**Salesforce Project Manifest File & Permission Set Assignment: A Guide for Reviewers**

This document provides a detailed explanation of the package.xml manifest file and my submission for this Salesforce project. The manifest file is the authoritative source for all metadata components developed, ensuring a complete and verifiable submission for review.

**1. Project Organization and Component Location**

The project adheres to the standard Salesforce DX folder structure. All Apex classes, Lightning Web Components, custom objects, and other assets, are organized under the force-app/main/default directory. The package.xml manifest, which lists all deployable components, is placed in the manifest folder to serve as the central reference point.

**2. Components of package.xml (COMPLETE PACKAGE)**

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<Package xmlns="http://soap.sforce.com/2006/04/metadata">

    <types>

        <members>Account.Last\_High\_Value\_Opportunity\_\_c</members>

        <members>Opportunity.Closed\_Won\_Reason\_\_c</members>

        <members>Opportunity.High\_Value\_Processed\_Date\_\_c</members>

        <members>Opportunity.High\_Value\_Processed\_\_c</members>

        <members>Opportunity\_Win\_Processing\_Error\_\_c.Opportunity\_\_c</members>

        <members>Opportunity\_Win\_Processing\_Error\_\_c.Error\_Message\_\_c</members>

        <members>Opportunity\_Win\_Processing\_Error\_\_c.Is\_Active\_\_c</members>

        <name>CustomField</name> *<!--pending some field check once-->*

    </types>

    <types>

        <members>Opportunity\_Build\_Permission\_Set</members>

        <name>PermissionSet</name>

    </types>

    <types>

        <members>OpportunityWinProcessor</members>

        <members>OpportunityTriggerHandler</members>

        <members>OpportunityWinProcessorTest</members>

        <members>OpportunityTriggerHandlerTest</members>

        <members>OpportunityRetryController</members>

        <members>OpportunityRetryControllerTest</members>

*<!--<members>TriggerControl</members>-->*

        <name>ApexClass</name>

    </types>

    <types>

        <members>OpportunityTrigger</members>

        <name>ApexTrigger</name>

    </types>

    <types>

        <members>highValueOpportunityDashboard</members>

        <members>processingErrorList</members>

        <members>closedWonOpportunityList</members>*<!--check if wire working properly-->*

        <name>LightningComponentBundle</name>

    </types>

    <version>60.0</version>

</Package>

**3. Permission Set Assignment:**

**List**<String> permissionSetNames = new **List**<String>{‘Opportunity\_Build\_Permission\_Set'};

String profile = 'System Administrator';

**List**<**PermissionSet**> perSetList = [Select Id,Name From PermissionSet where Name = :permissionSetNames];

**List**<**User**> userList = [SELECT Id, Name from User where Profile.Name =:profile and Name in ('Aditya Gade') OR  Name like '%Aditya%' ];

System.debug('UserList size = '+userList.size());

System.debug('UserList = '+userList);

System.debug('PermissionSet = '+ perSetList);

if(!userList.isEmpty() && !perSetList.isEmpty()){

**List**<**PermissionSetAssignment**> permSetAssList = new **List**<**PermissionSetAssignment**>();

    for(**PermissionSet** perSet : perSetList){

        for(**User** usr : userList){

**PermissionSetAssignment** perSetAss = new **PermissionSetAssignment**();

        perSetAss.AssigneeId = usr.ID;

        perSetAss.PermissionSetId = perSet.Id;

        permSetAssList.add(perSetAss);

        }

    }

    if(!permSetAssList.isEmpty()){

        upsert permSetAssList;

        System.debug('PermissionSetAssignment = '+ permSetAssList);

    }

}

**4. Components and Classes Overview:**

* **Opportunity\_Build\_Permission\_Set   
  TO enable our user to interact with the Opportunity Dashboard and Error Logging components.**
* **Apex Classes used :**

| * + **Apex Class** | * + **Role** |
| --- | --- |
| * + **OpportunityWinProcessor** | * + **Main business logic to process high-value opportunities (creates Task, updates Account, sets flags).** |
| * + **OpportunityTriggerHandler** | * + **Handles after-insert and after-update logic on Opportunities to auto-process them if they qualify.** |
| * + **OpportunityRetryController** | * + **Exposed to LWC via @AuraEnabled, fetches recent Opportunities, failed errors, and retries processing.** |
| * + **OpportunityWinProcessorTest** | * + **Unit test for OpportunityWinProcessor.** |
| * + **OpportunityTriggerHandlerTest** | * + **Unit test for trigger logic.** |
| * + **OpportunityRetryControllerTest** | * + **Unit test for LWC-exposed methods and retry behavior.** |

* **ApexTrigger: (OpportunityTrigger)**

**Trigger file to call the OpportunityTriggerHandler for after insert and after update.  
It auto-processes any newly created or updated Opportunity that meets high-value conditions and is bulkified.**

* **LWC :**

| **Component** | **Role** |
| --- | --- |
| highValueOpportunityDashboard | **Parent LWC** – Loads recent opportunities and errors, handles retry events, shows spinners/toasts. |
| processingErrorList | **Child LWC 1** – Displays failed opportunity processing errors in a datatable with retry buttons. |
| closedWonOpportunityList | **Child LWC 2** – Displays the 10 most recent Closed Won opportunities in a clean table. |