2023

LendInvest

SALESFORCE ENGINEER CODING CHALLENGE

SAPARIA, KUSH

Contents

Requirement 1

Scenario

The LendInvest Finance team has been working on a way to classify customers into different tiers based on their Total Customer Spend. The numbers they have defined will offer them a key metric for Loyalty programs as well as for business development purposes. This metric should be displayed at Account level.

Total Customer Spend is defined as the Sum of the Amount of closed-won opportunities for an Account. On Account, the finance team need to see both the amount of Total Customer Spend (in a new custom field), and an additional field - “Customer classification”- which classifies the customer as follows:

- GOLD = Total Customer Spend of £50,000 and above,
- SILVER = Total Customer Spend of £25,000 and above,
- BRONZE = Total Customer Spend of £10,000 and above,
- <blank> = Total Customer Spend is less than £10,000, leave the field blank.

Approach Taken

Created Components

Component Name	Metadata Type	Description
Total_Customer_Spend__c	Account > Field > Roll-up Summary	Sum of all the Closed Won Opportunities related to the Account
Customer_Classification__c	Account > Field > Formula	Classifies the Customer based on the Customer's total amount spent

Usage

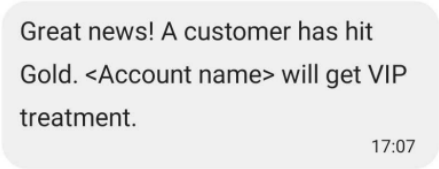
Total Customer Spend is calculated by aggregating and summing the *Amount* from all *Closed Won* Opportunities related to that account. The *Customer Classification* field uses this value to assign the appropriate classification set in the Scenario.

Requirement 2

Scenario

LendInvest's CEO needs to know as soon as possible when a customer reaches Gold status. When an account reaches Gold status, they would like to receive an SMS to their business mobile. This should be completed using the following requirements:

- This notification should only run when the customer reaches gold status for the first time.
- The SMS should read as follows:



- The CEO's number changes regularly, so it should be easily configurable by a Salesforce administrator.
- This should be done using Apex and the Twilio Rest API (do not use the Twilio for Salesforce package or any other managed package - we want to see your Apex integration skills at work!)
- Please follow any other best practices you can think of to deliver this requirement.

Approach Taken

Created Components

Component Name	Metadata Type	Description
twilioUtils.cls	Apex Class	Utility Class used for interacting with Twilio REST API
twilioSMSGold.cls	Apex Class	Class for sending SMS message to the CEO when GOLD classification is reached on Accounts
TwilioSMS	Custom Settings (Custom Object)	Used to store Twilio API credentials and the CEO's phone number (receiver of GOLD classification alerts)
Twilio	Remote Site Settings	Allows Salesforce to invoke the Twilio REST API
accountTwilioSendSMSGold	Flow	Triggers the Twilio SMS Gold Alert when an account has been updated to meet the GOLD classification

Usage

When the *Customer_Classification__c* field is updated to 'GOLD', the *accountTwilioSendSMSGold* flow is triggered which uses *twilioSMSGold.cls* and *twilioUtils.cls* to send a text message to the phone number stored in the custom setting field *GoldAlertNumber__c*. Storing the number in a custom setting means it is very easy for administrators to update.

Requirement 3

Scenario

Our Salesforce internal users want to have the ability to send a custom SMS to any gold customer via the customer's Account page. To achieve this, we would like you to build a Lightning Web Component.

- Again, leverage the Twilio API + Apex to achieve this task
- Users need be able to input free text into an input box on the Account page and hit a "send" button to send the text to the customer.
- The LWC should only be available for Gold customer accounts

Approach Taken

Created/Modified Components

Component Name	Metadata Type	Description
twilioSendCustomSMSGold	Lightning Web Component	Allows front-end users to send custom text messages to GOLD Accounts
twilioSendCustomSMSGoldController.cls	Apex Class (Controller)	Controller component for twilioSendCustomSMSGold the LWC. Allows LWC to interact with server-side to send POST request to Twilio
Account Record Page	Account > Lightning Page Layout	New lightning page layout with the added LWC component
Enforce_E164	Account > Validation Rules	Enforces the E.164 format for phone numbers

Usage

A new LWC, *twilioSendCustomSMSGold*, is available on the Lightning App Account Record Page. The LWC is only visible when the customer classification is GOLD (via Component Visibility). This LWC combined with the controller and *twilioUtils.cls* is used to send text messages to customers.

The Account MUST have a valid phone number (formatted in E.164) for the LWC to allow the sending of messages and the message cannot be blank. A blank phone number will result in the component showing an error message. The validation rule on the *Phone* field is to enforce the E.164 standard. This standard is enforced so that the Twilio REST API can work properly.

Given more time (and depending on the API limits), we could use Twilio's validation and formatting tools from their Lookup API to validate phone numbers. This is a more robust solution because there are a range of numbers in the E.164 format that are still invalid numbers and Twilio can flag them. However, this solution comes at the cost of increased complexity and API callouts.

Unit Tests

Component Name	Metadata Type	Description
TwilioUtilsTest.cls	Apex Class (Unit Test)	Testing Class for Twilio related classes

The unit tests ensure 100% code coverage BUT can be more meaningful. They ensure that there are no errors with the variables that are passed in. However, given more time, there is still room for

improvement. The responses from invoking some of the callouts are not included in any assertions because the callout is a future request and returns *void*. We can improve the tests by creating a custom object for logging responses and then examine the records that are stored as this object type after the test stops. This will allow us to properly examine the responses from invoking Twilio API. It could also be used to store inbound messages from when the customer responds to the text (chat log), which we could display to the front-end user.