBatch.apxc | Log executeAnonymous @9/25/2025, 7:01:26 PM | Log executeAnonymous @9/25/2025, 7:02:56 PM

Execution Log

Timestamp	Event	Details
19:09:57:000	CODE_UNIT_ST...	[EXTERNAL]|execute_anonymous_apex
19:09:57:002	STATEMENT_EX...	[1]|
19:09:57:002	STATEMENT_EX...	[2]|
19:09:57:003	VARIABLE_ASSI...	[2]|appIds|["a01dM000033rM7YQAU","a01dM000033zQAbQAM","a01dM0000340BKcQAM"]|0x86ca265
19:09:57:003	STATEMENT_EX...	[9]|
19:09:57:038	METHOD_ENTRY	[1]|01pdM00000M1wEz|SchemeStatusQueueable.SchemeStatusQueueable()
19:09:57:038	STATEMENT_EX...	[1]|
19:09:57:038	STATEMENT_EX...	[1]|
19:09:57:038	METHOD_EXIT	[1]|SchemeStatusQueueable
19:09:57:039	CONSTRUCTOR...	[9]|01pdM00000M1wEz|<init>(List<Id>)|SchemeStatusQueueable
19:09:57:039	VARIABLE_ASSI...	[4]|this|{}|0x22b89f0a
19:09:57:039	VARIABLE_ASSI...	[4]|appIds|["a01dM000033rM7YQAU","a01dM000033zQAbQAM","a01dM0000340BKcQAM"]|0x86ca265
19:09:57:039	STATEMENT_EX...	[1]|
19:09:57:039	STATEMENT_EX...	[2]|
19:09:57:041	STATEMENT_EX...	[4]|
19:09:57:041	STATEMENT_EX...	[5]|
19:09:57:041	VARIABLE_ASSI...	[5]|this.appIds|["a01dM000033rM7YQAU","a01dM000033zQAbQAM","a01dM0000340BKcQAM"]|0x22b89f0a
19:09:57:041	CONSTRUCTOR...	[9]|01pdM00000M1wEz|<init>(List<Id>)|SchemeStatusQueueable
19:09:57:076	CODE_UNIT_FI...	execute_anonymous_apex

scheduled Apex & Future Methods

Notes:

- Scheduled Apex was optional; batch/queueable jobs can be run on-demand.
- Queueable Apex was preferred over Future methods for bulk updates due to **chaining support and better flexibility**.

6. Exception Handling

Implementation:

- Basic error handling using System.debug() was applied.
- Ensured prototype stage ran without critical failures.

Future Scope:

- In production, advanced exception handling with **logs and notifications** can be implemented.

7. Test Classes

Notes:

- Formal unit tests were not included in the project prototype.
- In a production scenario, **test classes are mandatory** to achieve ≥75% code coverage and verify business logic.