coResolute

# ADVANCED REAL TIME USER SENTIMENT ANALYSIS WITH SALESFORCE & OPENAI INTEGRATION

salesforce

+

OpenAI

BY: CORESOLUTE LLC

Salesforce brings companies and customers together. It unites company marketing, sales, commerce, service, and IT teams from anywhere with Customer 360 — one integrated CRM platform that powers our entire suite of connected apps. With Salesforce, you can focus your employees on what's important right now: stabilizing your business, reopening, and getting back to delivering exceptional customer experiences.

**OpenAI**

OpenAI is an American artificial intelligence research laboratory consisting of the non-profit with investments from Microsoft and other premium companies. OpenAI Incorporated and its for-profit subsidiary corporation OpenAI Limited Partnership. OpenAI conducts AI research to promote and develop friendly AI in a way that benefits all humanity.

## SOLUTION SUMMARY

You always want to do an effective customer success planning for your customers.

The automation here helps achieve it seamlessly through Salesforce hyper automation and integrating OpenAI capabilities with below three ways,

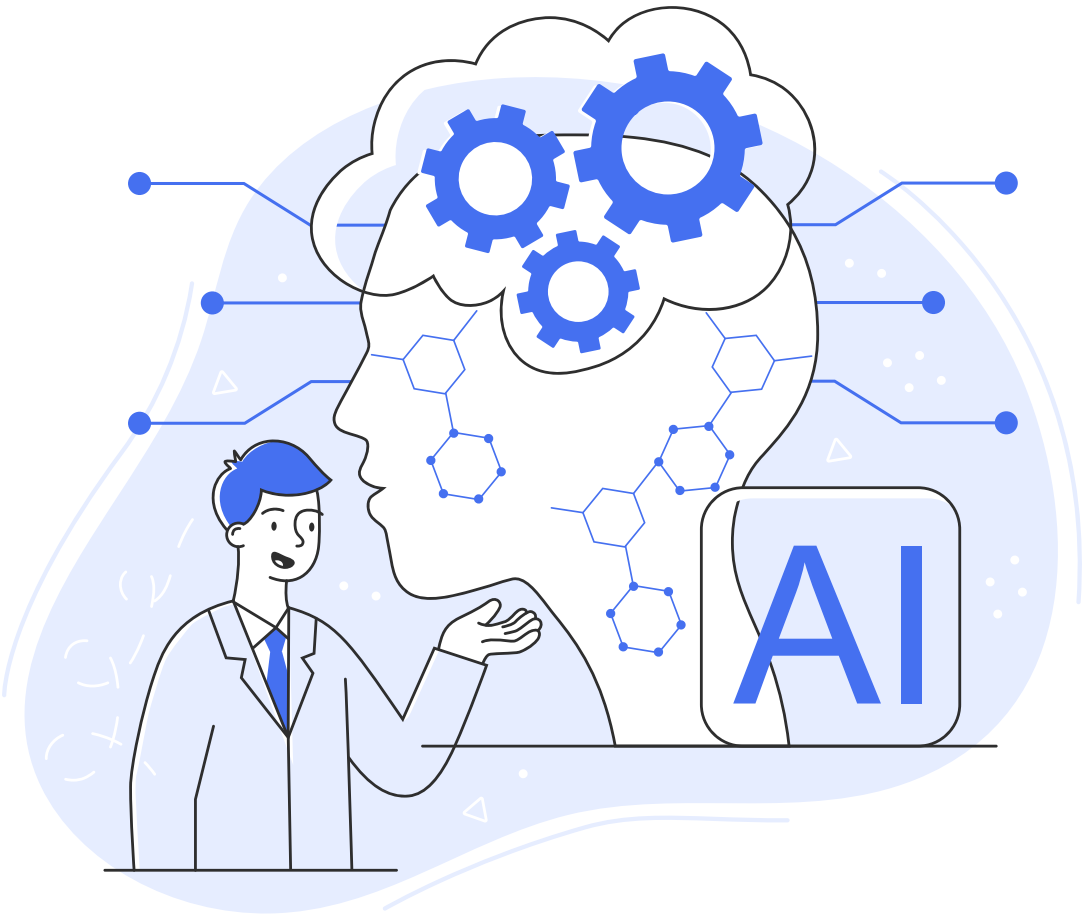
You get real time customer sentiment like Positive, Neutral, Negative or Frustrated by reviewing customer case/ incident description with OpenAI sentiment analysis.

An effective summary from long case description is automatically created to quickly grasp and get a gist of customer challenge in hand.

An automated call to action activity and notifications are generated in Salesforce for account managers when customer sentiment is negative/frustrated prompting them take a quick action.

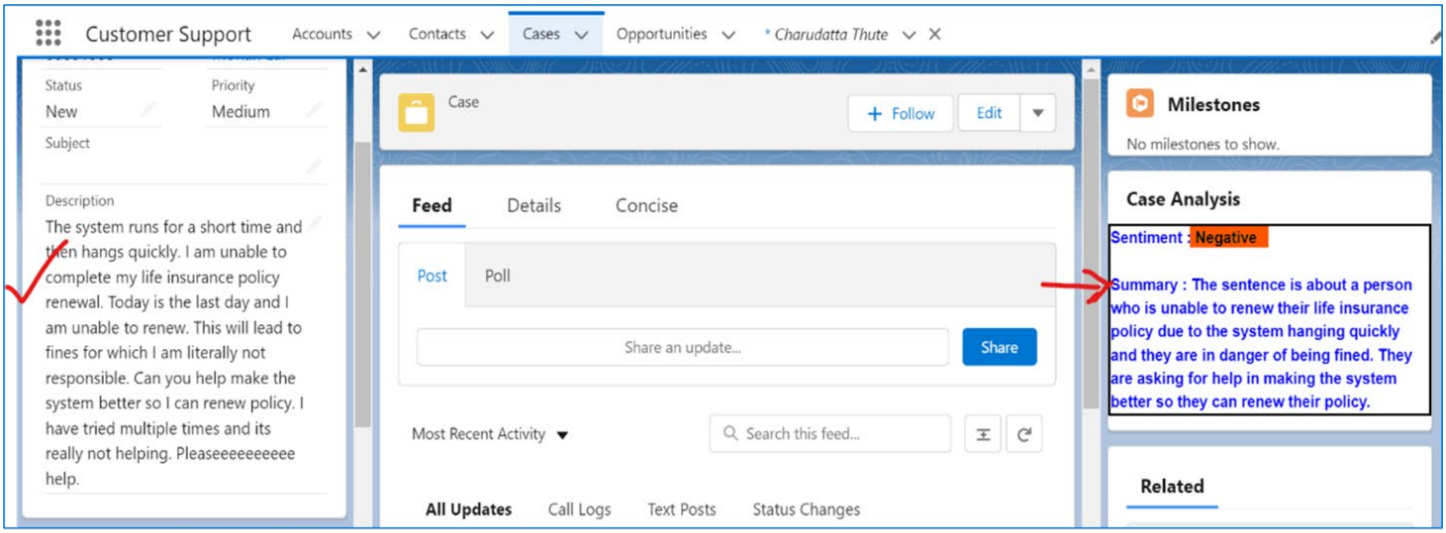
The entire automation makes company plan a proactive approach for customer management and results in effective customer success planning.

CoResolute helps bring this customer success with AI capabilities to your Salesforce Org for a fraction of Einstein cost.




## Bringing Salesforce OpenAI integration for Salesforce Cases Sentiment analysis and content Summarization:

The OpenAI text analyzer and summarization APIs are integrated with Salesforce case management for better user sentiment tracking through cases.

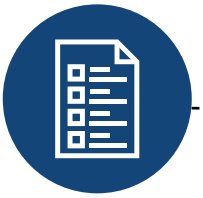



Here based on *Case Description* the *user sentiment & case summary* is automatically analyzed with developed AI capabilities in Salesforce.

# 3 STEPS FUNCTION

The sentiment is automatically analyzed with artificial Intelligence in backend and shows if a person has Positive, Neutral, Negative or Frustrated sentiment in the Case/ incident raised.

The concise Summary is also done from lengthy description for quick reference with the AI analysis.



An automated call to action is generated in Salesforce to account managers where negative or frustrated sentiment is recorded, helping companies do quick and effective customer success planning.

# POTENTIAL BENEFITS

- Improved real time customer insights.
- Enhanced decision making, reduced manual work and level of engagement plan.
- Better customer success planning
- Effective customer support planning by customer representatives
- Better Marketing operations planning for Happy vs Not so Happy customers.
- Efficient handling of grievances and cases based on sentiment. Can plan appropriate SLA's
- Better reporting
- Better Sales planning based on customer sentiment tracking.
- Substantial cost savings with Open AI than using native Salesforce Einstein AI

# TECHNICAL DEVELOPMENT

**Objects used**

- Case

**Custom fields**

- Case>Sentiment
- Case>Summary

**APIs used.**

- OpenAI (Services: Tweet Classifier, Completions)

**Lightning Web Components developed**

- chatgptCompo
- Here is code:

## Apex Classes

1. **CaseDescResponse:**
   - The class in Salesforce org, helps specify what type of description user provides? Is it Positive, Neutral, Negative or Frustrated?
   - It also helps get sentiment summary as well in lightning component. It will summarize the Sentiment.
   - For Sentiment Analysis we refer Case Description field.
   - In sentimentAnalysisMethod function, we retrieve the Case Description field from Salesforce and make API call to Tweet Classifier where we set Endpoint as 'Completions'.
   - We put Prompt here to decide whether Case Description is Positive, Negative or Neutral and set that prompt as body.
   - Similarly, in tldrSummaryMethod function, we retrieve the Case Description field from Salesforce and make API call to Tweet Classifier where we set Endpoint as 'Completions'.
   - In this function we use different prompt 'briefly summarize the sentence' which will summarize the sentiment in brief.

**Below is code for Sentiment Analysis**

```
public with sharing class CaseDescResponse {
    @AuraEnabled
  public static string sentimentAnalysisMethod(List<Id> caseIds){
    String sentiment ='';
    try {
    // Retrieve the cases
   for(Case objCase : [SELECT Id, Description FROM Case WHERE Id IN  :caseIds]){
        // Make the API call to the tweet classifier
      Http http = new Http();
      HttpRequest request = new HttpRequest();
      request.setEndpoint('https://api.openai.com/v1/completions');
      request.setMethod('POST');
      request.setHeader('Authorization', 'Bearer sk-Provide Bearer Id');
      request.setHeader('OpenAI-Organization', 'org-provide Org Id');
      request.setHeader('Content-Type','application/json');
      // The prompt is the text that the model will complete
      String prompt = 'Decide whether a sentence sentiment is positive, neutral,Frustrated or negative' + objCase.Description;
          // Additional parameters can be added as needed
      String body = '{"prompt": "' + prompt + '", "model": "text-davinci-003","temperature": 0, "max_tokens": 60,"top_p": 1, "frequency_penalty": 0, "presence_penalty": 0}';
      request.setBody(body);
        HttpResponse response = http.send(request);
     System.debug('response ' + response);
      if (response.getStatusCode() == 200) {
        // Parse the JSON response
        Map<String, Object> jsonResponse = (Map<String, Object>) JSON.deserializeUntyped(response.getBody());
        List<Object> completions = (List<Object>) jsonResponse.get('choices');
```

```
            Map<String, Object> completion = (Map<String, Object>) completions[0];
             sentiment = (String) completion.get('text');

        } else {
            // Handle error
            System.debug('Error: ' + response.getStatusCode() + ' ' + response.getStatus());
            //return 'Error: ' + response.getStatusCode() + ' ' + response.getStatus();
        }
       }
     } catch (Exception e) {
       System.debug('### ' + e.getMessage());
     }
     return sentiment;
   }
      @AuraEnabled
   public static string tldrSummaryMethod(List<Id> caseId){
            String label='';
        try {
            // Retrieve the cases
            for(Case objCase : [SELECT Id, Description FROM Case WHERE Id IN : caseId]){

            // Make the API call to the summary
            Http http = new Http();
            HttpRequest request = new HttpRequest();
            request.setEndpoint('https://api.openai.com/v1/completions');
            request.setMethod('POST');
            request.setHeader('Authorization', 'Bearer sk-B5avLEicDIZNp5zLpTcLT3BlbkFJzSp1g5M3ARD0ox-6nl58E');
            request.setHeader('OpenAI-Organization', 'org-96p4npIEq7HHtOBwwDIHadSS');
            request.setHeader('Content-Type','application/json');
```

```
            // The prompt is the text that the model will complete
            String prompt = 'briefly summarize the sentence' + objCase.Description;

            // Additional parameters can be added as needed
            String body = '{"prompt": "' + prompt + '", "model": "text-davinci-003","temperature": 0.7, "max_tokens": 160,"top_p": 1.0, "frequency_penalty": 0.0, "presence_penalty": 1}';
            request.setBody(body);
            HttpResponse response = http.send(request);

                    if (response.getStatusCode() == 200) {
                // Parse the JSON response
            Map<String, Object> json = (Map<String, Object>) JSON.deserializeUntyped(response.getBody());
                System.debug('json '+ json);
            List<Object> completions = (List<Object>) json.get('choices');
            Map<String, Object> firstCompletion = (Map<String, Object>) completions[0];
            label = (String) firstCompletion.get('text');

            }
            else {
                // Handle error
                System.debug('Error: ' + response.getStatusCode() + ' ' + response.getStatus());
                //return 'Error: ' + response.getStatusCode() + ' ' + response.getStatus();
            }
          }
       } catch (Exception e) {
          System.debug(e.getMessage());
       }
       return label;
      }
}
```

## Mock Classes

For testing our Api callout methods we would require MockClasses first.

Below are Mock Classes for Sentiment Analysis Testing:

1. CaseDescResopnseMockClass:

Below is code for CaseDescResopnseMockClass:

```
@isTest
global class CaseDescResponseMockClass implements HttpCalloutMock{
    global HttpResponse respond (HttpRequest req) {
    HttpResponse response = new HttpResponse();
    response.setHeader('Content-Type', 'application/json');
   // response.setBody('{"choices": [{"text": "positive"}]}');
    response.setBody('{"choices": [{"text": "Positive"},{"text": "Negative"},{"text": "Neutral"},{"text": "Frustrated"}]}');
    // {choices= ({finish_reason=stop, index=0, logprobs=null, text=The support departmentg.}), created=1675496183, id=cmpl-6g7NvgXYWx0PqzpXPMF2SWOkXANzJ, model=text-davinci-003, object=text_completion, usage= {completion_tokens=28, prompt_tokens=47, total_tokens=75}}
    response.setStatusCode(200);
    system.debug('#Response='+response);
    return response;
  }
}
```

2. CaseDescResponseMockClass1:

Below is code for CaseDescResopnseMockClass1:

```
@isTest
global class CaseDescResponseMockClass1 implements HttpCalloutMock{
  global HttpResponse respond(HttpRequest req) {
    HttpResponse res = new HttpResponse();
    res.setHeader('Content-Type', 'application/json');
    res.setBody('{"choices":[{"text":"Brief summary"}]}');
```

```
        res.setStatusCode(200);
        system.debug('#Response='+res);
        return res;
    }
}
```

- Test Classes

For testing apex class, we created test class.

Below is test class for apex classes:

1. CaseDescResponseTestClass:

Code for CaseDescResponseTestClass:

```
@isTest
private class CaseDescResponseTestClass {
    @testSetup
    static void setup(){
        Case c = new Case(Description='This is a positive sentiment');
        c.Priority = 'Low';
        insert c;
 Case c1 = new Case(Description='Test case description');
        c1.Priority = 'High';
        insert c1;
    }

    @isTest
    static void testSentimentAnalysisMethod(){
        List<Id> caseIds = new List<Id>();
        for(Case c : [SELECT Id FROM Case WHERE Priority = 'Low']){
            caseIds.add(c.Id);
        }
```

```
        System.debug('Id '+ caseIds);
        // Set up the mock callout class
        Test.setMock(HttpCalloutMock.class, new CaseDescResponseMockClass());
    // call the method being tested
        Test.startTest();
        string result = CaseDescResponse.sentimentAnalysisMethod(caseIds);
        system.debug('#Sentiment='+result);
        // validate the result
        System.assertEquals('Positive', result,'The result is not positive');
        System.assertEquals(true, String.isNotBlank(result));
        Test.stopTest();
    }
@isTest
    static void testTldrSummaryMethod(){
        List<Id> caseIds = new List<Id>();
        // create test cases
        for(Case c : [SELECT Id FROM Case WHERE Priority = 'High']){
            caseIds.add(c.Id);
        }
        System.debug('Id '+ caseIds);
                // Set up the mock callout class
        Test.setMock(HttpCalloutMock.class, new CaseDescResponseMockClass1());
                // call the method being tested
        Test.startTest();
        String result = CaseDescResponse.tldrSummaryMethod(caseIds);
        system.debug('#Result='+result);
                // validate the result
        System.assertEquals('Brief summary', result);
        Test.stopTest();
```

Create an automated Salesforce Flow for call to action creation when Case sentiment is negative or frustrated.

## Steps:

## ABOUT CORESOLUTE

CoResolute is a CRM & Technology company headquartered in Austin, Texas,US with development resources in Pune, India. Our positioning is to streamline complex customer & revenue operations with advanced CRM expertise and custom app development.

The depth and breadth of our CRM consultancy to cover CRM lifecycle are,

- Highly specialized in broker-dealer & financial services CRM setup and implementation
- Net Zero Cloud configuration and implementation expertise
- Salesforce CRM process consultancy & journey mapping expertise
- Specialized in maintenance, integration, and licenses evaluation.
- Custom app development capability as a SF AppExchange partner

## FOR MORE DETAILS CONTACT

- Visit www.coresolute.com to know more about us.
- Jason.linkswiler@coresolute.com
- Charudatta.thute@coresolute.com