**How to make Custom Approval Request using Salesforce Screen Flow**Implementing a Custom Approval Request using Salesforce Screen Flow allows you to approve or reject requests in bulk. The standard approval Request in Salesforce does not allow for bulk approval or rejection of requests, making this customization essential for organizations looking to streamline their workflow. Key features include:

* **Bulk Approvals:** Quickly approve or reject multiple requests at once, saving time and effort.
* **Custom Logic:** Tailor the approval process to meet specific business needs, including conditional approvals.
* **User-Friendly Interface:** Enhance user experience with an intuitive flow that simplifies navigation and decision-making.
* **Real-Time Notifications:** Keep stakeholders informed with immediate updates on approval status.

By leveraging this customized approach, organizations can improve efficiency and maintain better control over their approval processes, overcoming the limitations of the standard process.

**Step 1:** **Create a New Custom Object:**

* In Object Manager, click the *Create* button, then select *Custom Object* from the dropdown.

 **Enter Custom Object Information:**

* **Label:** ProcessInstanceDisplay
* **Plural Label:** ProcessInstanceDisplays
* **Object Name:** This will automatically populate as ProcessInstanceDisplay.
* **Record Name:** Choose a data type for your record name field (e.g., Text or Auto Number).

 **Set Optional Settings:**

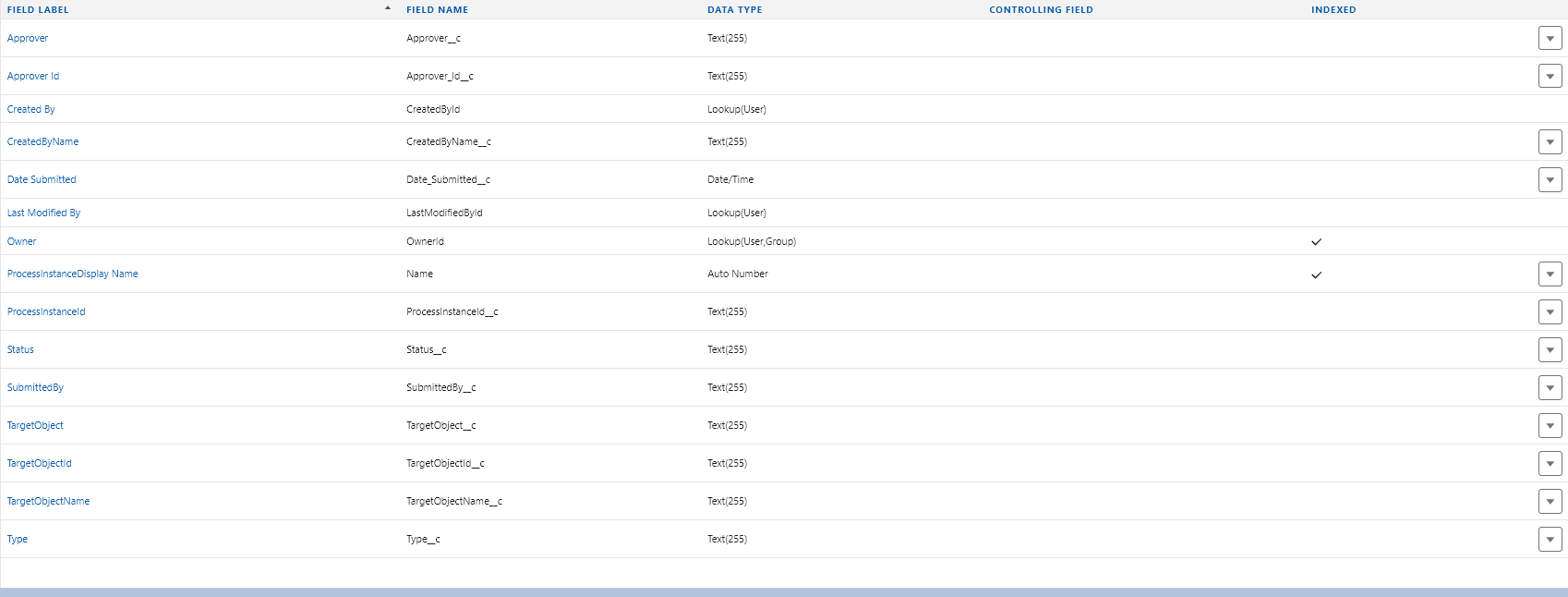
* Set any optional features such as enabling reports, allowing activities, tracking field history, etc.

 **Save the Custom Object:**

* Click *Save* to create the object.

### Step 2: Add Custom Fields List of Fields to Add:

1. **Approver**
   * **Data Type:** Text (255)
   * **Field Name:** Approver\_\_c
2. **Approver Id**
   * **Data Type:** Text (255)
   * **Field Name:** Approver\_Id\_\_c
3. **Created By Name**
   * **Data Type:** Text (255)
   * **Field Name:** CreatedByName\_\_c
4. **Date Submitted**
   * **Data Type:** Date/Time
   * **Field Name:** Date\_Submitted\_\_c
5. **ProcessInstanceId**
   * **Data Type:** Text (255)
   * **Field Name:** ProcessInstanceId\_\_c
6. **Status**
   * **Data Type:** Text (255)
   * **Field Name:** Status\_\_c
7. **SubmittedBy**
   * **Data Type:** Text (255)
   * **Field Name:** SubmittedBy\_\_c
8. **TargetObject**
   * **Data Type:** Text (255)
   * **Field Name:** TargetObject\_\_c
9. **TargetObjectId**
   * **Data Type:** Text (255)
   * **Field Name:** TargetObjectId\_\_c
10. **TargetObject Name**
    * **Data Type:** Text (255)
    * **Field Name:** TargetObjectName\_\_c
11. **Type**
    * **Data Type:** Text (255)
    * **Field Name:** Type\_\_c  
        
        
        
        
        
        
        
        
        
        
      **// Screenshot Attached for your Reference**



### Step 3: Steps to Create the Apex Class:

1. **Login to Salesforce:**
   * Go to your Salesforce org and log in with your credentials.
2. **Navigate to the Apex Classes:**
   * Click the gear icon on the top right and go to **Setup**.
   * In the Quick Find box, search for "Apex Classes" and click on it under the "Custom Code" section.
3. **Create a New Apex Class:**
   * In the Apex Classes page, click the New button to create a new class.
4. **Write the Code for PopulateProcessInstanceDisplayBatch:**
   * In the editor that opens, copy and paste the following code:

global class PopulateProcessInstanceDisplayBatch implements Database.Batchable<SObject> {

global Database.QueryLocator start(Database.BatchableContext BC) {

// Query only ProcessInstance records with a Status of 'Pending'

return Database.getQueryLocator([

SELECT Id, Status, CreatedBy.Name, TargetObjectId, SubmittedById, CreatedDate

FROM ProcessInstance

WHERE Status = 'Pending' // Filter for pending items

]);

}

global void execute(Database.BatchableContext BC, List<ProcessInstance> scope) {

List<ProcessInstanceDisplay\_\_c> recordsToInsert = new List<ProcessInstanceDisplay\_\_c>();

Set<Id> targetObjectIds = new Set<Id>();

Set<Id> existingProcessInstanceIds = new Set<Id>();

Set<Id> processInstanceIds = new Set<Id>();

// Collect all TargetObjectIds and ProcessInstanceIds from the scope

for (ProcessInstance instance : scope) {

if (instance.TargetObjectId != null) {

targetObjectIds.add(instance.TargetObjectId);

}

if (instance.Id != null) {

processInstanceIds.add(instance.Id);

}

}

// Map to store TargetObjectId to Name (dynamic approach)

Map<Id, String> targetObjectNameMap = new Map<Id, String>();

// Fetch the TargetObject names dynamically using describe and SOQL

for (Id targetId : targetObjectIds) {

String sObjectType = targetId.getSObjectType().getDescribe().getName();

if (Schema.getGlobalDescribe().get(sObjectType).getDescribe().fields.getMap().containsKey('Name')) {

try {

String query = 'SELECT Name FROM ' + sObjectType + ' WHERE Id = :targetId';

SObject targetRecord = Database.query(query);

if (targetRecord != null && targetRecord.get('Name') != null) {

targetObjectNameMap.put(targetId, (String) targetRecord.get('Name'));

}

} catch (Exception ex) {

System.debug('Failed to fetch Name for ' + targetId + ': ' + ex.getMessage());

}

} else {

System.debug('No Name field for object type: ' + sObjectType);

}

}

// Get existing records to avoid duplication

for (ProcessInstance instance : scope) {

if (instance.Id != null) {

existingProcessInstanceIds.add(instance.Id);

}

}

List<ProcessInstanceDisplay\_\_c> existingRecords = [

SELECT ProcessInstanceId\_\_c

FROM ProcessInstanceDisplay\_\_c

WHERE ProcessInstanceId\_\_c IN :existingProcessInstanceIds

];

Set<Id> existingProcessInstanceIdsSet = new Set<Id>();

for (ProcessInstanceDisplay\_\_c record : existingRecords) {

existingProcessInstanceIdsSet.add(record.ProcessInstanceId\_\_c);

}

// Query OriginalActorId from ProcessInstanceWorkitem

Map<Id, Id> approverIdMap = new Map<Id, Id>();

List<ProcessInstanceWorkitem> workitems = [

SELECT Id, ProcessInstanceId, OriginalActorId

FROM ProcessInstanceWorkitem

WHERE ProcessInstanceId IN :processInstanceIds

];

for (ProcessInstanceWorkitem workitem : workitems) {

approverIdMap.put(workitem.ProcessInstanceId, workitem.OriginalActorId); // Map ProcessInstanceId to OriginalActorId

}

// Process each ProcessInstance and map to the display object

for (ProcessInstance instance : scope) {

if (existingProcessInstanceIdsSet.contains(instance.Id)) {

System.debug('Record already exists for ProcessInstance: ' + instance.Id);

continue; // Skip already existing records

}

System.debug('Processing instance: ' + instance.Id);

ProcessInstanceDisplay\_\_c displayRecord = new ProcessInstanceDisplay\_\_c();

displayRecord.ProcessInstanceId\_\_c = instance.Id;

displayRecord.Status\_\_c = instance.Status;

displayRecord.CreatedByName\_\_c = instance.CreatedBy.Name; // Assuming this is valid

// Set the Target Object ID and Name dynamically

displayRecord.TargetObjectId\_\_c = instance.TargetObjectId; // Store the ID

if (instance.TargetObjectId != null && targetObjectNameMap.containsKey(instance.TargetObjectId)) {

displayRecord.TargetObjectName\_\_c = targetObjectNameMap.get(instance.TargetObjectId); // Store the Name

}

// Set the Type based on the TargetObjectId

if (instance.TargetObjectId != null) {

String objectTypeName = instance.TargetObjectId.getSObjectType().getDescribe().getLabel();

displayRecord.Type\_\_c = objectTypeName; // Populate Type\_\_c with the object name

}

// Populate Date\_Submitted\_\_c with the CreatedDate from ProcessInstance

displayRecord.Date\_Submitted\_\_c = instance.CreatedDate; // Set the date submitted

// Query SubmittedBy name

if (instance.SubmittedById != null) {

String submittedByName = [SELECT Name FROM User WHERE Id = :instance.SubmittedById].Name;

displayRecord.SubmittedBy\_\_c = submittedByName;

}

// Populate the Approver\_\_c field from the approverIdMap

if (approverIdMap.containsKey(instance.Id)) {

Id originalApproverId = approverIdMap.get(instance.Id); // Get the OriginalActorId

String approverName = [SELECT Name FROM User WHERE Id = :originalApproverId].Name; // Query the User's Name

displayRecord.Approver\_\_c = approverName; // Set the Approver's Name

displayRecord.Approver\_Id\_\_c=originalApproverId;

System.debug('Approver populated for instance: ' + instance.Id + ' - ' + displayRecord.Approver\_\_c);

} else {

System.debug('No approver found for ProcessInstance: ' + instance.Id);

}

recordsToInsert.add(displayRecord);

}

// Insert the records if there are any to insert

System.debug('Records to insert: ' + recordsToInsert);

if (!recordsToInsert.isEmpty()) {

try {

insert recordsToInsert;

} catch (DmlException e) {

System.debug('Error inserting records: ' + e.getMessage());

}

}

}

global void finish(Database.BatchableContext BC) {

// Optional: Send a notification or log completion

System.debug('Batch job finished successfully.');

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
**Create Another Apex class label it as BulkApprovalProcess**// paste the given below code  
  
public class BulkApprovalProcess {

@InvocableMethod

public static void bulkApprove(List<Id> processInstanceIds) {

// Querying ProcessInstanceWorkitem based on the provided ProcessInstance IDs

List<ProcessInstanceWorkitem> items = [

SELECT Id

FROM ProcessInstanceWorkitem

WHERE ProcessInstanceId IN :processInstanceIds

];

// Loop through each work item and process the approval

for (ProcessInstanceWorkitem item : items) {

Approval.ProcessWorkitemRequest req = new Approval.ProcessWorkitemRequest();

req.setWorkItemId(item.Id);

req.setAction('Approve'); // Action for approval

Approval.process(req);

}

}

}  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Create Another Apex class Label it as BulkRejectionProcess**  
  
//Paste the Given Below code  
  
  
public class BulkRejectionProcess {

@InvocableMethod

public static void bulkReject(List<Id> processInstanceIds) {

// Querying ProcessInstanceWorkitem based on the provided ProcessInstance IDs

List<ProcessInstanceWorkitem> items = [

SELECT Id

FROM ProcessInstanceWorkitem

WHERE ProcessInstanceId IN :processInstanceIds

];

// Loop through each work item and process the rejection

for (ProcessInstanceWorkitem item : items) {

Approval.ProcessWorkitemRequest req = new Approval.ProcessWorkitemRequest();

req.setWorkItemId(item.Id);

req.setAction('Reject'); // Action for rejection

Approval.process(req);

}

}

}  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Steps 4: Steps to Create the Scheduled Apex Class:

1. **Login to Salesforce:**
   * Log in to your Salesforce org with your credentials.
2. **Navigate to Apex Classes:**
   * In the top-right corner, click the gear icon and select **Setup**.
   * In the Quick Find box, type Apex Classes and click on it under the "Custom Code" section.
3. **Create a New Apex Class:**
   * In the **Apex Classes** page, click the New button to create a new class.
4. **Write the label it   
   ScheduleProcessInstanceDisplayBatch:**
   * In the editor that opens, paste the following code:

global class ScheduleProcessInstanceDisplayBatch implements Schedulable {

global void execute(SchedulableContext SC) {

// Instantiate the batch class

PopulateProcessInstanceDisplayBatch batch = new PopulateProcessInstanceDisplayBatch();

// Execute the batch job

Database.executeBatch(batch, 200); // You can specify the batch size here, e.g., 200

}

}  
  
  
  
  
  
  
**Note->**

**Important Note: How to Open Anonymous Window**  
  
  **Log in to Salesforce**:

* Go to your Salesforce organization and log in.

 **Open Developer Console**:

* Click on the gear icon (⚙️) in the upper right corner.
* Select **Developer Console** from the dropdown menu.

 **Open the Execute Anonymous Window**:

* In the Developer Console, click on **Debug** in the menu.
* Select **Open Execute Anonymous Window** (or press Ctrl + E on Windows, Command + E on Mac).
* If you want to Run the Batch Instant then paste the below code in Anonymous Window

**// Run the Batch instant**

PopulateProcessInstanceDisplayBatch batchJob = new PopulateProcessInstanceDisplayBatch();

Database.executeBatch(batchJob);

* If you want to Schedule the Batch for every 1 hour then paste the below code in Anonymous Window

**// Start the Batch**

String jobName = 'Scheduled ProcessInstance Display Batch';

String cronExpression = '0 0 \* \* \* ?'; // Every hour at minute 0

System.schedule(jobName, cronExpression, new ScheduleProcessInstanceDisplayBatch());

**Note: This Batch will Run at every 1 hour**

It is scheduled

* If you want to Stop the Scheduled Batch Get the Job Id and Paste the below code in Anonymous Window

**//Stop the Batch**

### // The name of the scheduled job to stop

### String jobName = 'Scheduled ProcessInstance Display Batch';

### // Query for the job in the CronTrigger object

### CronTrigger ct = [SELECT Id, State FROM CronTrigger WHERE CronJobDetail.Name = :jobName LIMIT 1];

### // Check if the job exists and is active

### if (ct != null && ct.State == 'WAITING') {

### // Abort the scheduled job

### System.abortJob(ct.Id);

### System.debug('Scheduled job stopped successfully.');

### } else {

### System.debug('No active scheduled job found with the name: ' + jobName);

### } Step 5: Create a Flow Builder

1. **Log in to Salesforce**.
2. Click on the **App Launcher** (grid icon) in the upper left corner.
3. Search for and select **Flows**.
4. Click on the **New Flow** button.

### Step 2: Choose Flow Type

1. In the **Flow Builder** modal, select **Screen Flow**.
2. Click on **Next**.

### Step 3: Configure Flow

1. **Set Flow Label**:
   * In the Flow Builder, you’ll see a section on the left for your flow properties.
   * Set the **Flow Label** to **Bulk Approval Custom**.
   * The **API Name** will auto-generate based on your label, but you can modify it if needed.

