Use Case

There are two kinds of people in this world: those who would travel in a recreational vehicle (RV) and those who wouldn't. The RV community is increasing exponentially across the globe. Over the past few years, the world's largest RV rental company, HowWeRoll Rentals, has increased its global footprint and camper fleet 10 fold. HowWeRoll offers travelers superior RV rental and roadside assistance services. Their tagline is, "We have great service, because that's How We Roll!" Their rental fleet includes every style of camper vehicle, from extra large, luxurious homes on wheels to bare bones, retro Winnebagos. If you're plagued with wanderlust, they have the cure!

You have been hired as the lead Salesforce developer to automate and scale HowWeRoll's reach. For travelers, not every journey is champagne wishes and caviar dreams. Unfortunately, there's bound to be a bump in the road at some point along the way. Thankfully, HowWeRoll has an amazing RV repair squad who can attend to any maintenance request, no matter where you are. These repairs address a variety of technical difficulties, anything from a broken axle to a seemingly futile battle against a clogged septic tank.

As the company grows, so does HowWeRoll's rental fleet. Great for business, but the current service and maintenance process is challenging to scale. In addition to service requests for broken or malfunctioning equipment, routine maintenance requests for vehicles have grown exponentially. Without routine maintenance checks, the rental fleet is susceptible to avoidable breakdowns.

That's where you come in! HowWeRoll needs you to automate their routine Salesforce-based maintenance system. You'll ensure that nothing falls through the cracks and causes unnecessary damage to the vehicle, or worse, endangers the customer! You'll also integrate Salesforce with HowWeRoll's back-office system that keeps track of warehouse inventory. This completely separate system needs to sync on a regular basis with Salesforce. Synchronization ensures that HQ knows exactly how many spare parts are available when making a service request, and alerts them when they need to reorder equipment parts.

Standard Objects

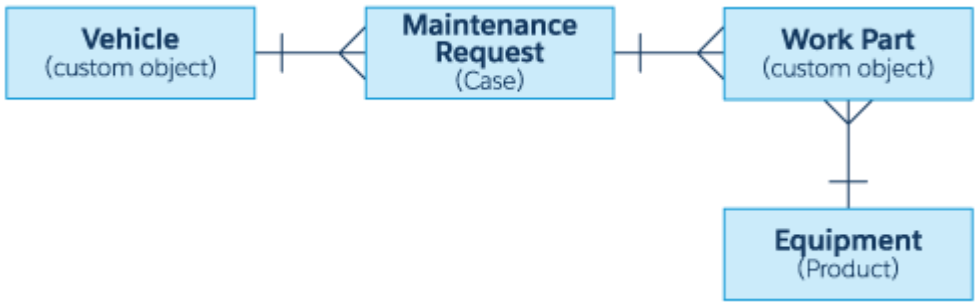
You'll be working with the following standard objects:

- **Maintenance Request (renamed Case) —** Service requests for broken vehicles, malfunctions, and routine maintenance.
- **Equipment (renamed Product) —** Items in the warehouse used to fix or maintain RVs. Repair parts have a flag indicating that they are for repair.

Custom Objects

- **Vehicle —** Vehicles in HowWeRoll's rental fleet. These records contain all stock parts that will eventually need replacement or maintenance.
- **Work Part —** Equipment used for a maintenance request.

Entity Diagram



Business Requirements

This section represents the culmination of your meetings with key HowWeRoll stakeholders. It's your blueprint to programmatically automate the support and maintenance side of their business.

Automate Maintenance Requests

With the exponential increase in RV popularity worldwide, HowWeRoll is supplying hundreds more luxury and economy vehicles around the globe. Along with this increase in their rental stock comes an inevitable increase in equipment failure. HowWeRoll has an existing process to handle these failures, but what they want you to do is build in automation for their routine maintenance. One hard truth is that all parts eventually need servicing. You'll build a programmatic process that automatically schedules regular checkups on these parts based on the date that the equipment and parts were installed.

When an existing maintenance request of type Repair or Routine Maintenance is Closed, you create a new maintenance request for a future routine checkup. This new request is tied to the same vehicle and piece of equipment to be serviced as the original closed service request. This new request's Type should be set as Routine Maintenance. The Subject should not be null and the Report Date field reflects the day the request was created. Another aspect about parts is they all have different lifespans. Therefore, you need to calculate and set the next due date using the maintenance cycle defined on the related work part records. If multiple work parts are used in the maintenance request, choose the shortest maintenance cycle to define the service date.

Design this automated process to work for both single maintenance requests and bulk requests. Bulkify the system to successfully process approximately 300 records of offline maintenance requests that are scheduled to import together. For the time being, you don't need to worry about changes that occur on the equipment record itself.

The logic needs to be exposed for other uses in the org, so separate the trigger (named **MaintenanceRequest**) from the application logic in the handler (named **MaintenanceRequestHelper**). This setup makes it easier to delegate actions and simpler to extend the app in the future.

Synchronize Inventory Management

In addition to equipment maintenance, also automate HowWeRoll's inventory data synchronization with the external system within the service parts warehouse. Although the entire HQ office is using Salesforce, the warehouse team is still working on a separate legacy system. With this integration, the inventory in Salesforce updates after parts are taken from the warehouse to service a vehicle.

Write a 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. Beyond inventory, you also ensure that other potential warehouse changes carry over to Salesforce. Your class maps the following fields: replacement part (always true), cost, current inventory, lifespan, maintenance cycle, and warehouse SKU. Use the warehouse SKU as the external ID to identify which equipment records to update within Salesforce.

Although HowWeRoll is an international company, the remote offices follow the lead of the HQ's work schedule. Therefore, all maintenance requests are processed during HQ's normal business hours. You need to update Salesforce data during off hours (at 1:00 AM PST). This logic runs daily so that the inventory is up to date every morning at HQ.

Create Unit Tests

Like any good developer, you test your code to ensure that it's bulletproof before deploying it to production.

The first thing to test is your trigger to ensure that it works as expected. Again, you follow best practices by testing for both positive use cases (when the trigger needs to fire) and negative use cases (when the trigger must not fire). Of course, you know passing a test doesn't necessarily mean you got it all right. So add assertions into your code to ensure that you aren't getting false positives. For your positive test, assert that everything was created correctly, including the relationships to the vehicle and equipment, as well as the due date. For your negative test, assert that no work orders were created.

As mentioned previously, the huge wave of maintenance requests could potentially be loaded at once. The number will probably be around 200, but to account for potential spikes, pad your class to successfully handle at least 300 records. To test this, include a positive use case for 300 maintenance requests and assert that your test ran as expected.

When you have 100% code coverage on your trigger and handler, write test cases for your callout and scheduled Apex classes. You need to have 100% code coverage for all Apex in your org.

Ensure that your code operates as expected in the scheduled context by validating that this executes after Test.stopTest() without exception. Also assert that a scheduled asynchronous job is in the queue. The test classes for the callout service and scheduled test must also have 100% test coverage.