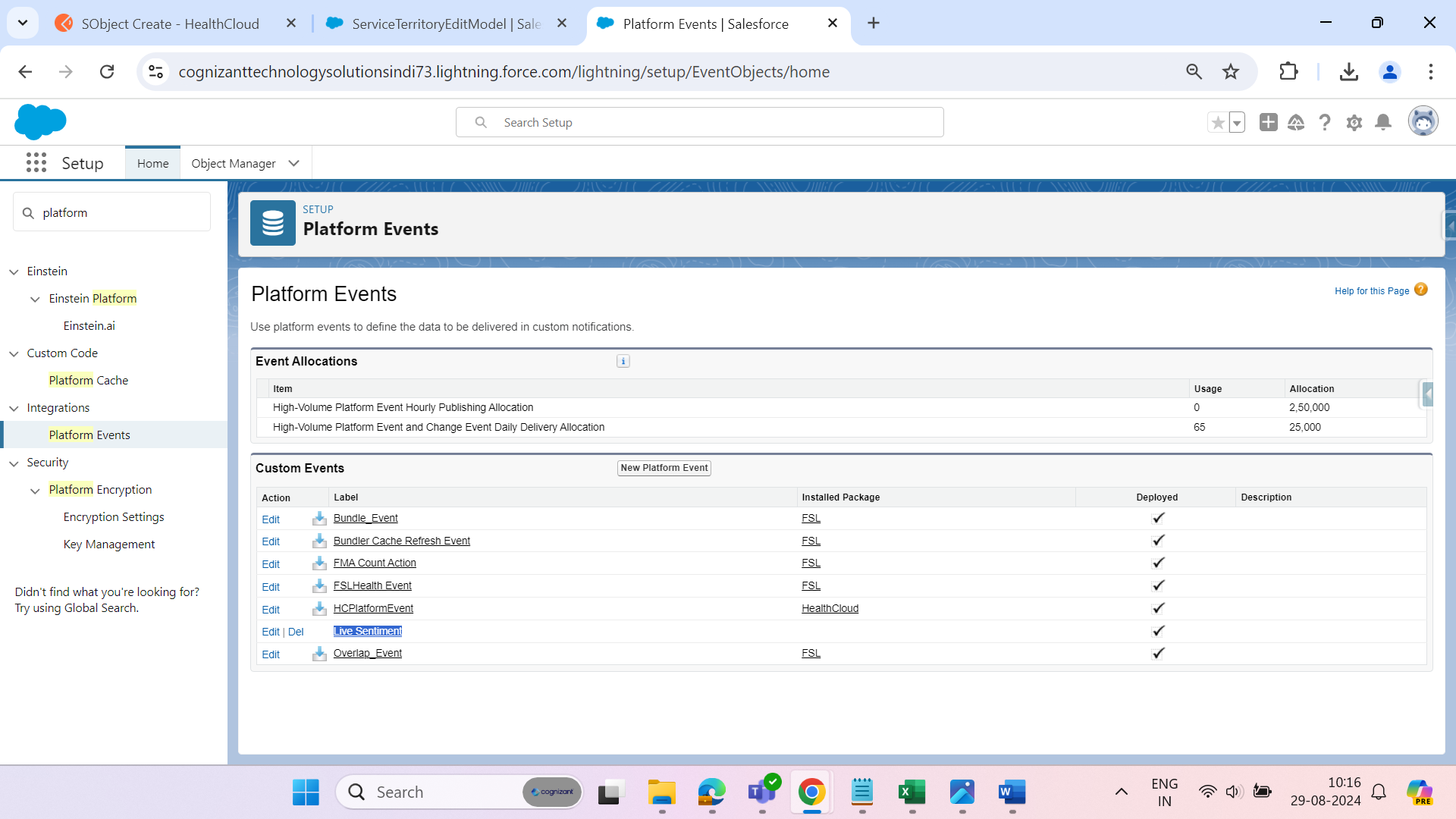
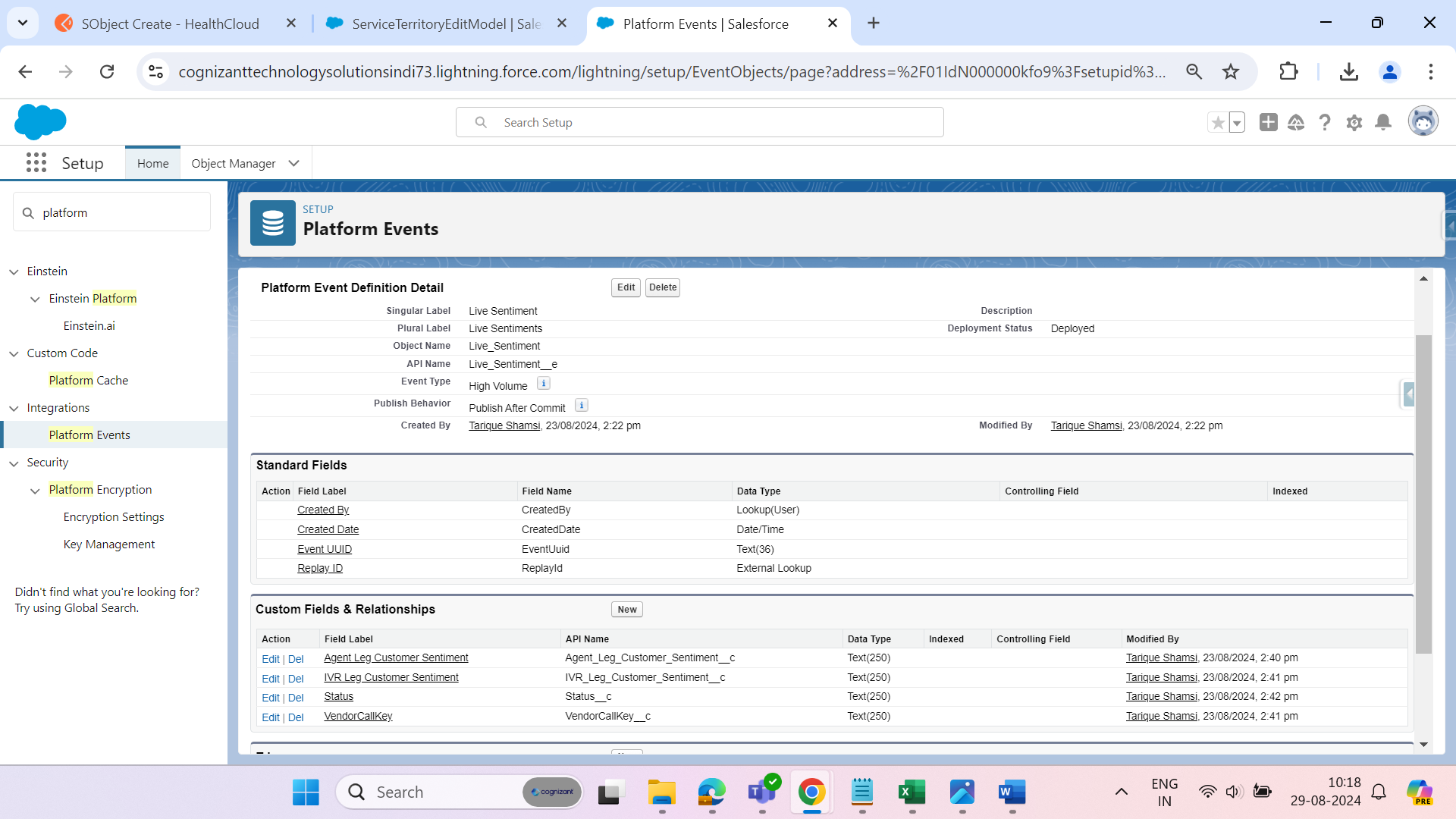
Sentiment for Health Org

Step1: Create Platform Event in Org.

Name: LiveSentiment

Step 2: Create Fields inside Platform Event



We are using 2 fields VenderCallKay\_\_c and Agent\_Leg\_Customer\_Sentiment\_\_c.

VenderCallKay\_\_c: it help to validate the Field that we are on Correct Voice Call Record.

Agent\_Leg\_Customer\_Sentiment\_\_c: This field data is Push by the Genesys team during the call

Step 3: upload Emoji In Static Resource to update the Agent\_Leg\_Customer\_Sentiment\_\_c and IVR\_Leg\_Customer\_Sentiment\_\_c.

There is total 5 Emoji.

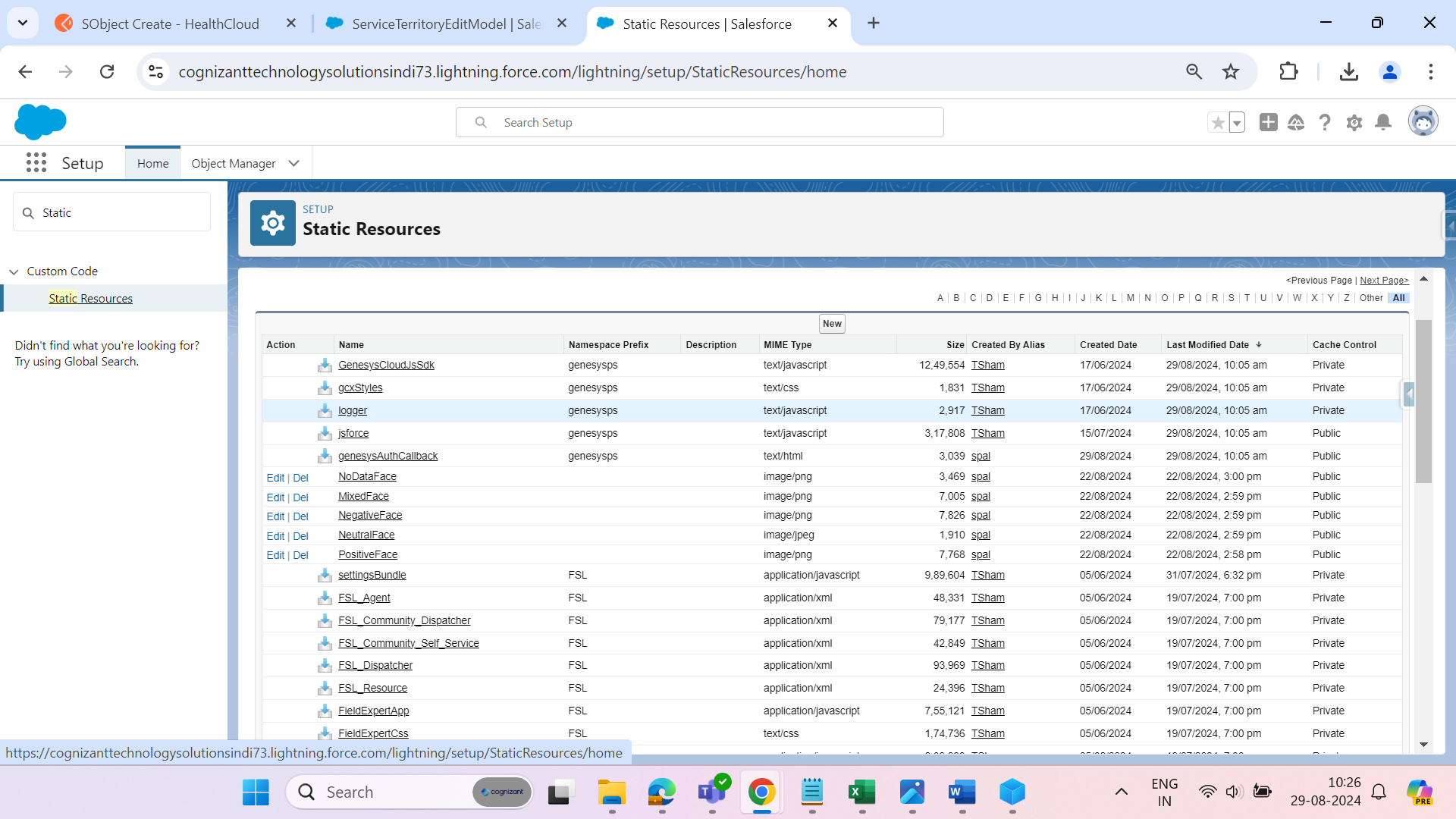
Positive

Negative

Mixed

Neutral

NoData



Step 4: Create a LWC Component

Name: realTimeSentiment

JS Code:

import { LightningElement , track, wire, api} from 'lwc';

import { getRecord } from 'lightning/uiRecordApi';

import { subscribe } from 'lightning/empApi';

import positiveEmoji from '@salesforce/resourceUrl/PositiveFace';

import neutralEmoji from '@salesforce/resourceUrl/NeutralFace';

import mixedEmoji from '@salesforce/resourceUrl/MixedFace';

import negativeEmoji from '@salesforce/resourceUrl/NegativeFace';

import noDataEmoji from '@salesforce/resourceUrl/NoDataFace';

const FIELDS= ['VoiceCall.CallDisposition','VoiceCall.VendorCallKey','VoiceCall.IVR\_Leg\_Customer\_Sentiment\_\_c'] ;

export default class RealTimeSentiment extends LightningElement {

    @track agentLegCustomerSentimentFromEvent = 'No Data Yet';    *//Agent\_Leg\_Customer\_Sentiment\_\_c to be populated from Platform Event*

    @track ivrLegCustomerSentimentFromVCR = 'No Data';    *//IVR\_Leg\_Customer\_Sentiment\_\_c from Voice Call Record*

    @track agentLegEmoji = noDataEmoji ;   *// Agent Emoji*

    @track ivrLegEmoji =noDataEmoji ;     *// Customer Emoji*

    @track vendorCallKeyFromVCR; *// VendorCallKey from Voice Call Record*

    @track statusFromVCR; *// Status from Voice Call Record*

    @track handleChangeIVR = true;   *// Conditional Rendaring for Emoji used in HTML*

    @track handleChangeAgent =true;  *// Conditional Rendaring for Emoji used in HTML*

    @track agentLegData = false;    *//  Hinding Agent Data*

    @track subscription = {};

    @api channelName = '/event/Live\_Sentiment\_\_e';

    @api recordId; *//current VoiceCall Id*

    @wire(getRecord, { recordId:'$recordId', fields:FIELDS})

    voiceCall({data,error}){

        if(data){

            console.log('BG data ', data);

            this.ivrLegCustomerSentimentFromVCR = data.fields.IVR\_Leg\_Customer\_Sentiment\_\_c.value;

            this.vendorCallKeyFromVCR = data.fields.VendorCallKey.value;

            this.statusFromVCR = data.fields.CallDisposition.value;

                if(this.statusFromVCR !== 'completed' || this.statusFromVCR ==='in-progress'){       *//  Hiding Condition for Agent*

                    this.agentLegData =true;

                    this.handleChangeAgent =true;

                    this.agentLegEmoji =noDataEmoji;

                    this.agentLegCustomerSentimentFromEvent ='No Data Yet';

                }

            if(this.ivrLegCustomerSentimentFromVCR !== null ){

                if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'POSITIVE'){

                    this.ivrLegEmoji = positiveEmoji;

                    this.handleChangeIVR = true;

                }else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEGATIVE'){

                    this.ivrLegEmoji = negativeEmoji;

                    this.handleChangeIVR = true;

                }else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'MIXED'){

                    this.ivrLegEmoji = mixedEmoji;

                    this.handleChangeIVR = true;

                }else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEUTRAL'){

                    this.ivrLegEmoji = neutralEmoji;

                    this.handleChangeIVR = true;

                }else{

                    this.ivrLegEmoji ='';

                    this.handleChangeIVR = false;

                }

            }

            else{

                this.ivrLegEmoji = noDataEmoji;

                this.ivrLegCustomerSentimentFromVCR='No Data';

                this.handleChangeIVR = true;

            }

        }

        else{

            console.error(error)

        }

    }

    connectedCallback() {

        console.log('BG in connected callback');

        this.handleSubscribe();

    }

    handleSubscribe() {

        const messageCallback = (response)=> {

            console.log('BG New Message received:', JSON.stringify(response));

            console.log('BG this.vendorCallKeyFromVCR: ', this.vendorCallKeyFromVCR);

            console.log(response['data']['payload']['VendorCallKey\_\_c']);

            if(this.statusFromVCR !== 'completed'){

                this.agentLegData =true;

                this.handleChangeAgent =true;

                if(this.vendorCallKeyFromVCR.slice(-36) == response['data']['payload']['VendorCallKey\_\_c'] && this.statusFromVCR !== 'completed') {

                    this.agentLegCustomerSentimentFromEvent = response['data']['payload']['Agent\_Leg\_Customer\_Sentiment\_\_c']

                    if(this.agentLegCustomerSentimentFromEvent !== null){

                        if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'POSITIVE'){

                            this.agentLegEmoji = positiveEmoji;

                            this.handleChangeAgent =true;

                            this.agentLegData =true;

                        }else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEGATIVE'){

                            this.agentLegEmoji = negativeEmoji;

                            this.handleChangeAgent =true;

                            this.agentLegData =true;

                        }else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'MIXED'){

                            this.agentLegEmoji = mixedEmoji;

                            this.handleChangeAgent =true;

                            this.agentLegData =true;

                        }else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEUTRAL'){

                            this.agentLegEmoji = neutralEmoji;

                            this.handleChangeAgent =true;

                            this.agentLegData =true;

                        }else{

                            this.agentLegEmoji ='';

                            this.handleChangeAgent =false;

                            this.agentLegData =true;

                        }

                    }

                    else{

                        this.handleChangeAgent =true;

                        this.agentLegData =true;

                        this.agentLegEmoji =noDataEmoji;

                        this.agentLegCustomerSentimentFromEvent ='No Data Yet';

                    }

                }

            }

            else{

                this.agentLegData = false;

            }

        };

        subscribe(this.channelName, -1, messageCallback).then(response => {

            console.log('BG Subscription request sent to: ', JSON.stringify(response.channel));

            this.subscription = response;

        });

    }

}

Note: difference between Health Org Sentiments vs Future Insurance Sentiments.

We are Comparing Vander call key in Future Insurance Org but in Health Org we are Using only a part of Vender call kay.

Full Vender call kay value:   
ac921dc3-3d5b-418d-84ef-5a2f7ef4aad0:5b7a77a7-0280-403f-aea9-07f371ba9d81

But in Health org we are using only after Colan part: 5b7a77a7-0280-403f-aea9-07f371ba9d81

Because the Genesys them need this type of requirement.

Code where we implement:

                if(this.vendorCallKeyFromVCR.slice(-36) == response['data']['payload']['VendorCallKey\_\_c'] && this.statusFromVCR !== 'completed') {

HTML and CSS part are as it is.

HTML:

<template>

    <lightning-card >

        <div *class*=*"slds-page-header slds-page-header\_record-home slds-card\_\_header"* *style*=*"height: 69px"*>

            <header *class*=*"slds-media slds-media\_center slds-has-flexi-truncate"*>

                <div *class*=*"slds-media\_\_body"*>

                    <h2 *class*=*"slds-card\_\_header-title slds-m-top\_x-small slds-m-left\_x-small"*>

                        Customer Sentiment

                    </h2>

                </div>

            </header>

        </div>

        <div *class*=*"slds-card\_\_body slds-card\_\_body\_inner"*>

            <template *if*:*true* ={*handleChangeIVR*}>

                    <p *class*=*"slds-p-horizontal\_large slds-p-left\_medium changepc"*><strong>IVR Leg &nbsp;&nbsp; :&thinsp;&thinsp;</strong> <span *class*=*"slds-text-heading\_medium"*><img *src*={*ivrLegEmoji*} *height*=*"25px"* *width*=*"25px"*/></span> ({ivrLegCustomerSentimentFromVCR})</p>

            </template>

            <template *if*:*false* ={*handleChangeIVR*}>

                    <p *class*=*"slds-p-horizontal\_large slds-p-left\_medium changepc"*><strong>IVR Leg &nbsp;&nbsp;&thinsp; :&thinsp;&thinsp;</strong> <span *class*=*"slds-text-heading\_medium"*></span>({ivrLegCustomerSentimentFromVCR})</p>

            </template>

        </div>

        <div *class*=*"slds-card\_\_body slds-card\_\_body\_inner"*>

            <template *if*:*true* ={*agentLegData*}>

                <template *if*:*true* ={*handleChangeAgent*}>

                        <p *class*=*"slds-p-horizontal\_large slds-p-left\_medium changepc"*><strong>Agent Leg: </strong> <span *class*=*"slds-text-heading\_medium"*><img *src*={*agentLegEmoji*} *height*=*"25px"* *width*=*"25px"*/></span> ({agentLegCustomerSentimentFromEvent})</p>

                </template>

                <template *if*:*false* ={*handleChangeAgent*}>

                        <p *class*=*"slds-p-horizontal\_large slds-p-left\_medium changepc"*><strong>Agent Leg: </strong> <span *class*=*"slds-text-heading\_medium"*></span>&thinsp;({agentLegCustomerSentimentFromEvent})</p>

                </template>

            </template>

            <template *if*:*false* ={*agentLegData*}>

                <p><br><br></p>

            </template>

        </div>

    </lightning-card>

</template>

CSS:

.dummy{

*color :* *black*;

}

.slds-card\_\_header {

*background-color:* *#f3f3f3*;

*top:* *-25px*;

*margin-top:* *-30px*;

*border-style:* *outset*;

*font-weight:* *bold*;

}

.slds-card\_\_body{

*position:* *relative*;

*top:* *-8px*;

}

.slds-p-horizontal\_large{

*margin-left:* *-20px*;

}

.slds-page-header{

*position:* *relative*;

*top:* *5px*;

}

.slds-card\_\_body{

*position:* *relative*;

*top:* *-1px*;

}

Platform Event:

Platform Events exchange event data in real-time within the Salesforce platform, and between Salesforce and external platforms. Platform Events are a Salesforce-native feature that work like this:

1.Event producer: An event producer creates an event.

2.Event bus: The event gets added onto the event bus (aka channel), which operates as a queue, with a strict chronological order, and executes each event one after the other.

3.Event consumers: Event consumers subscribe to an event. The moment that event gets put onto the event bus, the event consumer will be notified.

Limitation:

Platform events cannot be queried through SOQL or SOSL.

Platform events cannot be used in reports, list views, and search.

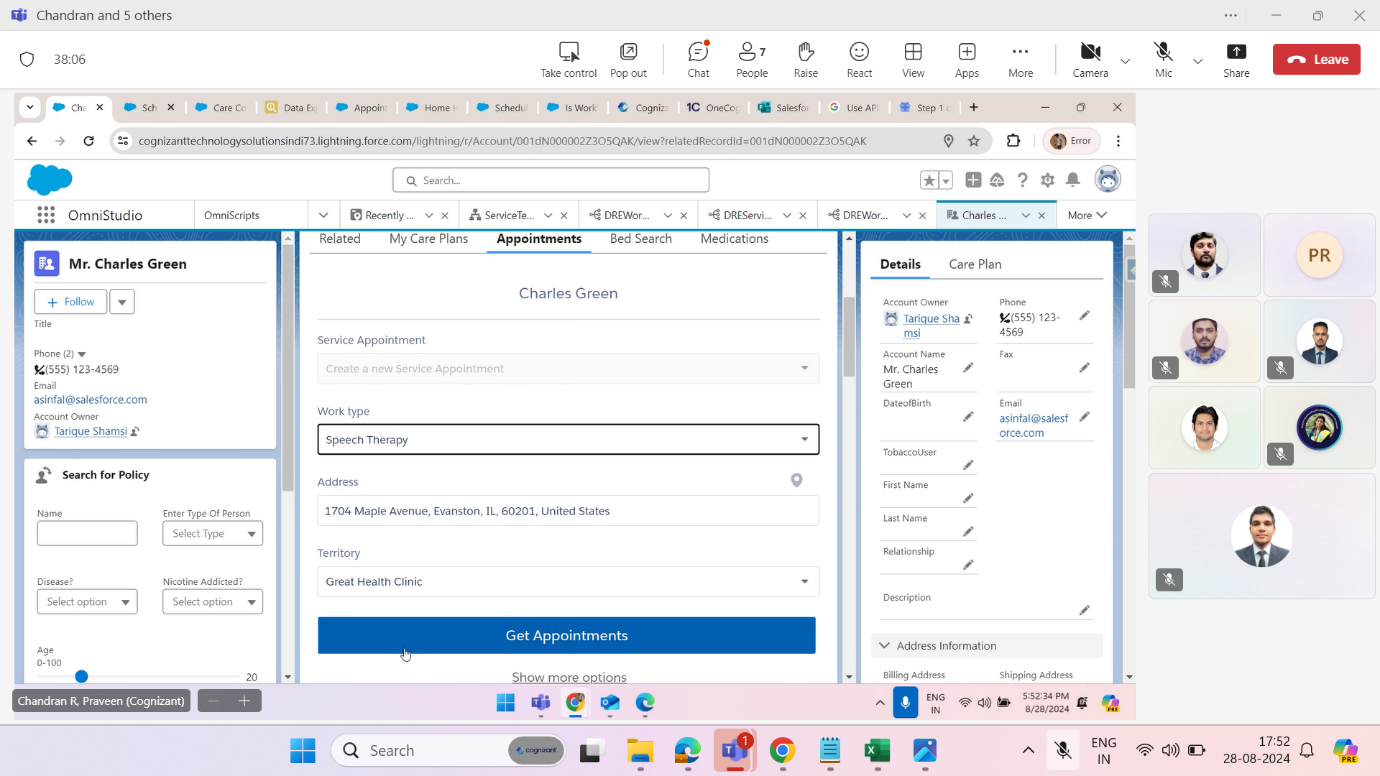
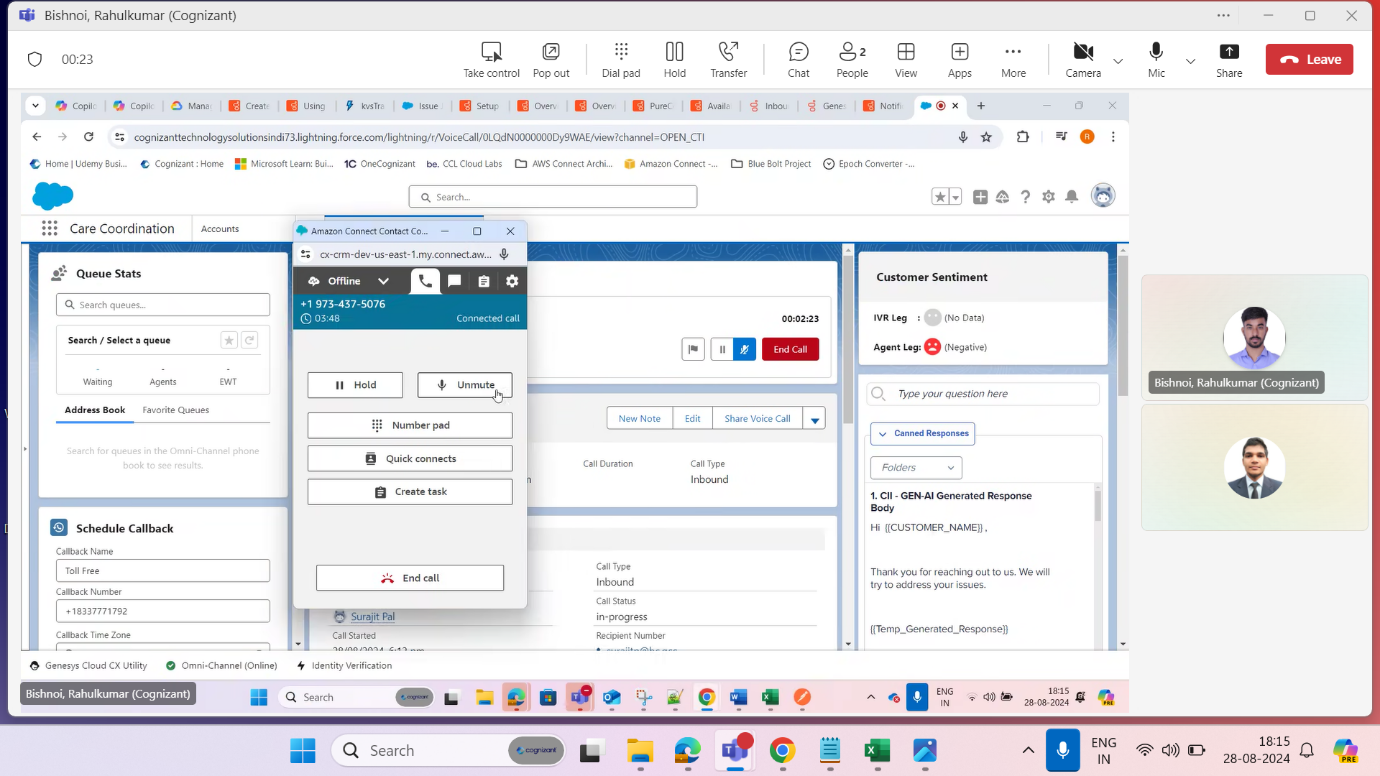
Platform events do not have a related tab.

Distributed Platform events cannot be rolled back.

All platform event fields are read-only by default.

Final output: A screenshot of a computer

Description automatically generated



Note: What value pass from the Postman.

{

"VendorCallKey\_\_c": "5b7a77a7-0280-403f-aea9-07f371ba9d81",

"Agent\_Leg\_Customer\_Sentiment\_\_c": "Positive"

}

Or if we want to test it from Developer console, we have to pass this value.

Live\_Sentiment\_\_e Ls = new Live\_Sentiment\_\_e(  
    Agent\_Leg\_Customer\_Sentiment\_\_c= 'Positive’,  
    VendorCallKey\_\_c=’5b7a77a7-0280-403f-aea9-07f371ba9d81’);  
EventBus.publish(Ls);

