HELLO WORLD LIGHTNING APP

helloWorld.html

```
<template>
  <lightning-card title="HelloWorld" icon-
name="custom:custom14">
    <div class="slds-m-around_medium">
      <p>Hello, {greeting}!</p>
      <lightning-input label="Name" value={greeting}
onchange={changeHandler}></lightning-input>
    </div>
  </lightning-card>
</template>
```

helloWorld.js

```
import { LightningElement } from 'lwc';
export default class HelloWorld extends LightningElement {
  greeting = 'World';
  changeHandler(event) {
    this.greeting = event.target.value;
  }
}
```

helloWorld.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata"
fqn="helloWorld">
  <apiVersion>52.0</apiVersion>
  <isExposed>true</isExposed>
  <targets>
```

```
    <target>lightning__AppPage</target>
    <target>lightning__RecordPage</target>
    <target>lightning__HomePage</target>
  </targets>
</LightningComponentBundle>
```

BIKECARD LIGHTNING WEB COMPONENT

bikeCard.html

```
<template>
    <div>
        <div>Name: {name}</div>
        <div>Description: {description}</div>
        <lightning-badge label={material}></lightning-badge>
        <lightning-badge label={category}></lightning-badge>
        <div>Price: {price}</div>
        <div><img src={pictureUrl}/></div>
    </div>
</template>
```

bikeCard.js

```
import { LightningElement } from 'lwc';
export default class BikeCard extends LightningElement {
   name = 'Electra X4';
   description = 'A sweet bike built for comfort.';
   category = 'Mountain';
   material = 'Steel';
   price = '$2,700';
   pictureUrl = 'https://s3-us-west-1.amazonaws.com/sfdc-
demo/ebikes/electrax4.jpg';
 }
```

bikeCard.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata">
    <!-- The apiVersion may need to be increased for the current
release -->
    <apiVersion>52.0</apiVersion>
    <isExposed>true</isExposed>
    <masterLabel>Product Card</masterLabel>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

ADD STYLES AND DATA TO BIKECARD LIGHTNING WEB COMPONENT

detail.css

```
body{
  margin: 0;
}
.price {
  color: green;
  font-weight: bold;
}
```

detail.html

```
<template>
    <template if:true={product}>
    <div class="container">
        <div class="slds-text-
heading_small">{product.fields.Name.value}</div>
        <div
class="price">{product.fields.MSRP__c.displayValue}</div>
        <div
class="description">{product.fields.Description__c.value}</div>
        <img class="product-img"
src={product.fields.Picture_URL__c.value}></img>
        <p>
            <lightning-badge
label={product.fields.Material__c.value}></lightning-badge>
            <lightning-badge
label={product.fields.Level__c.value}></lightning-badge>
        </p>
        <p>
            <lightning-badge
label={product.fields.Category__c.value}></lightning-badge>
        </p>
    </div>
    </template>
    <template if:false={product}>
    <div class="slds-text-heading_medium">Select a bike</div>
    </template>
</template>
```

selector.js

```
import { LightningElement, wire } from 'lwc';
import { getRecord, getFieldValue } from
'lightning/uiRecordApi';
import Id from '@salesforce/user/Id';
import NAME_FIELD from '@salesforce/schema/User.Name';
```

```
const fields = [NAME_FIELD];
export default class Selector extends LightningElement {
    selectedProductId;
    handleProductSelected(evt) {
        this.selectedProductId = evt.detail;
    }
    userId = Id;
    @wire(getRecord, { recordId: '$userId', fields })
    user;
    get name() {
        return getFieldValue(this.user.data, NAME_FIELD);
    }
}
```

selector.html

```
<template>
    <div class="wrapper">
    <header class="header">Available Bikes for {name}</header>
    <section class="content">
        <div class="columns">
        <main class="main" >
            <c-list
onproductselected={handleProductSelected}></c-list>
        </main>
        <aside class="sidebar-second">
            <c-detail product-id={selectedProductId}></c-detail>
        </aside>
        </div>
    </section>
    </div>
</template>
```

APEX REST CALLOUTS

AnimalLocator Class

```
public class AnimalLocator {
     public static String getAnimalNameById(Integer animalId) {
          String animalName;
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint('https://th-apex-http-
callout.herokuapp.com/animals/'+animalId);
        request.setMethod('GET');
        HttpResponse response = http.send(request);
        // If the request is successful, parse the JSON
response.
        if(response.getStatusCode() == 200) {
            Map<String, Object> r = (Map<String, Object>)
            JSON.deserializeUntyped(response.getBody());
            Map<String, Object> animal = ( Map<String, Object>)
r.get('animal');
            animalName = string.valueOf(animal.get('name'));
        }
        return animalName;
     }
}
```

AnimalLocatorMock Class

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
    // Implement this interface method
    global HTTPResponse respond(HTTPRequest request) {
        // Create a fake response
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
```

```
response.setBody('{"animal":{"id":1,"name":"chicken","eats":"chicken food","says":"cluck cluck"}}');
        response.setStatusCode(200);
        return response;
    }
}
```

AnimalLocatorTest Class

```
@isTest
private class AnimalLocatorTest {
    @isTest static void getAnimalNameByIdTest(){
    // Set mock callout class
    Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
    // This causes a fake response to be sent
    // from the class that implements HttpCalloutMock.
    String response = AnimalLocator.getAnimalNameById(1);

    System.assertEquals('chicken', response);
    }
}
```

APEX SOAP CALLOUTS

ParkLocator Class

```
public class ParkLocator {
    public static List<String> country(String country) {
        ParkService.ParksImplPort parkservice =
            new ParkService.ParksImplPort();
        return parkservice.byCountry(country);
    }
}
```

ParkServiceMock Class

```
@isTest
global class ParkServiceMock implements WebServiceMock {
   global void doInvoke(
           Object stub,
           Object request,
           Map<String, Object> response,
           String endpoint,
           String soapAction,
           String requestName,
           String responseNS,
           String responseName,
           String responseType) {
        // start - specify the response you want to send
        List<String> parks = new List<string>();
                parks.add('Yosemite');
                parks.add('Yellowstone');
                parks.add('Another Park');
        ParkService.byCountryResponse response_x =
            new ParkService.byCountryResponse();
        response_x.return_x = parks;
        // end
        response.put('response_x', response_x);
   }
}
```

ParkLocatorTest Class

```
@isTest
public class ParkLocatorTest {
    @isTest static void testCallout() {
        // This causes a fake response to be generated
        Test.setMock(WebServiceMock.class, new
```

```
ParkServiceMock());
        // Call the method that invokes a callout
        String country = 'United States';
        List<String> result = ParkLocator.country(country);
        List<String> parks = new List<String> ();
        parks.add('Yosemite');
        parks.add('Yellowstone');
        parks.add('Another Park');
        // Verify that a fake result is returned
        System.assertEquals(parks, result);
    }
}
```

APEX WEB SERVICES

AccountManager Class

```
@RestResource(urlMapping='/Accounts/*/contacts')
global with sharing class AccountManager {
    @HttpGet
    global static Account getAccount() {
        RestRequest request = RestContext.request;
        // grab the caseId from the end of the URL
        String accountId =
request.requestURI.substringBetween('/Accounts/','/contacts');

        Account result =  [SELECT Id, Name, (Select Id, Name
from Contacts ) from Account where Id = :accountId];
        return result;
    }
}
```

AccountManagerTest Class

```
@IsTest
private class AccountManagerTest {
    @isTest static void testGetContactsByAccountId() {
        Id recordId = createTestRecord();
        // Set up a test request
        RestRequest request = new RestRequest();
        request.requestUri =
            'https://yourInstance.my.salesforce.com/services/apexrest/Accounts/'+recordId+'/contacts';
        request.httpMethod = 'GET';
        RestContext.request = request;
        // Call the method to test
        Account thisAccount = AccountManager.getAccount();
        // Verify results
        System.assert(thisAccount != null);
        System.assertEquals('Test record', thisAccount.Name);
    }
    // Helper method
    static Id createTestRecord() {
        // Create test record
        Account accountTest = new Account(
            Name='Test record');
        insert accountTest;

        Contact contactTest = new Contact(
            FirstName = 'John',
            LastName = 'Doe',
            AccountId = accountTest.Id
        );
            insert contactTest;

            return accountTest.Id;
    }
}
```

CREATE AND EDIT VISUALFORCE PAGES

Visualforce page

```
<apex:page showHeader="false" sidebar="false">

    <apex:image
url="https://developer.salesforce.com/files/salesforce-
developer-network-logo.png"/>

</apex:page>
```

USE SIMPLE VARIABLE AND FORMULAS

Simple Information Display

```
<apex:page>

    <apex:pageBlock title="User Status">
        <apex:pageBlockSection columns="1">

            {! $User.FirstName } {! $User.LastName }
            ({! $User.Username }) <br />

            {! $User.FirstName & ' ' & $User.LastName } <br />

            <p> Today's Date is {! TODAY() } </p>
            <p> Next week it will be {! TODAY() + 7 } </p>

            <p>The year today is {! YEAR(TODAY()) }</p>
            <p>Tomorrow will be day number  {! DAY(TODAY() + 1)
```

```
}</p>
            <p>Let's find a maximum: {!
MAX(1,2,3,4,5,6,5,4,3,2,1) } </p>
            <p>The square root of 49 is {! SQRT(49) }</p>
            <p>Is it true?  {! CONTAINS('salesforce.com',
'force.com') }</p>

            <p>{! IF(
CONTAINS('salesforce.com','force.com'),'Yep', 'Nope') }</p>
            <p>{! IF( DAY(TODAY()) < 15,'Before the 15th', 'The
15th or after') }</p>

        </apex:pageBlockSection>
    </apex:pageBlock>

</apex:page>
```

User FirstName Display

```
<apex:page >
    {! $User.FirstName}

</apex:page>
```

STANDARD CONTROLLER

```
<apex:page standardController="Contact">

    <apex:pageBlock title="Contact Summary">

        <apex:pageBlockSection>

            First Name: {! Contact.FirstName } <br/>
```

```
            Last Name: {! Contact.LastName } <br/>
            Owner Email: {! Contact.Owner.Email } <br/>

        </apex:pageBlockSection>

    </apex:pageBlock>

</apex:page>
```

OPPURTUNITY DETAIL

```
<apex:page standardController="Opportunity" >

    <apex:outputField value="{!Opportunity.Name}" /> <br />
    <apex:outputField value="{!Opportunity.Amount}" /> <br />
    <apex:outputField value="{!Opportunity.CloseDate}" /> <br />
    <apex:outputField value="{!Opportunity.Account.Name}" /> <br
/>

</apex:page>
```

CREATE CONTACT

```
<apex:page  standardController="Contact" >

    <apex:form>

        <apex:inputField label = "First Name"
value="{!Contact.FirstName}" /> <br />
        <apex:inputField label = "Last Name"
value="{!Contact.LastName}" /> <br />
        <apex:inputField label = "Email"
```

```
value="{!Contact.Email}" /> <br /> <br />

        <apex:commandButton action="{!save}" value="save" />

    </apex:form>

</apex:page>
```

STANDARD LIST CONTROLLER

```
<apex:page standardController="Account" recordSetVar="accounts">
    <apex:form>

        <apex:repeat var="a" value="{!accounts}" >

            <li >
                <apex:outputLink value="/{!a.id}" target="_new">
                    {!a.name}
                </apex:outputLink>
            </li>

        </apex:repeat>

    </apex:form>
</apex:page>
```

STATIC RESOURCES

```
<apex:page >
```

```
    <apex:image alt="cat" title="cat"
                url="{!URLFOR($Resource.vfimagetest,
'cats/kitten1.jpg')}"/>

</apex:page>
```

CUSTOM CONTROLLER

NewCaseList

```
<apex:page controller="NewCaseListController">

    <apex:repeat value="{!NewCases}" var="case" >
       <li> <apex:outputLink value="/{!case.id}" target="_new">
            {!case.CaseNumber}
        </apex:outputLink> </li>

    </apex:repeat>

</apex:page>
```

NewCaseListController

```
public class NewCaseListController{
```

```
public List<Case> getNewCases() {

    List<Case> results = Database.query(
        'SELECT Id, CaseNumber from Case where Status =
\'New\'');
    return results;
  }
}
```

CONTACT FORM

```
<apex:page standardController="Contact">
          <head>
                        <meta charset="utf-8" />
                        <meta name="viewport"
content="width=device-width, initial-scale=1" />
                        <title>Quick Start: Visualforce</title>
                        <!-- Import the Design System style sheet
-->
                        <apex:slds />
          </head>
          <body>
                      <apex:form >
                      <apex:pageBlock title="New Contact">
                      <!--Buttons -->
                      <apex:pageBlockButtons >
                      <apex:commandButton action="{!save}"
value="Save"/>
                      </apex:pageBlockButtons>
                      <!--Input form -->
                      <apex:pageBlockSection columns="1">
                      <apex:inputField
value="{!Contact.Firstname}"/>
                      <apex:inputField
```

```
value="{!Contact.Lastname}"/>
                        <apex:inputField value="{!Contact.Email}"/>
                        </apex:pageBlockSection>
                        </apex:pageBlock>
                        </apex:form>

           </body>
</apex:page>
```

APEX SPECIALIST

MAINTENANCE REQUEST

```
trigger MaintenanceRequest on Case (before update, after update)
{

    if(Trigger.isUpdate && Trigger.isAfter){

         MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
Trigger.OldMap);

    }

}
```

MAINTENANCE REQUEST HELPER

```
public with sharing class MaintenanceRequestHelper {
    public static void updateworkOrders(List<Case>
```

```
updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();


        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' &&
c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine
Maintenance'){
                    validIds.add(c.Id);


                }
            }
        }

        if (!validIds.isEmpty()){
            List<Case> newCases = new List<Case>();
            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT
Id, Vehicle__c, Equipment__c,
Equipment__r.Maintenance_Cycle__c,(SELECT
Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
                                                          FROM
Case WHERE Id IN :validIds]);
            Map<Id,Decimal> maintenanceCycles = new
Map<ID,Decimal>();
            AggregateResult[] results = [SELECT
Maintenance_Request__c,
MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
Equipment_Maintenance_Item__c WHERE Maintenance_Request__c IN
:ValidIds GROUP BY Maintenance_Request__c];

        for (AggregateResult ar : results){
            maintenanceCycles.put((Id)
ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }
```

```
            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
                    Subject = 'Routine Maintenance',
                    Type = 'Routine Maintenance',
                    Vehicle__c = cc.Vehicle__c,
                    Equipment__c =cc.Equipment__c,
                    Origin = 'Web',
                    Date_Reported__c = Date.Today()

                );

                If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_Due__c =
Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
                } else {
                    nc.Date_Due__c =
Date.today().addDays((Integer)
cc.Equipment__r.maintenance_Cycle__c);
                }

                newCases.add(nc);
             }

            insert newCases;

            List<Equipment_Maintenance_Item__c> clonedWPs = new
List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c wp :
closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c wpClone =
wp.clone();
                    wpClone.Maintenance_Request__c = nc.Id;
                    ClonedWPs.add(wpClone);
```

```
                }
            }
            insert ClonedWPs;
        }
    }
}
```

WAREHOUSE CALLOUT SERVICE

```
public with sharing class WarehouseCalloutService implements
Queueable {
    private static final String WAREHOUSE_URL = 'https://th-
superbadge-apex.herokuapp.com/equipment';

    //class that makes a REST callout to an external warehouse
system to get a list of equipment that needs to be updated.
    //The callout's JSON response returns the equipment records
that you upsert in Salesforce.

    @future(callout=true)
    public static void runWarehouseEquipmentSync(){
        Http http = new Http();
        HttpRequest request = new HttpRequest();

        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);

        List<Product2> warehouseEq = new List<Product2>();

        if (response.getStatusCode() == 200){
            List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            //class maps the following fields: replacement part
```

```
(always true), cost, current inventory, lifespan, maintenance
cycle, and warehouse SKU
            //warehouse SKU will be external ID for identifying
which equipment records to update within Salesforce
            for (Object eq : jsonResponse){
                Map<String,Object> mapJson =
(Map<String,Object>)eq;
                Product2 myEq = new Product2();
                myEq.Replacement_Part__c = (Boolean)
mapJson.get('replacement');
                myEq.Name = (String) mapJson.get('name');
                myEq.Maintenance_Cycle__c = (Integer)
mapJson.get('maintenanceperiod');
                myEq.Lifespan_Months__c = (Integer)
mapJson.get('lifespan');
                myEq.Cost__c = (Integer) mapJson.get('cost');
                myEq.Warehouse_SKU__c = (String)
mapJson.get('sku');
                myEq.Current_Inventory__c = (Double)
mapJson.get('quantity');
                myEq.ProductCode = (String) mapJson.get('_id');
                warehouseEq.add(myEq);
            }

            if (warehouseEq.size() > 0){
                upsert warehouseEq;
                System.debug('Your equipment was synced with the
warehouse one');
            }
        }
    }

    public static void execute (QueueableContext context){
        runWarehouseEquipmentSync();
    }

}
```

WAREHOUSE SYNC SCHEDULE

```
global with sharing class WarehouseSyncSchedule implements
Schedulable{
    global void execute(SchedulableContext ctx){
        System.enqueueJob(new WarehouseCalloutService());
    }
}
```

MAINTENANCE REQUEST HELPER TEST

```
@istest
public with sharing class MaintenanceRequestHelperTest {

    private static final string STATUS_NEW = 'New';
    private static final string WORKING = 'Working';
    private static final string CLOSED = 'Closed';
    private static final string REPAIR = 'Repair';
    private static final string REQUEST_ORIGIN = 'Web';
    private static final string REQUEST_TYPE = 'Routine
Maintenance';
    private static final string REQUEST_SUBJECT = 'Testing
subject';

    PRIVATE STATIC Vehicle__c createVehicle(){
        Vehicle__c Vehicle = new Vehicle__C(name =
'SuperTruck');
        return Vehicle;
    }

    PRIVATE STATIC Product2 createEq(){
        product2 equipment = new product2(name =
```

```
'SuperEquipment',
                                                lifespan_months__C =
10,
                                                maintenance_cycle__C =
10,
                                                replacement_part__c =
true);
        return equipment;
    }

    PRIVATE STATIC Case createMaintenanceRequest(id vehicleId,
id equipmentId){
        case cs = new case(Type=REPAIR,
                           Status=STATUS_NEW,
                           Origin=REQUEST_ORIGIN,
                           Subject=REQUEST_SUBJECT,
                           Equipment__c=equipmentId,
                           Vehicle__c=vehicleId);
        return cs;
    }

    PRIVATE STATIC Equipment_Maintenance_Item__c
createWorkPart(id equipmentId,id requestId){
        Equipment_Maintenance_Item__c wp = new
Equipment_Maintenance_Item__c(Equipment__c = equipmentId,

Maintenance_Request__c = requestId);
        return wp;
    }


    @istest
    private static void testMaintenanceRequestPositive(){
        Vehicle__c vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;
```

```
        Product2 equipment = createEq();
        insert equipment;
        id equipmentId = equipment.Id;

        case somethingToUpdate =
createMaintenanceRequest(vehicleId,equipmentId);
        insert somethingToUpdate;

        Equipment_Maintenance_Item__c workP =
createWorkPart(equipmentId,somethingToUpdate.id);
        insert workP;

        test.startTest();
        somethingToUpdate.status = CLOSED;
        update somethingToUpdate;
        test.stopTest();

        Case newReq = [Select id, subject, type, Equipment__c,
Date_Reported__c, Vehicle__c, Date_Due__c
                       from case
                       where status =:STATUS_NEW];

        Equipment_Maintenance_Item__c workPart = [select id
                                                  from
Equipment_Maintenance_Item__c
                                                  where
Maintenance_Request__c =:newReq.Id];

        system.assert(workPart != null);
        system.assert(newReq.Subject != null);
        system.assertEquals(newReq.Type, REQUEST_TYPE);
        SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
        SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
        SYSTEM.assertEquals(newReq.Date_Reported__c,
system.today());
    }
```

```
    @istest
    private static void testMaintenanceRequestNegative(){
        Vehicle__C vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        product2 equipment = createEq();
        insert equipment;
        id equipmentId = equipment.Id;

        case emptyReq =
createMaintenanceRequest(vehicleId,equipmentId);
        insert emptyReq;

        Equipment_Maintenance_Item__c workP =
createWorkPart(equipmentId, emptyReq.Id);
        insert workP;

        test.startTest();
        emptyReq.Status = WORKING;
        update emptyReq;
        test.stopTest();

        list<case> allRequest = [select id
                                 from case];

        Equipment_Maintenance_Item__c workPart = [select id
                                                  from
Equipment_Maintenance_Item__c
                                                  where
Maintenance_Request__c = :emptyReq.Id];

        system.assert(workPart != null);
        system.assert(allRequest.size() == 1);
    }

    @istest
```

```
    private static void testMaintenanceRequestBulk(){
        list<Vehicle__C> vehicleList = new list<Vehicle__C>();
        list<Product2> equipmentList = new list<Product2>();
        list<Equipment_Maintenance_Item__c> workPartList = new
list<Equipment_Maintenance_Item__c>();
        list<case> requestList = new list<case>();
        list<id> oldRequestIds = new list<id>();

        for(integer i = 0; i < 300; i++){
           vehicleList.add(createVehicle());
            equipmentList.add(createEq());
        }
        insert vehicleList;
        insert equipmentList;

        for(integer i = 0; i < 300; i++){

requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
equipmentList.get(i).id));
        }
        insert requestList;

        for(integer i = 0; i < 300; i++){

workPartList.add(createWorkPart(equipmentList.get(i).id,
requestList.get(i).id));
        }
        insert workPartList;

        test.startTest();
        for(case req : requestList){
            req.Status = CLOSED;
            oldRequestIds.add(req.Id);
        }
        update requestList;
        test.stopTest();
```

```
        list<case> allRequests = [select id
                                  from case
                                  where status =: STATUS_NEW];

        list<Equipment_Maintenance_Item__c> workParts = [select
id
                                                         from
Equipment_Maintenance_Item__c
                                                         where
Maintenance_Request__c in: oldRequestIds];

        system.assert(allRequests.size() == 300);
    }
}
```

MAINTENANCE REQUEST HELPER

```
public with sharing class MaintenanceRequestHelper {
    public static void updateworkOrders(List<Case>
updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();


        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' &&
c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine
Maintenance'){
                    validIds.add(c.Id);


                }
            }
        }

        if (!validIds.isEmpty()){
```

```
            List<Case> newCases = new List<Case>();
            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT
Id, Vehicle__c, Equipment__c,
Equipment__r.Maintenance_Cycle__c,(SELECT
Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
                                                          FROM
Case WHERE Id IN :validIds]);
            Map<Id,Decimal> maintenanceCycles = new
Map<ID,Decimal>();
            AggregateResult[] results = [SELECT
Maintenance_Request__c,
MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
Equipment_Maintenance_Item__c WHERE Maintenance_Request__c IN
:ValidIds GROUP BY Maintenance_Request__c];

        for (AggregateResult ar : results){
            maintenanceCycles.put((Id)
ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }

            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
                    Subject = 'Routine Maintenance',
                    Type = 'Routine Maintenance',
                    Vehicle__c = cc.Vehicle__c,
                    Equipment__c =cc.Equipment__c,
                    Origin = 'Web',
                    Date_Reported__c = Date.Today()

                );

                If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_Due__c =
Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
                }
```

```
                newCases.add(nc);
            }

            insert newCases;

            List<Equipment_Maintenance_Item__c> clonedWPs = new
List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c wp :
closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c wpClone =
wp.clone();
                    wpClone.Maintenance_Request__c = nc.Id;
                    ClonedWPs.add(wpClone);

                }
            }
            insert ClonedWPs;
        }
    }
}
```

MAINTENANCE REQUEST

```
trigger MaintenanceRequest on Case (before update, after update)
{
    if(Trigger.isUpdate && Trigger.isAfter){
        MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
Trigger.OldMap);
    }
}
```

WAREHOUSE CALLOUT SERVICE

```
public with sharing class WarehouseCalloutService {

    private static final String WAREHOUSE_URL = 'https://th-
superbadge-apex.herokuapp.com/equipment';

    //@future(callout=true)
    public static void runWarehouseEquipmentSync(){

        Http http = new Http();
        HttpRequest request = new HttpRequest();

        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);



        List<Product2> warehouseEq = new List<Product2>();

        if (response.getStatusCode() == 200){
            List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            for (Object eq : jsonResponse){
                Map<String,Object> mapJson =
(Map<String,Object>)eq;
                Product2 myEq = new Product2();
                myEq.Replacement_Part__c = (Boolean)
mapJson.get('replacement');
                myEq.Name = (String) mapJson.get('name');
                myEq.Maintenance_Cycle__c = (Integer)
mapJson.get('maintenanceperiod');
                myEq.Lifespan_Months__c = (Integer)
mapJson.get('lifespan');
                myEq.Cost__c = (Decimal)
mapJson.get('lifespan');
                myEq.Warehouse_SKU__c = (String)
```

```
mapJson.get('sku');
                myEq.Current_Inventory__c = (Double)
mapJson.get('quantity');
                warehouseEq.add(myEq);
            }

            if (warehouseEq.size() > 0){
                upsert warehouseEq;
                System.debug('Your equipment was synced with the
warehouse one');
                System.debug(warehouseEq);
            }

        }
    }
}
```

WAREHOUSE CALLOUT SERVICE TEST

```
@isTest

private class WarehouseCalloutServiceTest {
    @isTest
    static void testWareHouseCallout(){
        Test.startTest();
        // implement mock callout test here
        Test.setMock(HTTPCalloutMock.class, new
WarehouseCalloutServiceMock());
        WarehouseCalloutService.runWarehouseEquipmentSync();
        Test.stopTest();
        System.assertEquals(1, [SELECT count() FROM Product2]);
    }
}
```

WAREHOUSE CALLOUT SERVICE MOCK

```
@isTest
global class WarehouseCalloutServiceMock implements
HttpCalloutMock {
    // implement http mock callout
    global static HttpResponse respond(HttpRequest request){

        System.assertEquals('https://th-superbadge-
apex.herokuapp.com/equipment', request.getEndpoint());
        System.assertEquals('GET', request.getMethod());

        // Create a fake response
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
t":false,"quantity":5,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10
0003"}]');
        response.setStatusCode(200);
        return response;
    }
}
```

WAREHOUSE SYNC SCHEDULE

```
global class WarehouseSyncSchedule implements Schedulable {
    global void execute(SchedulableContext ctx) {

        WarehouseCalloutService.runWarehouseEquipmentSync();
    }
}
```

WAREHOUSE SYNC SCHEDULE TEST

```
@isTest
public class WarehouseSyncScheduleTest {

    @isTest static void WarehousescheduleTest(){
        String scheduleTime = '00 00 01 * * ?';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new
WarehouseCalloutServiceMock());
        String jobID=System.schedule('Warehouse Time To Schedule
to Test', scheduleTime, new WarehouseSyncSchedule());
        Test.stopTest();
        //Contains schedule information for a scheduled job.
CronTrigger is similar to a cron job on UNIX systems.
        // This object is available in API version 17.0 and
later.
        CronTrigger a=[SELECT Id FROM CronTrigger where
NextFireTime > today];
        System.assertEquals(jobID, a.Id,'Schedule ');


    }
}
```