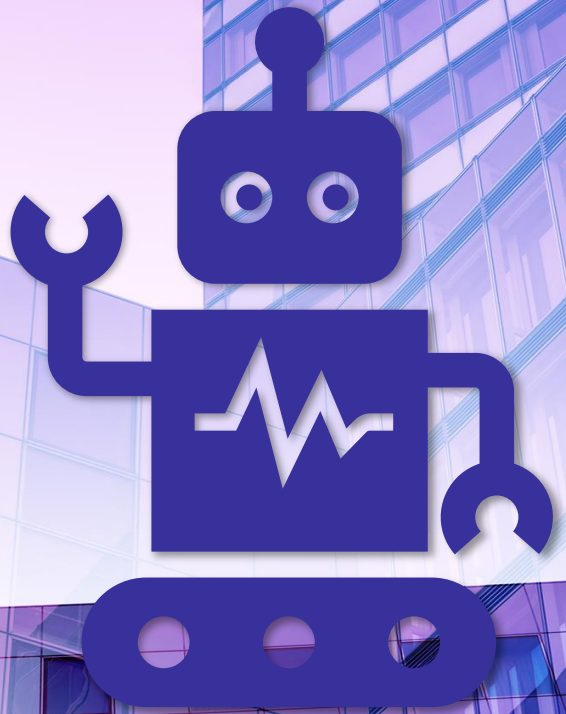
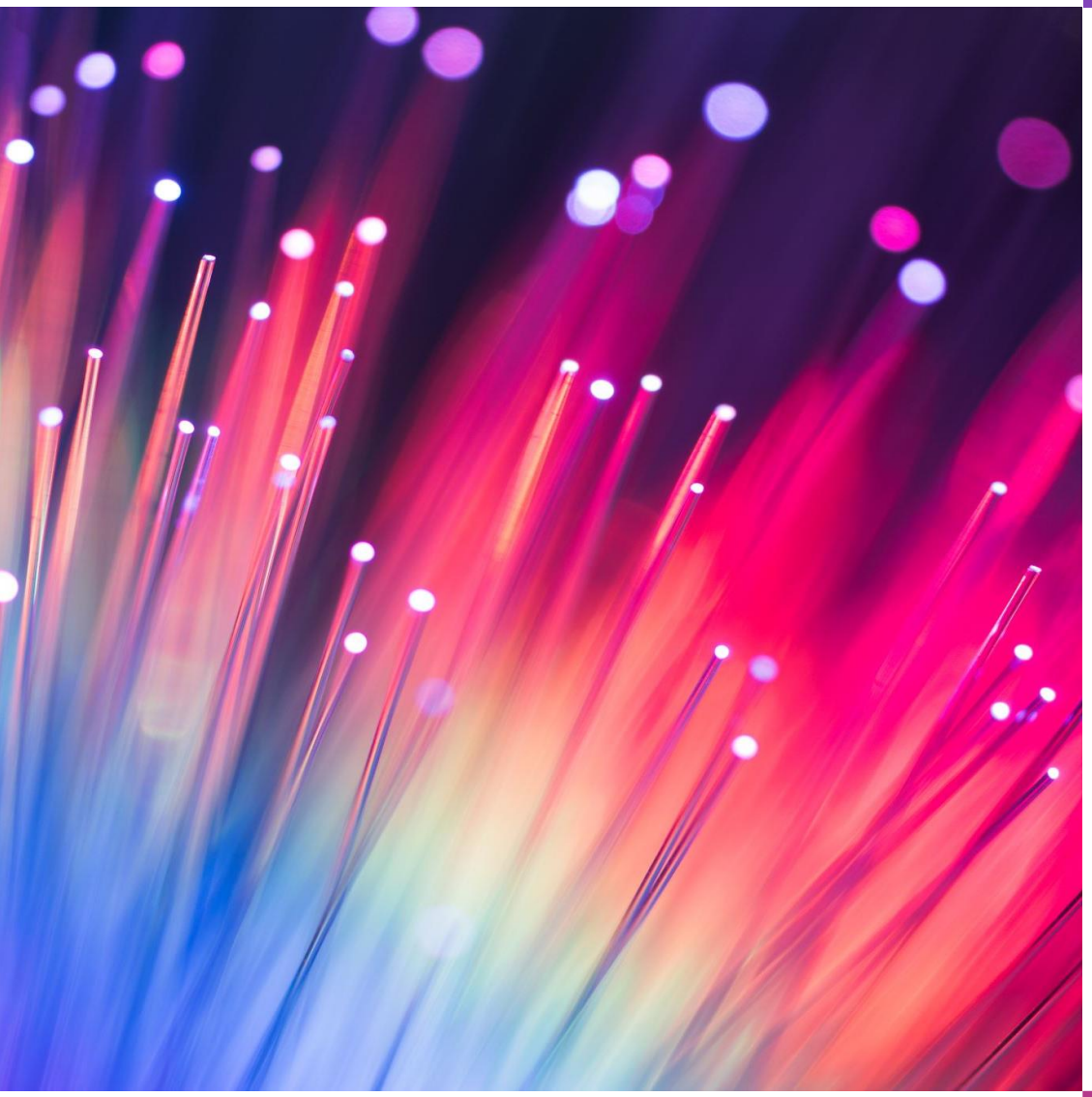
Supplemental Learning Dialogflow:

Maya Properties Bot

By TK Chung



Contents



Overview

Maya Properties Bot

+ Dialogflow chatbot for property consultancy
+ Provides information, property price estimate & helps customers schedule and cancel appointments with consultant
+ Deployed to Google Assistant for voice interface

Try it out at Web Link HERE

Functionalities

+ Answer usual questions
+ Check consultant availability
+ Make appointment
+ Leave message/comment
+ Cancel booked appointment
+ Check home price estimate
  - Parameters: locality, property type, area type of strata vs landed, & tenure of leasehold vs freehold
+ Appointments recorded in Google Calendar & Google Sheets

Overall Design

Dialogflow

Google Calendar

Gmail

Confirmation Email

Maya Properties Bot

Back End Fulfilment

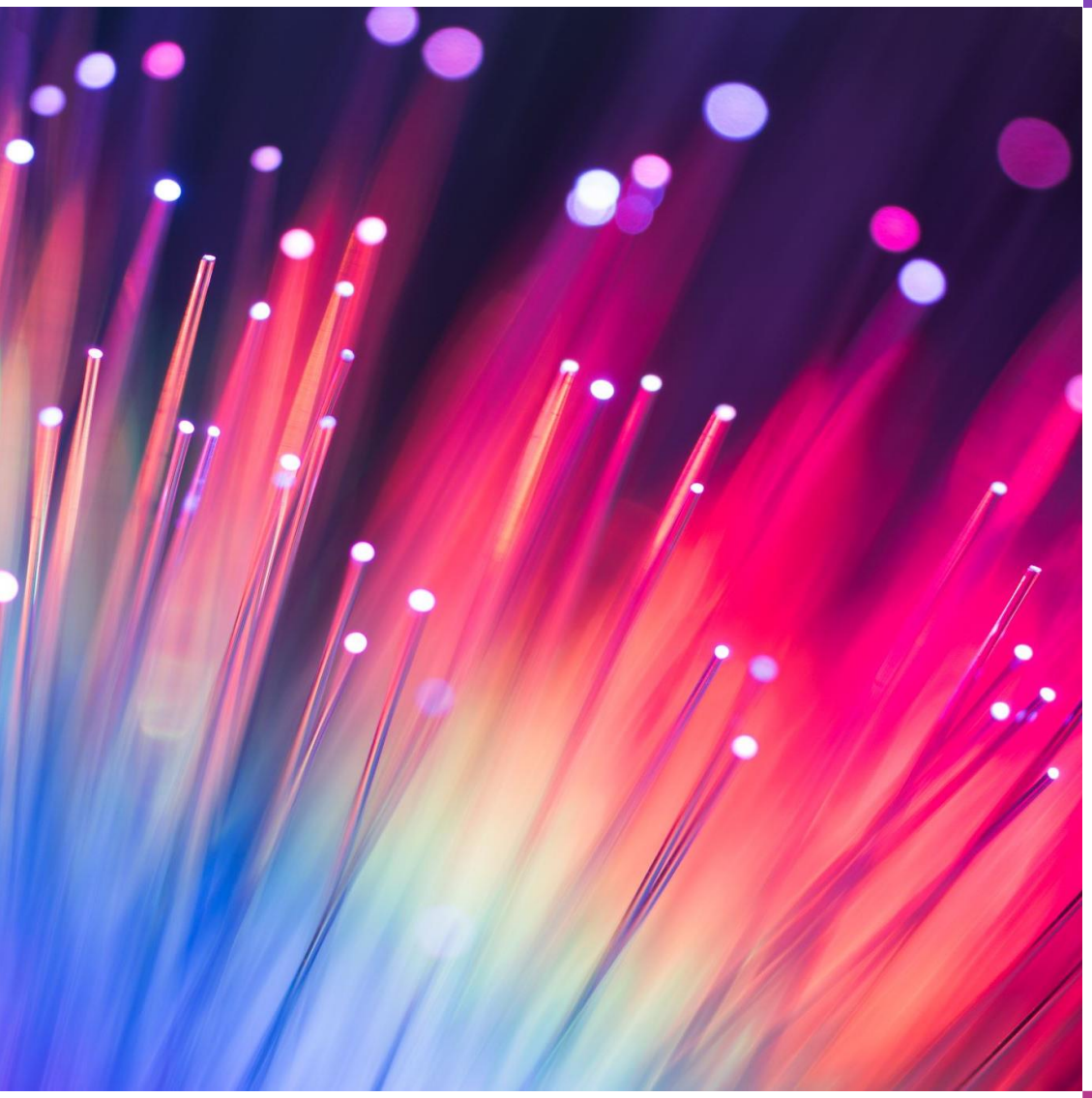
Google Sheets Database

Appointment Entries DB

Google Assistant

Customer

Front End Integration



Intents Map & Process Flow

Process Flow

Start

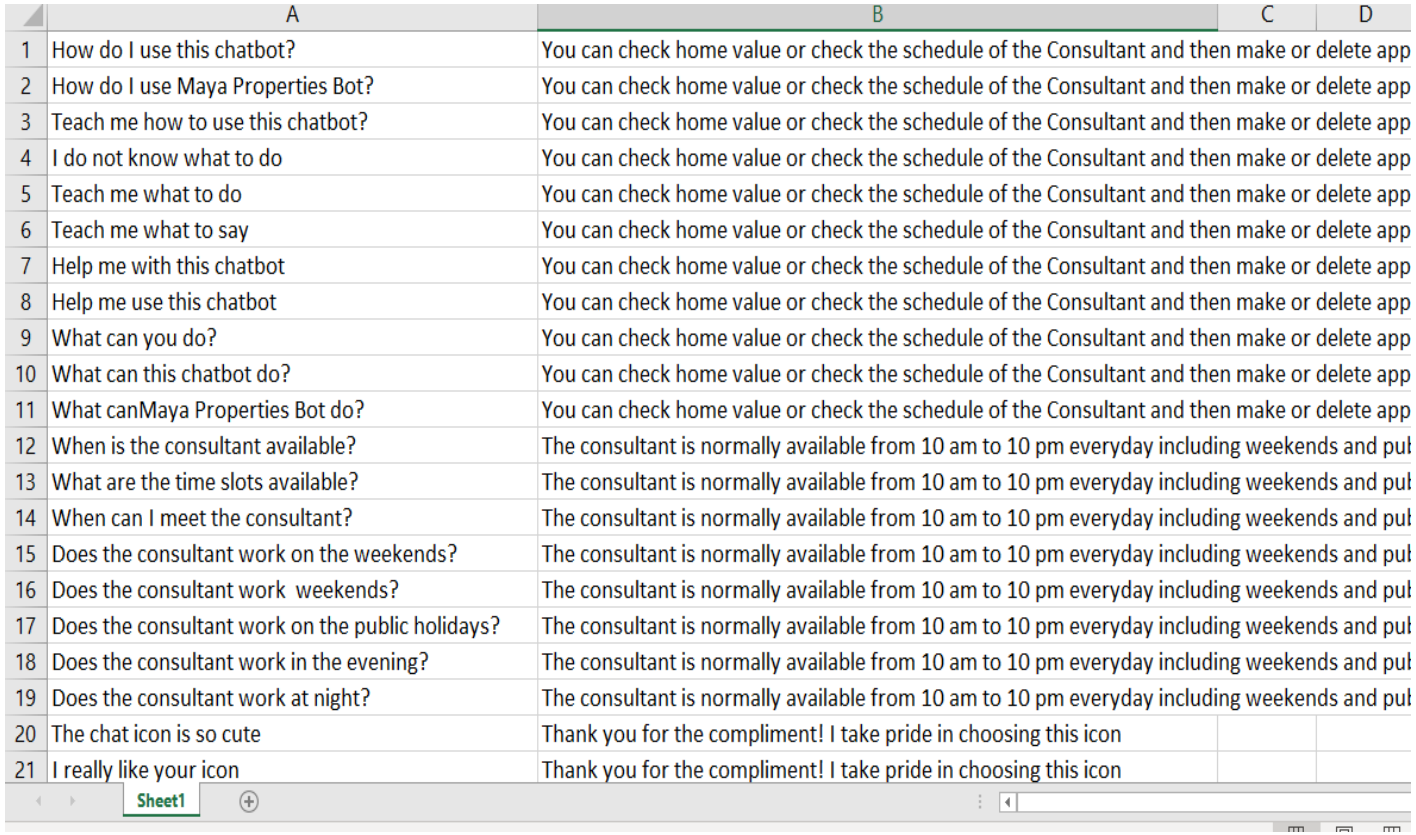
- Ask FAQ — **Answer Usual Questions**
- Check Schedule — **Check Availability on Specific Day**
- Make Appt → Failure Appt — LOOP
- Leave Comment
- Delete Appt
- Confirm Delete
- Home Value

Google Calendar · Gmail · G Sheets

Linear Regression Model Embedded as JSON object in Dialogflow backend code

Process 1: Ask FAQ Intent

- Gives FAQ information on Maya Properties Bot on business hours, how home value is derived etc
- Implement through **upload of csv file to Dialogflow Knowledge Base**

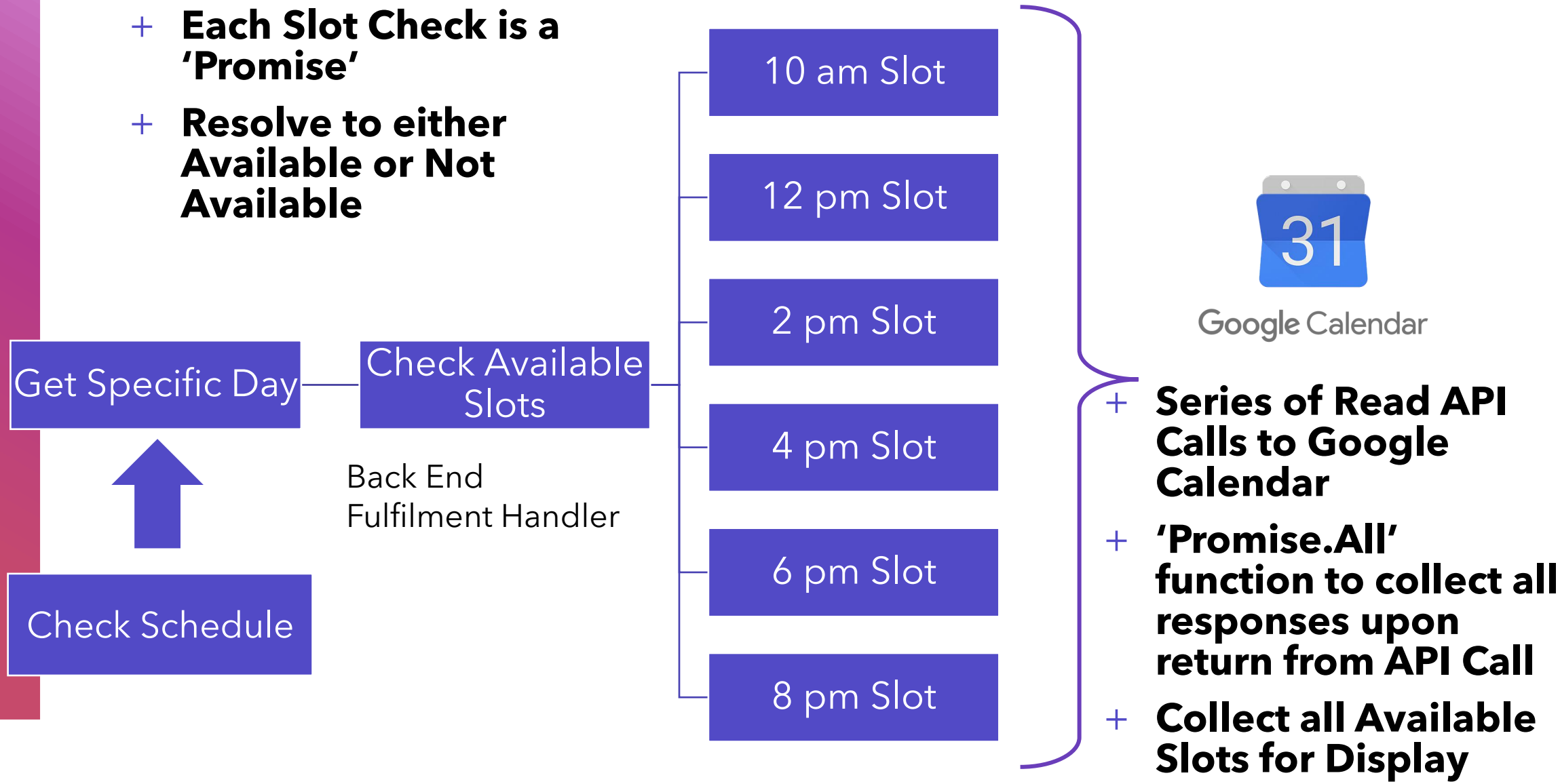


	A	B	C	D
1	How do I use this chatbot?	You can check home value or check the schedule of the Consultant and then make or delete app		
2	How do I use Maya Properties Bot?	You can check home value or check the schedule of the Consultant and then make or delete app		
3	Teach me how to use this chatbot?	You can check home value or check the schedule of the Consultant and then make or delete app		
4	I do not know what to do	You can check home value or check the schedule of the Consultant and then make or delete app		
5	Teach me what to do	You can check home value or check the schedule of the Consultant and then make or delete app		
6	Teach me what to say	You can check home value or check the schedule of the Consultant and then make or delete app		
7	Help me with this chatbot	You can check home value or check the schedule of the Consultant and then make or delete app		
8	Help me use this chatbot	You can check home value or check the schedule of the Consultant and then make or delete app		
9	What can you do?	You can check home value or check the schedule of the Consultant and then make or delete app		
10	What can this chatbot do?	You can check home value or check the schedule of the Consultant and then make or delete app		
11	What canMaya Properties Bot do?	You can check home value or check the schedule of the Consultant and then make or delete app		
12	When is the consultant available?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
13	What are the time slots available?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
14	When can I meet the consultant?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
15	Does the consultant work on the weekends?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
16	Does the consultant work weekends?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
17	Does the consultant work on the public holidays?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
18	Does the consultant work in the evening?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
19	Does the consultant work at night?	The consultant is normally available from 10 am to 10 pm everyday including weekends and pul		
20	The chat icon is so cute	Thank you for the compliment! I take pride in choosing this icon		
21	I really like your icon	Thank you for the compliment! I take pride in choosing this icon		

Sheet1

Process 2: Check Schedule Intent

- **Each Slot Check is a 'Promise'**
- **Resolve to either Available or Not Available**



- **Series of Read API Calls to Google Calendar**
- **'Promise.All' function to collect all responses upon return from API Call**
- **Collect all Available Slots for Display**

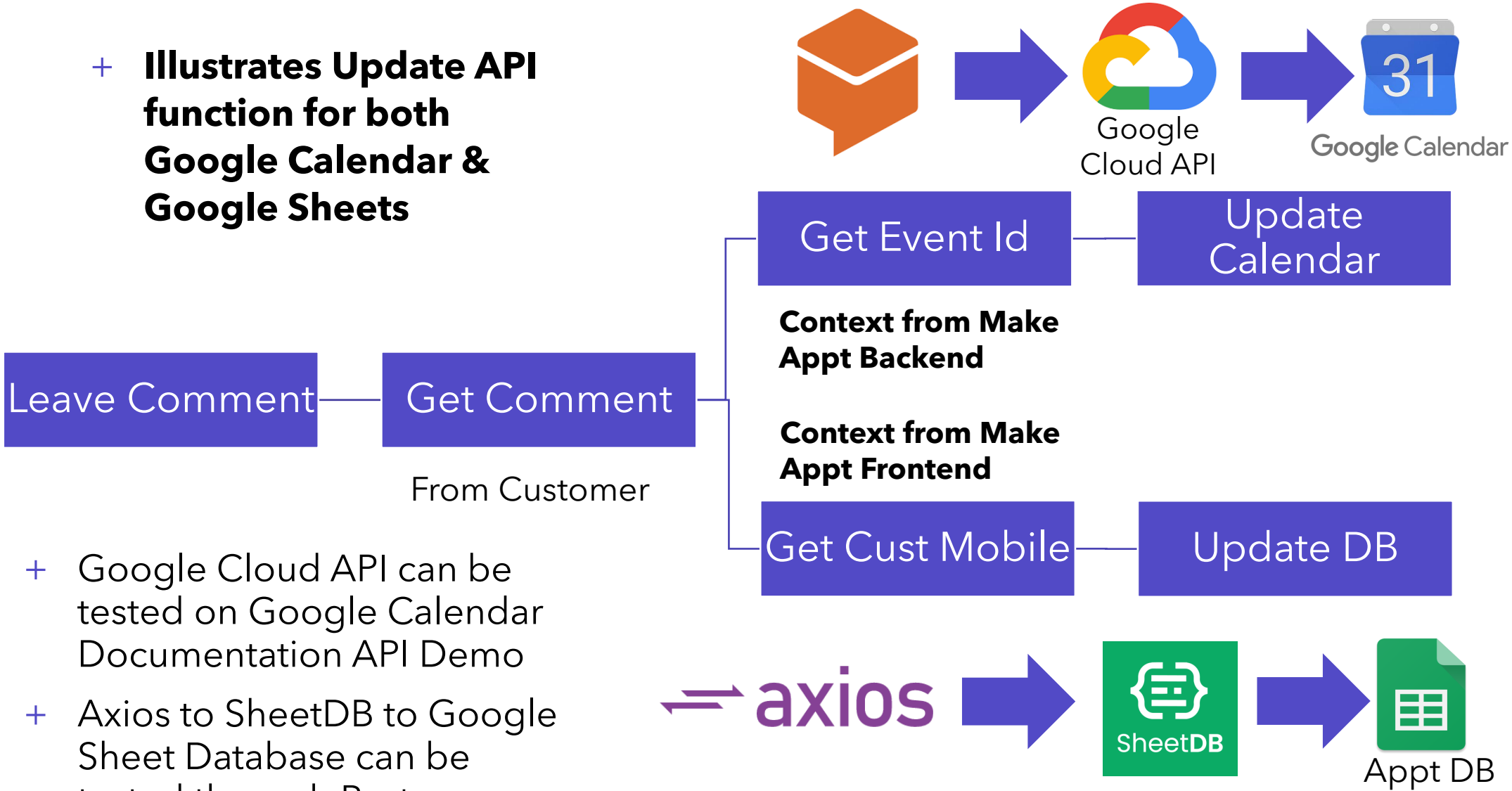
Process 3: Make Appt Intent

+ **Successful appointments recorded at Google Calendar, Google Sheets & Confirmation Email to customer**

+ **Slot Not Available triggers the Failure Appt Intent**
+ **Slot Available ends with question if customer wants to leave comment**
+ **Captures Yes response with Follow-up Leave Comment Intent**

Process 4: Leave Comment Intent

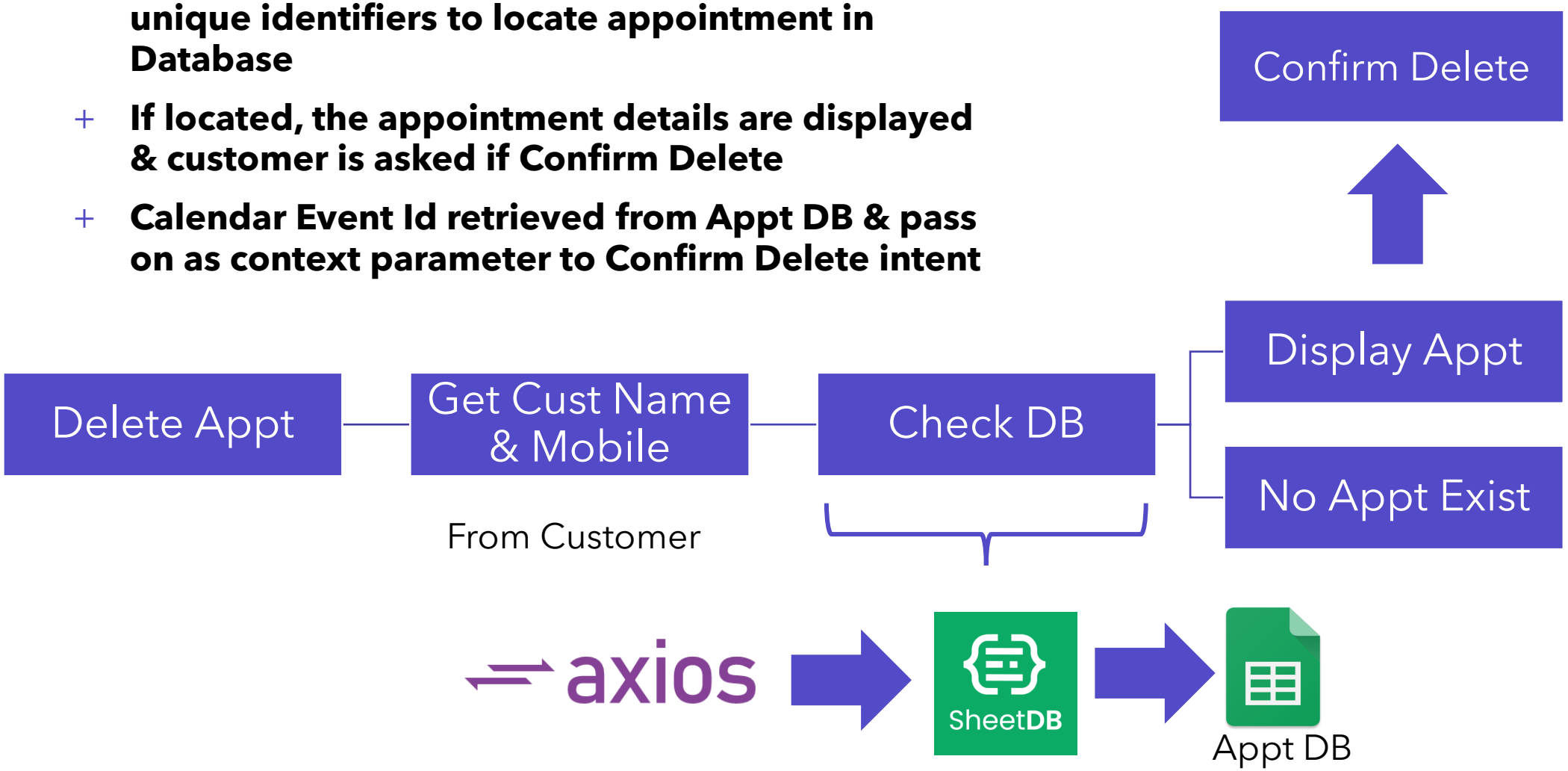
+ **Illustrates Update API function for both Google Calendar & Google Sheets**



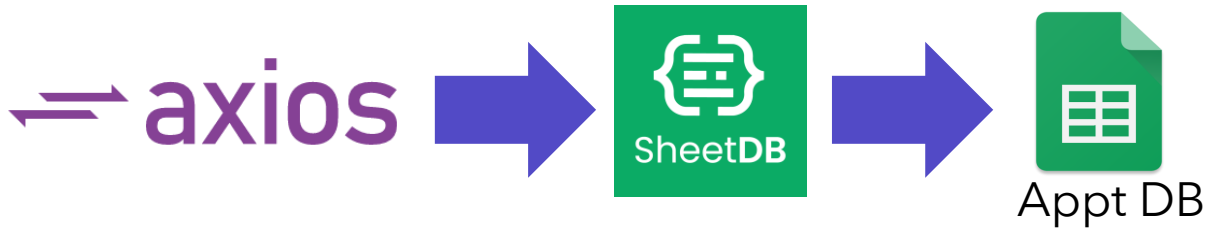
+ Google Cloud API can be tested on Google Calendar Documentation API Demo
+ Axios to SheetDB to Google Sheet Database can be tested through Postman

Process 5: Delete Appt Intent

- **Customer Name & Mobile used concurrently as unique identifiers to locate appointment in Database**
- **If located, the appointment details are displayed & customer is asked if Confirm Delete**
- **Calendar Event Id retrieved from Appt DB & pass on as context parameter to Confirm Delete intent**



Process 6: Confirm Delete Intent



Delete Appt

Get Cust Mobile → Update Appt DB

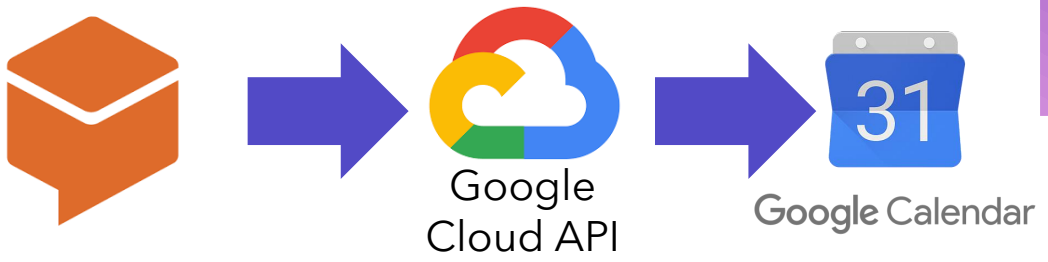
Context from Delete Appt Frontend

+ **Customer Mobile as unique identifier to update Appt DB**

Context from Delete Appt Backend

+ **Event Id as unique identifier to locate appt on Google Calendar for deletion**

Get Event Id → Delete Appt



Process 7: Check Home Value Intent

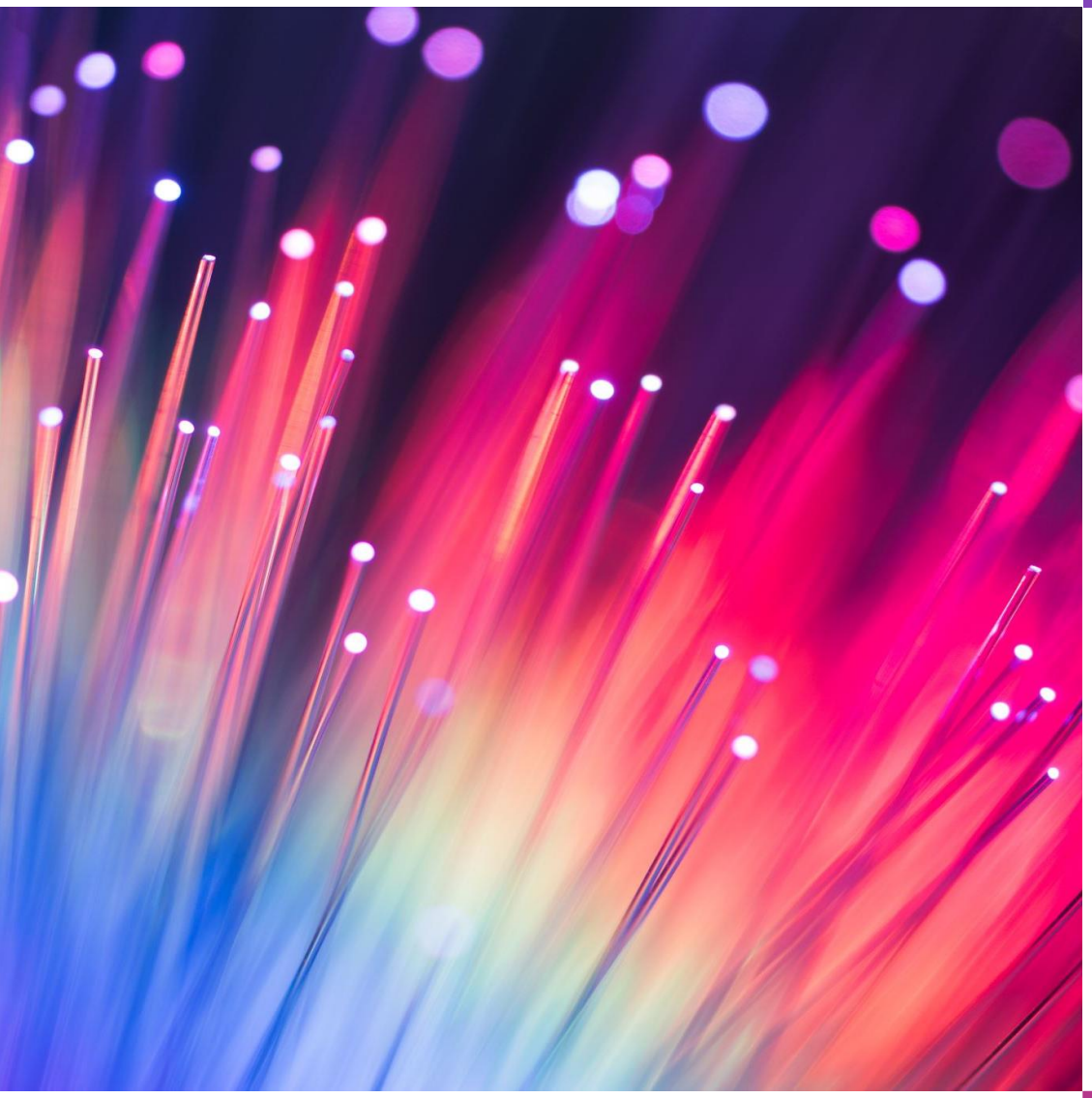
Check Home Value

- Get Locality
- Get Property Type
- Get Area Type
- Get Tenure Type
- Get Floor Area → Multiply to Model PSF

Model Outputs Price per Square Foot (PSF)

Linear Regression Model Coefficients embedded as JSON Object in Diagflow backend

Value Estimate



General Configuration

JS Maya Properties Bot V11 For Tutorial.js

```
1   // Maya Properties Bot Fulfillment Code
2   // Intents & Mapped Functions:
3   //      1. Make Appt => MakeAppt
4   //      2. Leave Comment => LeaveComment
5   //      3. Failure Appt => MakeAppt
6   //      4. Check Schedule => CheckSchedule
7   //      5. Delete Appt => DeleteAppt
8   //      6. Confirm Delete => ConfirmDelete
9   //      7. Home Value => HomeValue
10  // Configuration Needed for [1] to [7] Below
11
12  'use strict';
13
14  const functions = require('firebase-functions');
15  const { google } = require('googleapis');
16  const { WebhookClient } = require('dialogflow-fulfillment');
17  process.env.DEBUG = "dialogflow:debug";
18
19  // General Configuration
20  const AgentName = 'Maya'; // Change Agent Name as needed [1]
21  const AgentEmail = 'xxx@outlook.com'; // Agent Receive Mailbox Address [2]
22  const ChatbotEmail = 'xxx@gmail.com'; // Chatbot Send Gmail Address [3]
23
24  // Nodemailer Configuration
```

JS Maya Properties Bot V11 For Tutorial.js

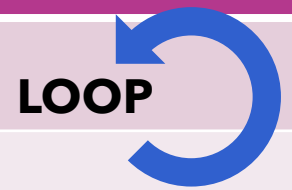
```
// General Configuration
const AgentName = 'Maya'; // Change Agent Name as needed [1]
const AgentEmail = 'xxx@outlook.com'; // Agent Receive Mailbox Address [2]
const ChatbotEmail = 'xxx@gmail.com'; // Chatbot Send Gmail Address [3]

// Nodemailer Configuration
// Add "nodemailer": "^4.6.7" dependency at package.json
const nodemailer = require('nodemailer');
const mailTransport = nodemailer.createTransport({
    host: 'smtp.gmail.com',
    port: '465',
    secure: 'true',
    service: 'Gmail',
    auth: {
        user: ChatbotEmail,
        pass: 'xxx' // Password of Chatbot Gmail (Set Less Secure) [4]
    }
});

// Google Calendar Configuration
const CalendarId = 'xxx@group.calendar.google.com'; //Enter CalendarId Here
[5]
const serviceAccount = {
```

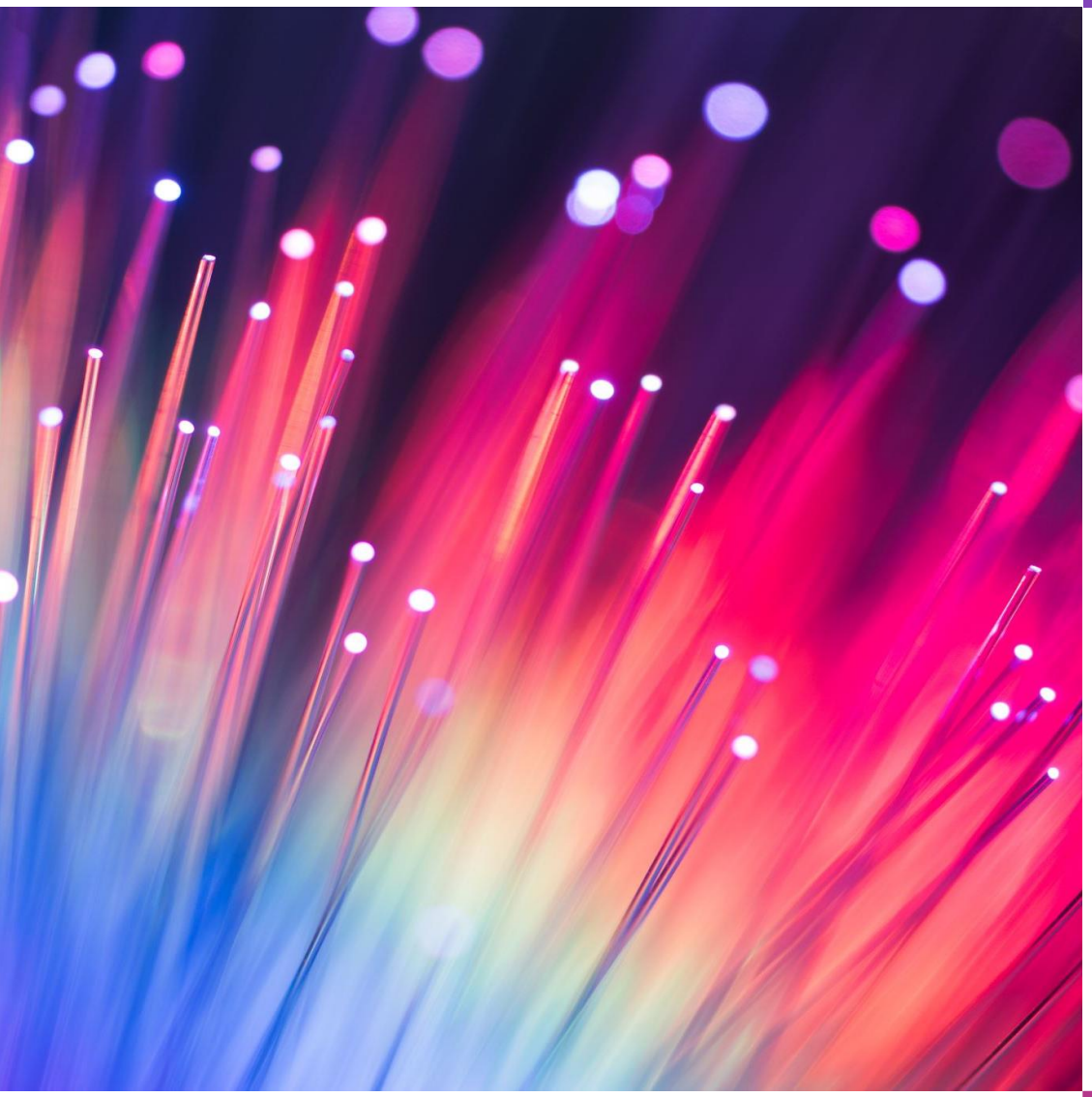
JS Maya Properties Bot V11 For Tutorial.js

```
37
38  // Google Calendar Configuration
39  const CalendarId = 'xxx@group.calendar.google.com'; //Enter CalendarId Here
    [5]
40  const serviceAccount = {
41      // Downloaded JSON Service Account Credentials
42      // Starting with: "type": "service_account", "project_id": [6]
43  };
44  const serviceAccountAuth = new google.auth.JWT({
45      email: serviceAccount.client_email,
46      key: serviceAccount.private_key,
47      scopes: 'https://www.googleapis.com/auth/calendar'
48  });
49  const calendar = google.calendar('v3');
50
51  // Axios API Call & SheetDB (Google Calendar) Configuration
52  // Add "axios": "^0.21.0" dependency at package.json
53  const axios = require('axios');
54  const EntryDBPath = 'https://sheetdb.io/api/xxxx'; // SheetDB Generated API
    [7]
55
56  exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request,
    response) => {
57      const agent = new WebhookClient({ request, response });
```

Fulfilment Code Intent-Function Handler Map

No	Intent	Function Handler in Index JS
1	Make Appt	MakeAppt
2	Failure Appt	MakeAppt
3	Leave Comment	LeaveComment
4	Check Schedule	CheckSchedule
5	DeleteAppt	DeleteAppt
6	Confirm Delete	ConfirmDelete
7	Home Value	HomeValue





Linear Regression Model & JSON Implementation

What is JSON?

- Javascript Object Format
- Key-Value Pairs in parentheses {}
- Values are data type like integer & string, or another JSON object, or array of JSON objects
- Supports Multi-level & Nested

- Example:

```
person = { name:"John",
           age:31,
           hobbies:["tennis", "golf"]};
```

How to Reference:

person.name is “John”
person.age is 31
person.hobbies[0] is “tennis”
person.hobbies[1] is “golf”

Python in Jupyter Notebook

Linear Regression Model for Predictor in Residential Value

- Singapore URA Real Estate Transactions Dataset
- 66,000 Historical Transactions of Private Property from 2017 to 2020
- Reduced Features Selection for Deployment to Dialogflow fulfillment

Data Exploration

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('dataset.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Project Name	Transacted Price ($)	Area (SQFT)	Unit Price ($ PSF)	Sale Date	Address	Type of Sale	Type of Area	Area (SQM)	Unit Price ($ PSM)	...

In [24]:
```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size =0.2, random_state=7)
```

In [26]:
```
model = LinearRegression()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
```

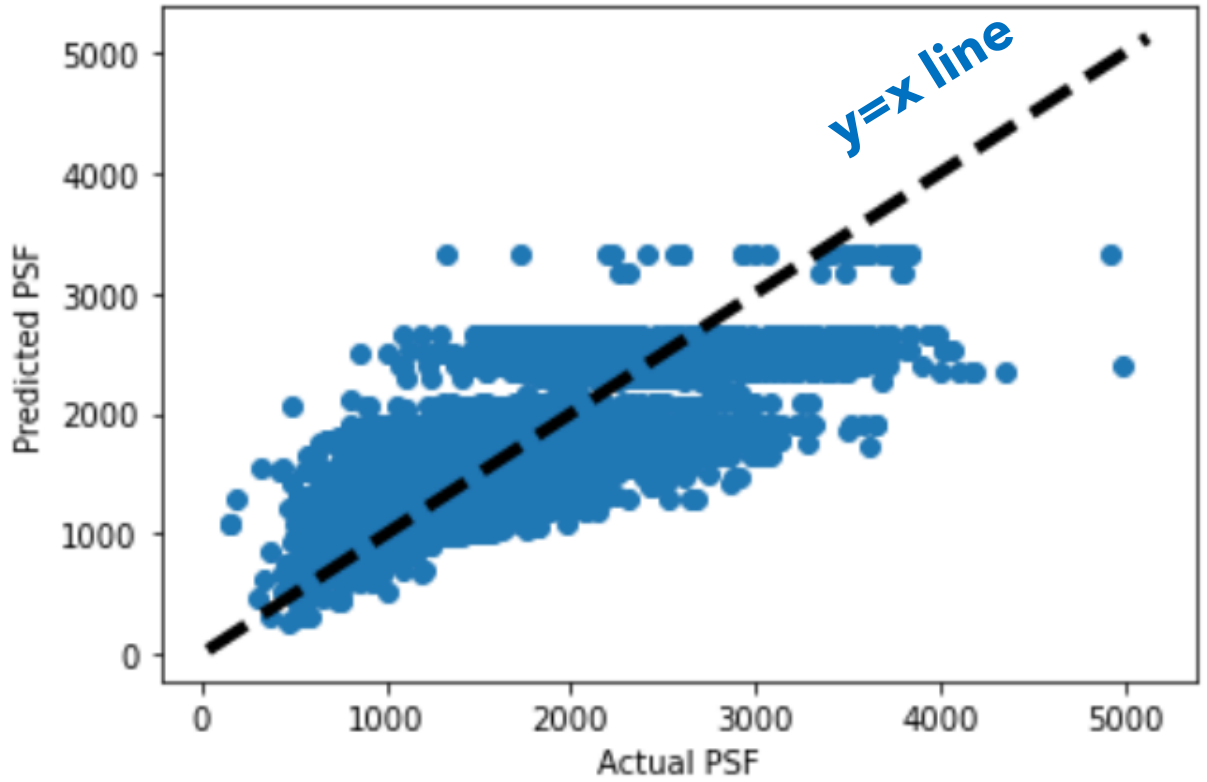
Model Evaluation

In [27]:
```
# importing r2_score module
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
# predicting the accuracy score
score=r2_score(y_test,y_predict)
score
```

Out[27]:
```
0.6051216417932486
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [37]:
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.scatter(y_test, y_predict)
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)
ax.set_xlabel('Actual PSF')
ax.set_ylabel('Predicted PSF')
plt.show()
```



In [31]:
```
# Save using sklearn_export
from sklearn_export import Export
export = Export(model)
result = export.to_json()
```

Visualising the Linear Regression Model & Predictions

In [32]:
```
y_predict.shape
```

Out[32]: (13358,)

In [33]:
```
y_test.shape
```

Out[33]: (13358,)

In [34]:
```
d = {'Prediction PSF':y_predict, 'Actual PSF': y_test}
```

In [35]:
```
output = pd.DataFrame(d)
```

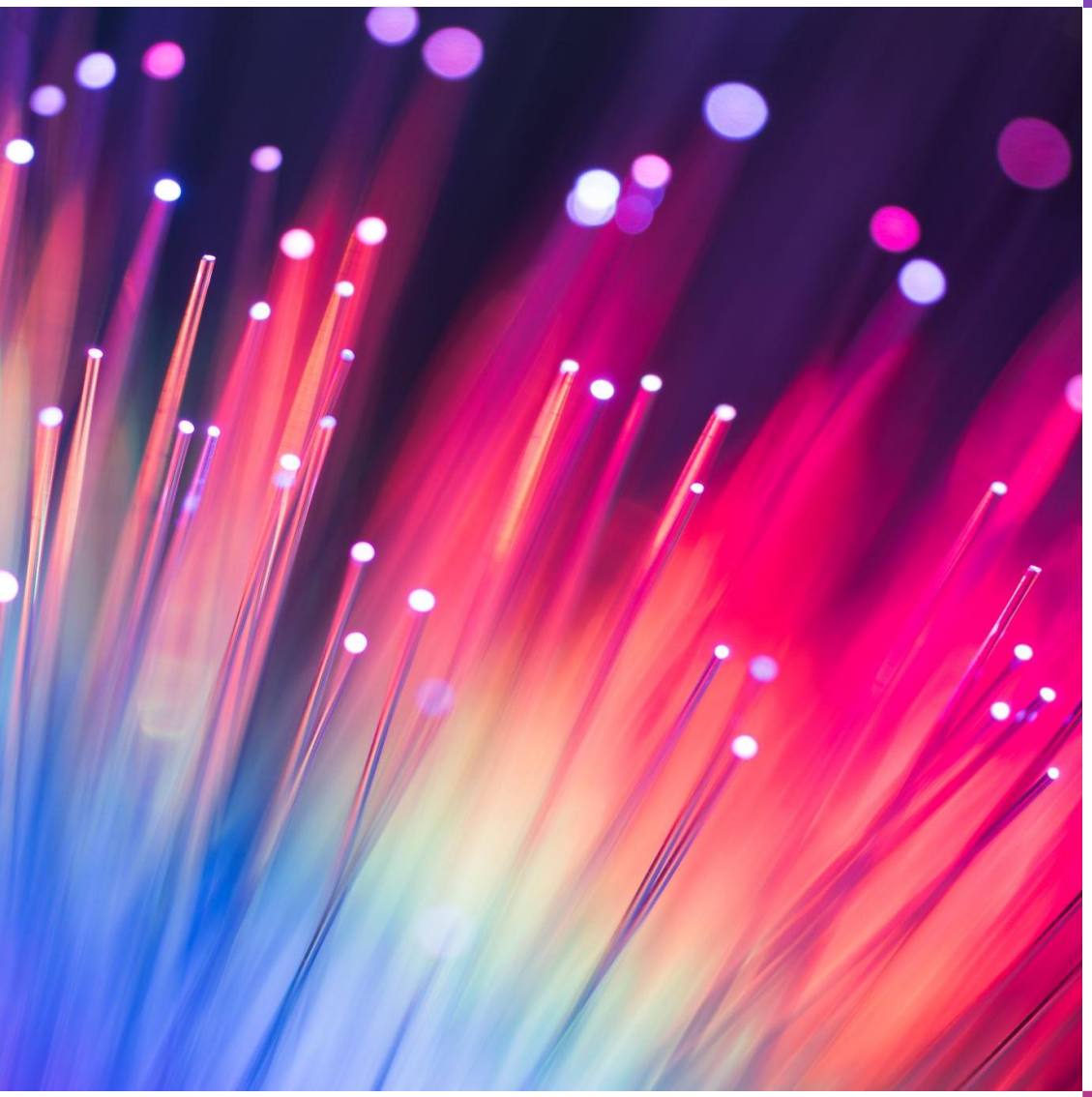
In [36]:
```
output.head()
```

```
53
54  exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request,
    response) => {
55      const agent = new WebhookClient({ request, response });
56
57      function HomeValue(agent) {
58          const Locality = agent.parameters.Locality;
59          const Tenure = agent.parameters.Tenure;
60          const Property = agent.parameters.Property;
61          const AreaType = agent.parameters.AreaType;
62          const Size = agent.parameters.Size;
63          const LRModel = {
64              "Locality": {
65                  "Ang Mo Kio": -93256244214866.8,
66                  "Bedok": -93256244214820.8,
67                  "Bishan": -93256244214474.9,
68                  "Bukit Batok": -93256244214893,
69                  "Bukit Merah": -93256244214247,
70                  "Bukit Panjang": -93256244214912.4,
71                  "Bukit Timah": -93256244214361.7,
72                  "Changi": -93256244214955.6,
73                  "Choa Chu Kang": -93256244215133,
74                  "Clementi": -93256244214704.3,
75                  "Downtown": -93256244213747.2,
```

```
    "Clementi": -93256244214704.5,
    "Downtown": -93256244213747.2,
    "Geylang": -93256244214619,
    "Hougang": -93256244214845.5,
    "Jurong East": -93256244214946.3,
    "Jurong West": -93256244215007.8,
    "Kallang": -93256244214612.7,
    "Mandai": -93256244215219.3,
    "Marine Parade": -93256244214246.9,
    "Museum": -93256244213846.2,
    "Newton": -93256244213657.5,
    "Novena": -93256244214351,
    "Orchard": -93256244212830.9,
    "Outram": -93256244213866.2,
    "Pasir Ris": -93256244215010.4,
    "Punggol": -93256244214811.9,
    "Queenstown": -93256244214377.8,
    "River Valley": -93256244213489.2,
    "Rochor": -93256244214367.3,
    "Sembawang": -93256244214860.8,
    "Sengkang": -93256244214797.6,
    "Serangoon": -93256244214684.4,
    "Singapore River": -93256244214058.7,
    "Southern Islands": -93256244214429,
```

```
            "Singapore River": -93256244214058.7,
            "Southern Islands": -93256244214429,
            "Sungei Kadut": -93256244215189,
            "Tampines": -93256244214775.7,
            "Tanglin": -93256244214100.8,
            "Toa Payoh": -93256244214420.9,
            "Woodlands": -93256244215106.6,
            "Yishun": -93256244214993.6
        },
        "Tenure": {
            "Freehold": 1281566521830990,
            "Leasehold": 1281566521830980
        },
        "Property": {
            "Apartment": 928522351986515,
            "Condominium": 928522351986370,
            "Detached House": 928522351985630,
            "Executive Condominium": 928522351986172,
            "Semi-Detached House": 928522351985687,
            "Terrace House": 928522351985876
        },
        "AreaType": {
            "Land": -361423160876231,
            "Strata": -361423160876819
```

```
                "Detached House": 928522351985630,
                "Executive Condominium": 928522351986172,
                "Semi-Detached House": 928522351985687,
                "Terrace House": 928522351985876
            },
            "AreaType": {
                "Land": -361423160876231,
                "Strata": -361423160876819
            },
            "intercept": -1755409468724530
        };
        const HomeValuePSF = LRModel.Locality[Locality] +
            LRModel.Tenure[Tenure] + LRModel.Property[Property] +
            LRModel.AreaType[AreaType] + LRModel.intercept;
        const HomeValue = '$' + parseInt(HomeValuePSF * Size).toLocaleString
        ("en-US");
        agent.add(`The estimated value of the home is ${HomeValue}. ` +
            `It is derived from a Linear Regression Model based on historical
            data. For more information on how this estimate is derived, you
            can check out Ask FAQ. ` +
            `You can make appointment with the consultant, go start or end
            call`);
    }
```



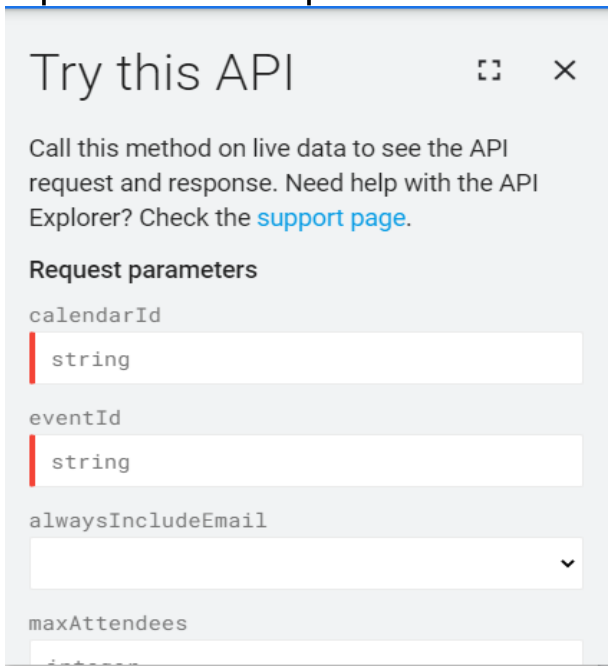
Google Calendar: CRUD Functions

What is API call?

- For accessing external cloud resources
- Request & Response bodies in JSON format
- Supports CRUD functions

Function	Possible Syntax Request
Create	Post, Insert
Read	Get, List
Update	Patch, Put
Delete	Delete

Google has API Explorer to test Request & Response bodies

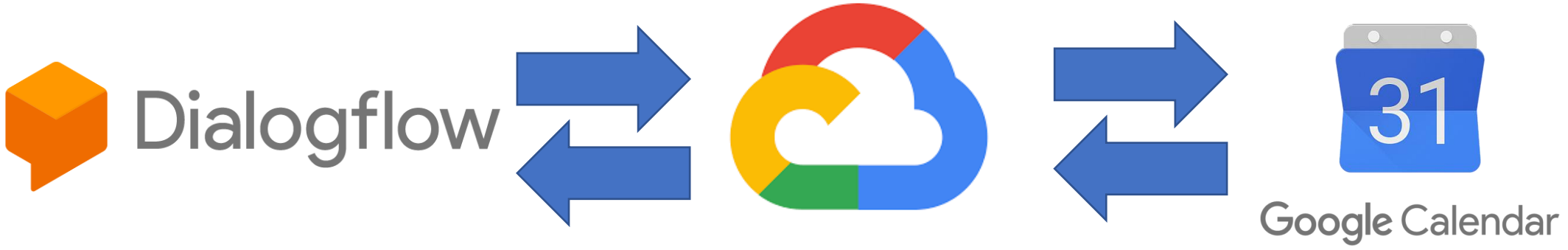


https://developers.google.com/calendar/api/v3/reference/events/get

Google Cloud API Call

Service Account Authentication

Calendar Sharing



CRUD Functions

Google Cloud API call

For more info:

https://codelabs.developers.google.com/codelabs/chatbots-dialogflow-fulfillment#0

```
131
132     function MakeAppt(agent) {
133         const CustName = agent.parameters.CustName;
134         const CustEmail = agent.parameters.CustEmail;
135         const CustMobile = agent.parameters.CustMobile;
136         const CustComment = 'NIL';
137         const ApptDate = agent.parameters.ApptDate.split('T')[0];
138         const ApptTime = agent.parameters.ApptTime.split('T')[1].substr(0, 14)
            ;
139         const dateTimeStart = new Date(ApptDate + 'T' + ApptTime);
140         const dateTimeEnd = new Date(new Date(dateTimeStart).setHours
            (dateTimeStart.getHours() + 1));
141         const appointment_type = CustName + ' ' + CustMobile + ' ' +
            CustEmail;
142         const ApptDateFormat = ApptDate.split('-').reverse().join('-');
143         const ApptTimeFormat = ApptTime.substr(0, 5);
144
145         // Email Notification to Customer
146         const EmailSubject1 = 'Appointment Confirmation from Maya Properties
            Bot';
147         const EmailBody1 = 'Dear ' + CustName +
148             '<p> Thank you for using Maya Properties.' +
149             '<p> The appointment is confirmed as follows : ' +
150             '<p><strong> Customer Name: </strong>' + CustName +
```

```
144
145         // Email Notification to Customer
146         const EmailSubject1 = 'Appointment Confirmation from Maya Properties
            Bot';
147         const EmailBody1 = 'Dear ' + CustName +
148             '<p> Thank you for using Maya Properties.' +
149             '<p> The appointment is confirmed as follows : ' +
150             '<p><strong> Customer Name: </strong>' + CustName +
151             '<br><strong> Appointment Date & Time: </strong>' +
                ApptDateFormat + ' ' + ApptTimeFormat +
152             '<br><strong> Consultant Name: </strong>' + AgentName +
153             '<p> The consultant will be contacting you shortly on the details
                of the appointment.' +
154             '<p> Have a nice day!' +
155             '<p> From: Maya Properties Bot';
156
157         // Email Notification to Consultant
158         const EmailSubject2 = 'Appointment Confirmation from Maya Properties
            Bot';
159         const EmailBody2 = 'Dear ' + AgentName +
160             '<p> There is a new customer appointment from Maya Properties Bot.
                ' +
161             '<p> The appointment is confirmed as follows : ' +
162             '<p><strong> Customer Name: </strong>' + CustName +
```

JS Maya Properties Bot V11 For Tutorial.js

```
156
157         // Email Notification to Consultant
158         const EmailSubject2 = 'Appointment Confirmation from Maya Properties
            Bot';
159         const EmailBody2 = 'Dear ' + AgentName +
160             '<p> There is a new customer appointment from Maya Properties Bot.
                ' +
161             '<p> The appointment is confirmed as follows : ' +
162             '<p><strong> Customer Name: </strong>' + CustName +
163             '<br><strong> Customer Mobile: </strong>' + CustMobile +
164             '<br><strong> Customer Email: </strong>' + CustEmail +
165             '<br><strong> Consultant Name: </strong>' + AgentName +
166             '<br><strong> Appointment Date & Time: </strong>' +
                ApptDateFormat + ' ' + ApptTimeFormat +
167             '<p> Please contact the customer quickly on the details of the
                appointment.' +
168             '<p> Have a nice day!' +
169             '<p> From: Maya Properties Bot';
170
171         // Check the time availibility and make appointment
172         return createCalendarEvent(dateTimeStart, dateTimeEnd,
            appointment_type).then((calendarResponse) => {
173             sendEmail(CustEmail, EmailSubject1, EmailBody1);
174             sendEmail(AgentEmail, EmailSubject2, EmailBody2);
```

JS Maya Properties Bot V11 For Tutorial.js

```
170
171         // Check the time availibility and make appointment
172         return createCalendarEvent(dateTimeStart, dateTimeEnd,
            appointment_type).then((calendarResponse) => {
173             sendEmail(CustEmail, EmailSubject1, EmailBody1);
174             sendEmail(AgentEmail, EmailSubject2, EmailBody2);
175             const EventId = calendarResponse.data.id;
176             agent.context.set({
177                 'name': 'MakeAppt-followup',
178                 'lifespan': 2,
179                 'parameters': {
180                     'EventId': EventId,
181                     'CalendarId': CalendarId
182                 }
183             });
184             CreateDBEntry(AgentName, ApptDateFormat, ApptTimeFormat,
            CustName, CustMobile, CustEmail, CustComment, EventId);
185             agent.add(`The appointment with consultant ${AgentName} on $
            {ApptDateFormat} at ${ApptTimeFormat} has been confirmed. ` +
186                 `The consultant will be contacting you shortly on the details
                of the appointment. Would you like to leave a comment for the
                consultant? ` +
187                 `If no, you can end call`);
188         }).catch(() => {
```

```
176             agent.context.set({
177                 'name': 'MakeAppt-followup',
178                 'lifespan': 2,
179                 'parameters': {
180                     'EventId': EventId,
181                     'CalendarId': CalendarId
182                 }
183             });
184             CreateDBEntry(AgentName, ApptDateFormat, ApptTimeFormat,
                CustName, CustMobile, CustEmail, CustComment, EventId);
185             agent.add(`The appointment with consultant ${AgentName} on $
                {ApptDateFormat} at ${ApptTimeFormat} has been confirmed. ` +
186                 `The consultant will be contacting you shortly on the details
                of the appointment. Would you like to leave a comment for the
                consultant? ` +
187                 `If no, you can end call`);
188         }).catch(() => {
189             agent.add(`I'm sorry, there are no slots available for $
                {AgentName} on ${ApptDate} at ${ApptTimeFormat}`);
190             agent.setFollowupEvent('FailureAppt');
191         });
192     }
193
194     function createCalendarEvent(dateTimeStart, dateTimeEnd,
```

JS Maya Properties Bot V11 For Tutorial.js

```
193
194     function createCalendarEvent(dateTimeStart, dateTimeEnd,
        appointment_type) {
195         return new Promise((resolve, reject) => {
196             calendar.events.list({                    READ
197                 auth: serviceAccountAuth,
198                 calendarId: CalendarId,
199                 timeMin: dateTimeStart.toISOString(),
200                 timeMax: dateTimeEnd.toISOString()
201             }, (err, calendarResponse) => {
202                 if (err || calendarResponse.data.items.length > 0) {
203                     reject(err || new Error('Requested time conflicts with
                        another appointment'));
204                 } else {
205                     // Create event for requested time period
206                     calendar.events.insert({          CREATE
207                         auth: serviceAccountAuth,
208                         calendarId: CalendarId,
209                         resource: {
210                             summary: appointment_type + ' Appointment',
                                description: appointment_type,
211                             start: { dateTime: dateTimeStart },
212                             end: { dateTime: dateTimeEnd }
213                         }
```

JS Maya Properties Bot V11 For Tutorial.js

```
208                         calendarId: calendarId,
209                         resource: {
210                             summary: appointment_type + ' Appointment',
                                description: appointment_type,
211                             start: { dateTime: dateTimeStart },
212                             end: { dateTime: dateTimeEnd }
213                         }
214                     }, (err, event) => {
215                         if (err) {
216                             console.log('There is error: ' + err);
217                             reject(err);
218                         }
219                         else {
220                             console.log('Event created EventID: ' + event.
                                data.id);
221                             resolve(event);
222                         }
223                     }
224                     );
225                 }
226             });
227         });
228     }
229
```

```
273
274     async function DeleteAppt(agent) {
275         var ApptDB = await GetDBData(EntryDBPath);
276         const CustName = agent.parameters.CustName;
277         const CustMobile = agent.parameters.CustMobile;
278
279         var ApptExist = false;
280         ApptDB.forEach(element => {
281             if ((CustName.toLowerCase() == element.CustName.toLowerCase()) &&
                (CustMobile == element.CustMobile) && element.Status == 'ACTIVE')
                {
282                 agent.add(`Yes, you have an existing appointment on ${element.
                    ApptDate} at ${element.ApptTime}. ` +
283                     `Please 'Confirm Delete' if you want to proceed with the
                        cancellation`);
284                 ApptExist = true;
285                 agent.context.set({
286                     'name': 'DeleteDetails',
287                     'lifespan': 2,
288                     'parameters': {
289                         'CustName': CustName,
290                         'CustMobile': CustMobile,
291                         'CustEmail': element.CustEmail,
292                         'ApptDate': element.ApptDate,
```

```
284                 ApptExist = true;
285                 agent.context.set({
286                     'name': 'DeleteDetails',
287                     'lifespan': 2,
288                     'parameters': {
289                         'CustName': CustName,
290                         'CustMobile': CustMobile,
291                         'CustEmail': element.CustEmail,
292                         'ApptDate': element.ApptDate,
293                         'ApptTime': element.ApptTime,
294                         'EventId': element.EventId
295                     }
296                 });
297             }
298         });
299         if (!ApptExist) {
300             agent.add(`I am sorry but I couldn't find any appointments
                registered to ${CustName} and ${CustMobile}. ` +
301                 `You can try again to delete appointment, go start or end
                call`);
302         }
303     }
304
305     function ConfirmDelete(agent) {
```

```
304
305     function ConfirmDelete(agent) {
306         const CustName = agent.parameters.CustName;
307         const CustEmail = agent.parameters.CustEmail;
308         const CustMobile = agent.parameters.CustMobile;
309         const ApptDateFormat = agent.parameters.ApptDate;
310         const ApptTimeFormat = agent.parameters.ApptTime;
311         const EventId = agent.parameters.EventId;
312
313         // Confirm Delete Routine Part 1: Update ACTIVE Status to DELETED at
            EntryDB Google Sheet
314         const DeletionPath = EntryDBPath + `/CustMobile/${CustMobile}`;
315         axios.put(DeletionPath, {
316             "data": {
317                 "Status": "DELETED",
318                 "TimeStampD": new Date()
319             }
320         }).then(res => {
321             console.log('Log from Update Delete Status Function ' + res.data);
322         }).catch(err => { console.log(err); });
323         // End of Part 1
324
325         // Confirm Delete Routine Part 2: Send Delete Confirmation Emails to
            Customer and Consultant
```

```
324
325         // Confirm Delete Routine Part 2: Send Delete Confirmation Emails to
            Customer and Consultant
326         // Email Notification to Customer
327         const EmailSubject1 = 'Appointment Deleted Confirmation from Maya
            Properties Bot';
328         const EmailBody1 = 'Dear ' + CustName +
329             '<p> Thank you for using Maya Properties.' +
330             '<p> The appointment below is DELETED : ' +
331             '<p><strong> Customer Name: </strong>' + CustName +
332             '<br><strong> Appointment Date & Time: </strong>' +
                ApptDateFormat + ' ' + ApptTimeFormat +
333             '<br><strong> Consultant Name: </strong>' + AgentName +
334             '<p> We hope to hear from you again soon!' +
335             '<p> Have a nice day!' +
336             '<p> From: Maya Properties Bot';
337
338         // Email Notification to Consultant
339         const EmailSubject2 = 'Appointment Deleted Confirmation from Maya
            Properties Bot';
340         const EmailBody2 = 'Dear ' + AgentName +
341             '<p> There is a appointment deleted notification from Maya
                Properties Bot.' +
342             '<p> The appointment below is DELETED : ' +
```

```
337
338     // Email Notification to Consultant
339     const EmailSubject2 = 'Appointment Deleted Confirmation from Maya
        Properties Bot';
340     const EmailBody2 = 'Dear ' + AgentName +
341         '<p> There is a appointment deleted notification from Maya
        Properties Bot.' +
342         '<p> The appointment below is DELETED : ' +
343         '<p><strong> Customer Name: </strong>' + CustName +
344         '<br><strong> Customer Mobile: </strong>' + CustMobile +
345         '<br><strong> Customer Email: </strong>' + CustEmail +
346         '<br><strong> Consultant Name: </strong>' + AgentName +
347         '<br><strong> Appointment Date & Time: </strong>' +
        ApptDateFormat + ' ' + ApptTimeFormat +
348         '<p> There is no need to contact the customer.' +
349         '<p> Have a nice day!' +
350         '<p> From: Maya Properties Bot';
351
352     sendEmail(CustEmail, EmailSubject1, EmailBody1);
353     sendEmail(AgentEmail, EmailSubject2, EmailBody2);
354     // End of Part 2
355
356     // Confirm Delete Routine Part 3: Delete Google Calendar Event
357     deleteCalendarEvent(CalendarId, EventId);
```

Maya Properties Bot V11 For Tutorial.js

```
355
356         // Confirm Delete Routine Part 3: Delete Google Calendar Event
357         deleteCalendarEvent(CalendarId, EventId);
358         // End of Part 3
359         agent.add(`The appointment has been successfully deleted. ` +
360             `You can go start or simply end call`);
361     }
362
363     function deleteCalendarEvent(CalendarId, EventId) {
364         return new Promise((resolve, reject) => {
365             calendar.events.delete({
366                 auth: serviceAccountAuth,
367                 calendarId: CalendarId,
368                 eventId: EventId
369             }, (err, event) => {
370                 if (err) { reject(err); }
371                 else { resolve(event); }
372             });
373         });
374     }
375
376     function LeaveComment(agent) {
377         const CustComment = agent.parameters.CustComment;
378         const CalendarId = agent.parameters.CalendarId;
```

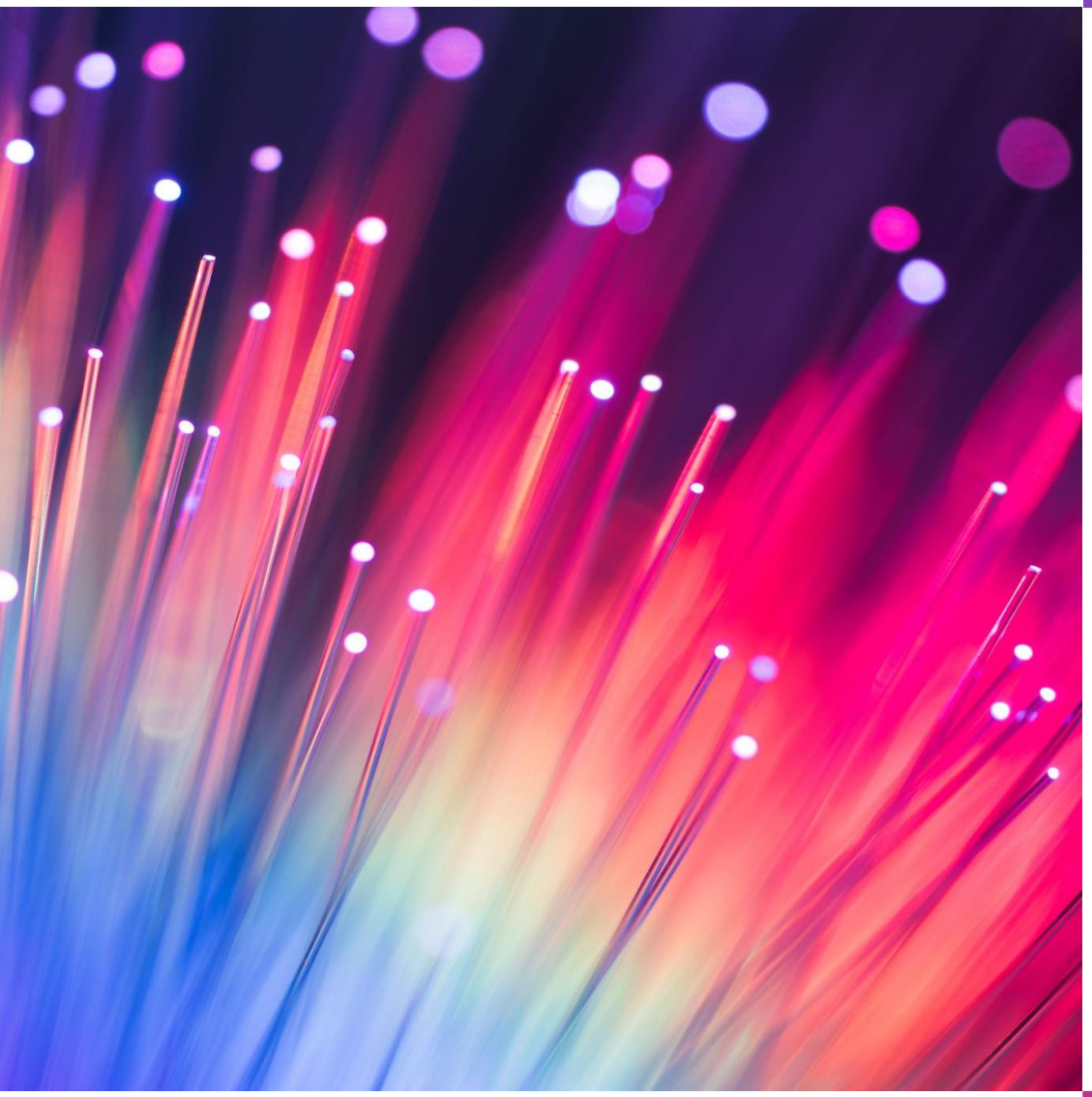
DELETE

```
375
376     function LeaveComment(agent) {
377         const CustComment = agent.parameters.CustComment;
378         const CalendarId = agent.parameters.CalendarId;
379         const EventId = agent.parameters.EventId;
380
381         // Insert Customer Comment to EntryDB Google Sheet Database
382         const UpdatePath = EntryDBPath + `/EventId/${EventId}`;
383         axios.put(UpdatePath, {
384             "data": {
385                 "CustComment": CustComment
386             }
387         }).then(res => {
388             console.log('Log from Leave Comment Update EntryDB ' + res.data);
389         }).catch(err => { console.log(err); });
390
391         // Insert Customer Comment to Google Calendar Appointment
392         const CustCommentM = 'Comment from Customer: ' + CustComment;
393         return UpdateCalendar(CalendarId, EventId, CustCommentM).then(
            (response) => {
394             console.log('Update Calendar function success ', response);
395             agent.add(`Your comment has been passed to the Consultant. ` +
396                 `You can go start or end call`);
397         }).catch((err) => {
```

```
380
381         // Insert Customer Comment to EntryDB Google Sheet Database
382         const UpdatePath = EntryDBPath + `/EventId/${EventId}`;
383         axios.put(UpdatePath, {
384             "data": {
385                 "CustComment": CustComment
386             }
387         }).then(res => {
388             console.log('Log from Leave Comment Update EntryDB ' + res.data);
389         }).catch(err => { console.log(err); });
390
391         // Insert Customer Comment to Google Calendar Appointment
392         const CustCommentM = 'Comment from Customer: ' + CustComment;
393         return UpdateCalendar(CalendarId, EventId, CustCommentM).then(
            (response) => {
394             console.log('Update Calendar function success ', response);
395             agent.add(`Your comment has been passed to the Consultant. ` +
396                 `You can go start or end call`);
397         }).catch((err) => {
398             console.log('Error in Update Calendar ', err);
399             agent.add(`It seems that something is wrong.`);
400         });
401     }
402
```

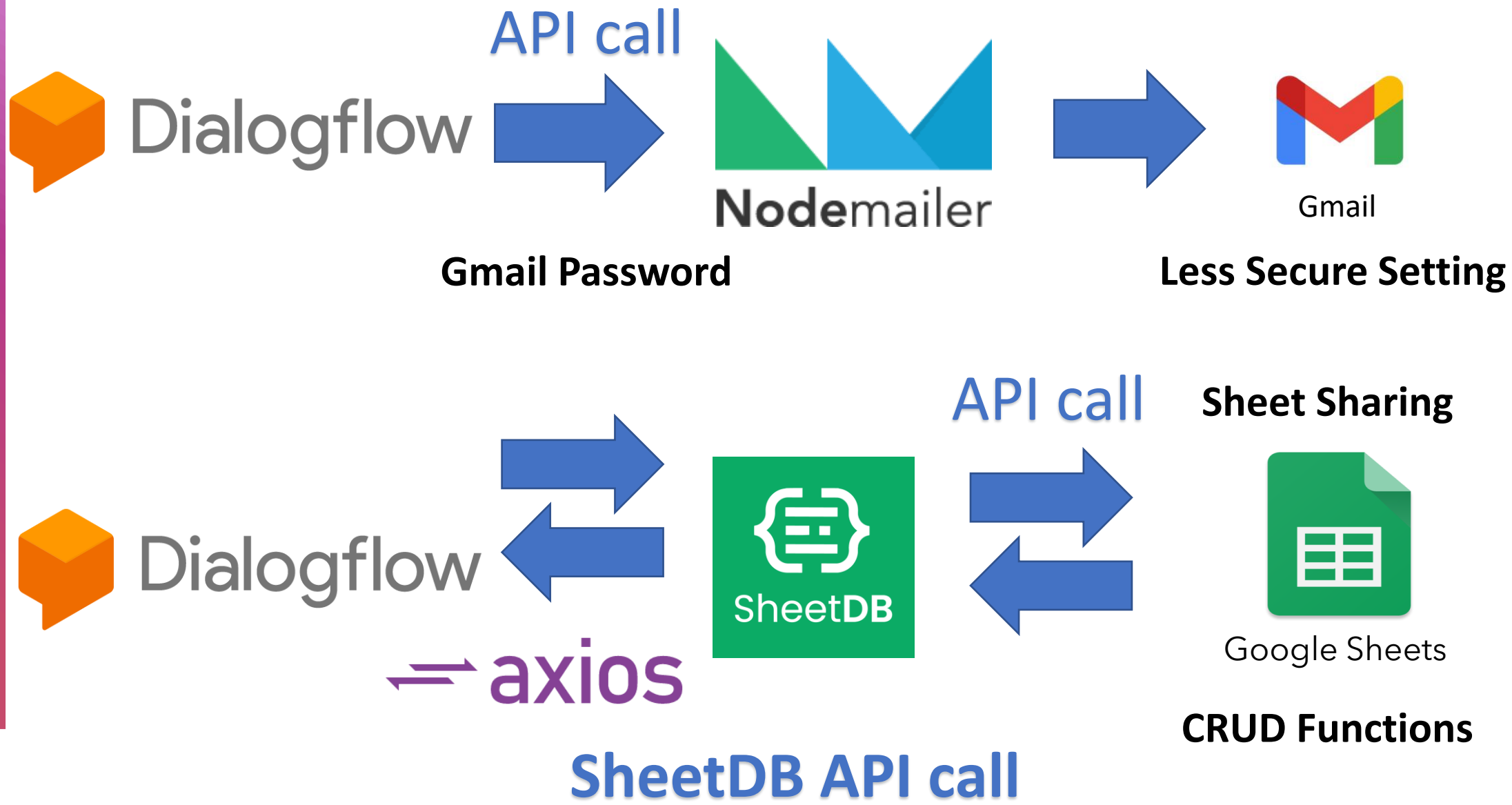
```
402
403     function UpdateCalendar(CalendarId, EventId, CustComment) {
404         return new Promise((resolve, reject) => {
405             calendar.events.patch({
406                 auth: serviceAccountAuth,
407                 calendarId: CalendarId,
408                 eventId: EventId,
409                 resource: {
410                     description: CustComment
411                 }
412             }, (err, event) => {
413                 if (err) { reject(err); }
414                 else { resolve(event); }
415             });
416         });
417     }
418
419     async function CheckSchedule(agent) {
420         const ApptDate = new Date(agent.parameters.ApptDate.split('T')[0] +
            'T' + '10:00:00+08:00');
421         const ApptDateFormat = ApptDate.toLocaleString(
422             'en-GB',
423             { weekday: 'long', day: 'numeric', month: 'long', year: "numeric"
                }
```

UPDATE



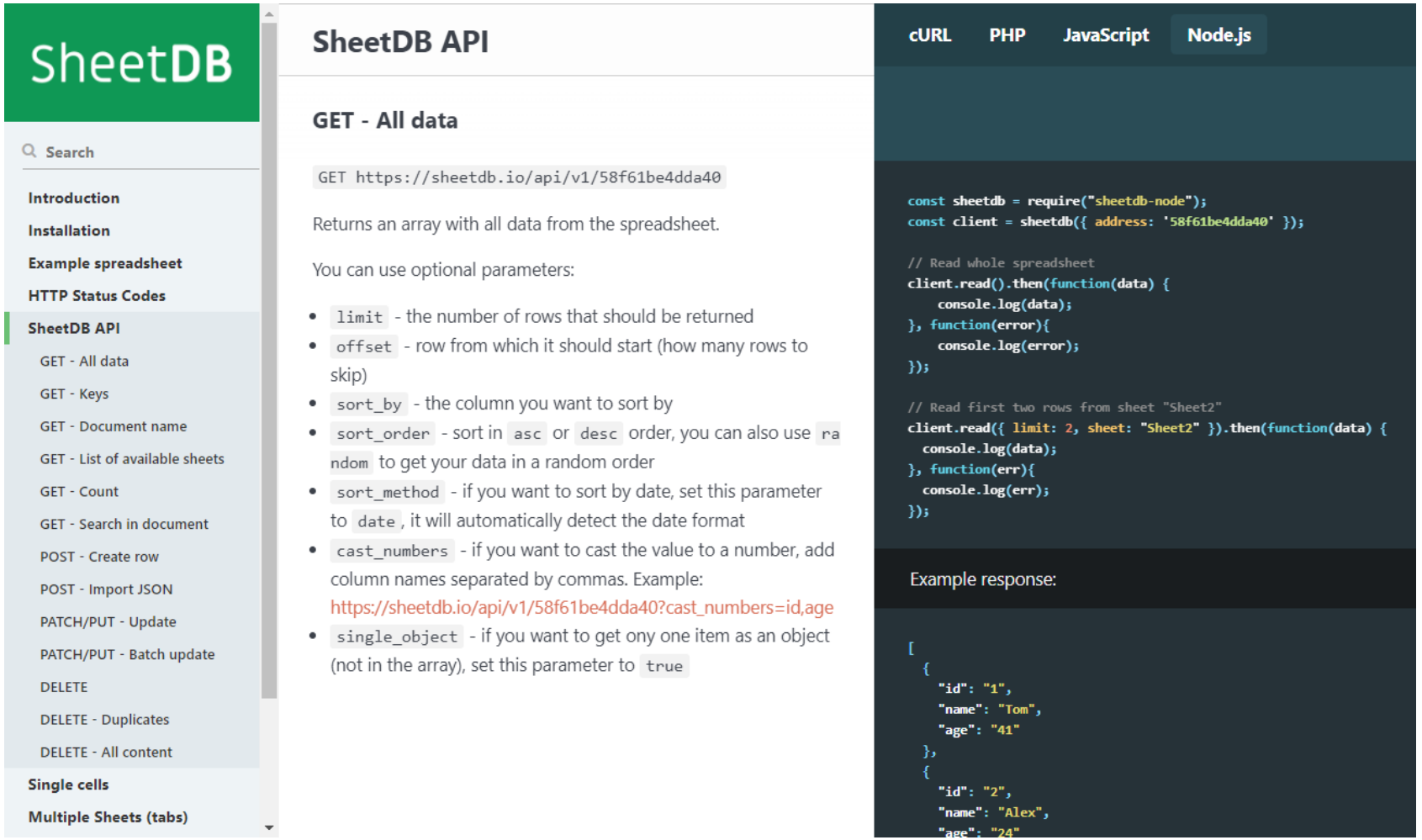
Google Sheets & SheetDB: CRU Functions

Third-Party Intermediary API Call



SheetDB API Call

+ SheetDB documentation: Structure of Request-Response API Call



Note:

Request & Response can be tested at Postman

https://docs.sheetdb.io/?javascript--node#sheetdb-api

JS Maya Properties Bot V11 For Tutorial.js

```
256
257     function CreateDBEntry(AgentName, ApptDate, ApptTime, CustName,
        CustMobile, CustEmail, CustComment, EventId) {
258         axios.post(EntryDBPath, {
259             "data": {
260                 "AgentName": AgentName,
261                 "ApptDate": ApptDate,
262                 "ApptTime": ApptTime,
263                 "CustName": CustName,
264                 "CustMobile": CustMobile,
265                 "CustEmail": CustEmail,
266                 "TimeStampC": new Date(),
267                 "Status": "ACTIVE",
268                 "CustComment": CustComment,
269                 "EventId": EventId
270             }
271         });
272     }
273
274     async function DeleteAppt(agent) {
275         var ApptDB = await GetDBData(EntryDBPath);
276         const CustName = agent.parameters.CustName;
277         const CustMobile = agent.parameters.CustMobile;
278
```

CREATE

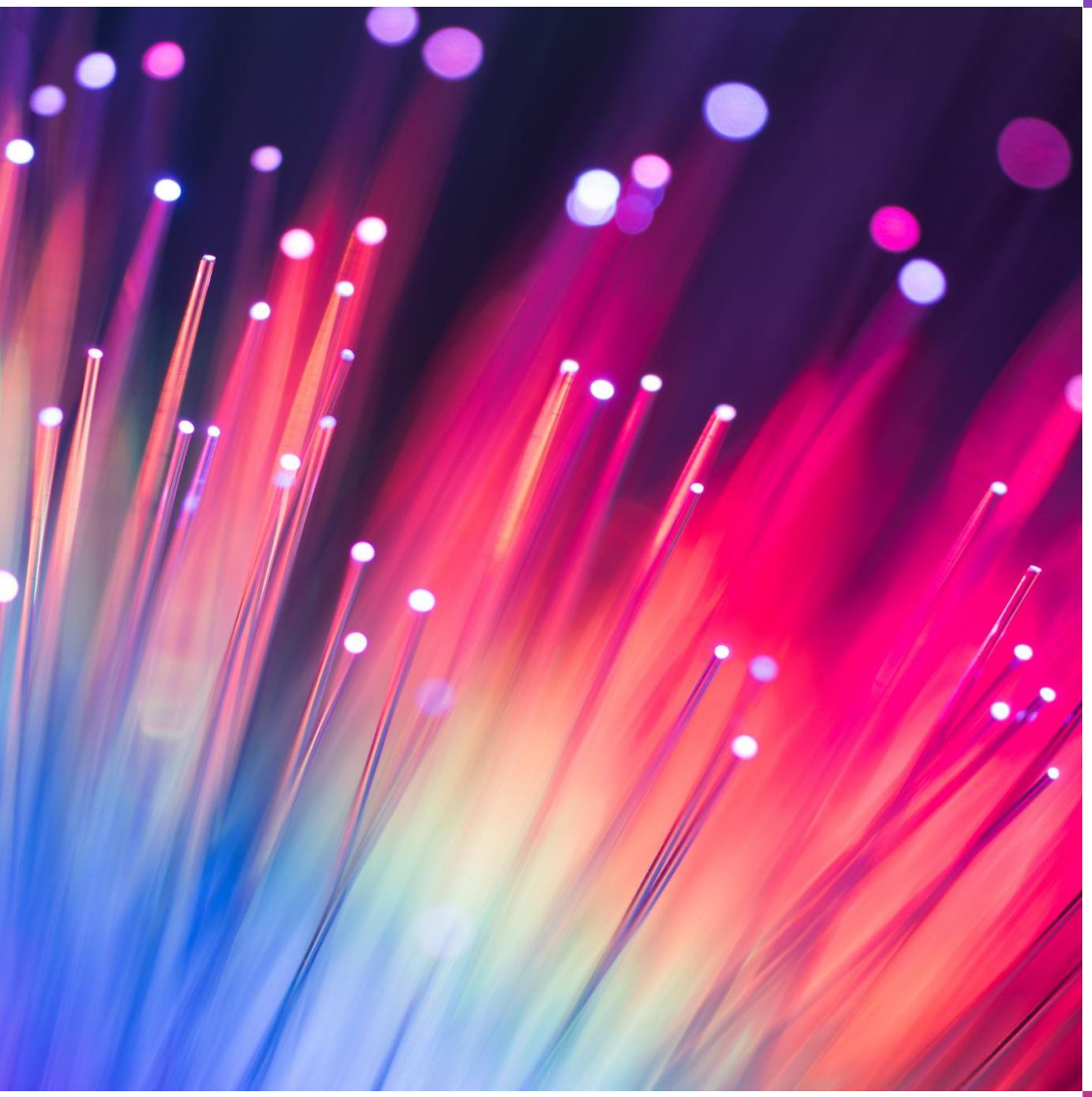
JS Maya Properties Bot V11 For Tutorial.js

```
241
242     function GetDBData(DBPath) {
243         return new Promise((resolve, reject) => {
244             axios.get(DBPath).then(
245                 (response) => {
246                     var result = response.data;
247                     console.log('Processing Request');
248                     resolve(result);
249                 },
250                 (error) => {
251                     reject(error);
252                 }
253             );
254         });
255     }
256
257     function CreateDBEntry(AgentName, ApptDate, ApptTime, CustName,
        CustMobile, CustEmail, CustComment, EventId) {
258         axios.post(EntryDBPath, {
259             "data": {
260                 "AgentName": AgentName,
261                 "ApptDate": ApptDate,
262                 "ApptTime": ApptTime,
263                 "CustName": CustName,
```

READ

```
304
305     function ConfirmDelete(agent) {
306         const CustName = agent.parameters.CustName;
307         const CustEmail = agent.parameters.CustEmail;
308         const CustMobile = agent.parameters.CustMobile;
309         const ApptDateFormat = agent.parameters.ApptDate;
310         const ApptTimeFormat = agent.parameters.ApptTime;
311         const EventId = agent.parameters.EventId;
312
313         // Confirm Delete Routine Part 1: Update ACTIVE Status to DELETED at
            EntryDB Google Sheet
314         const DeletionPath = EntryDBPath + `/CustMobile/${CustMobile}`;
315         axios.put(DeletionPath, {
316             "data": {
317                 "Status": "DELETED",
318                 "TimeStampD": new Date()
319             }
320         }).then(res => {
321             console.log('Log from Update Delete Status Function ' + res.data);
322         }).catch(err => { console.log(err); });
323         // End of Part 1
324
325         // Confirm Delete Routine Part 2: Send Delete Confirmation Emails to
            Customer and Consultant
```

UPDATE



Check Schedule: Async/Await, Promise & Promise.All

More on API Call

- API Call is Asynchronous function: Time Lag
- Structured as Promise
- Promise returned as fulfilled (resolve or reject value) or error
- Promise handled upon return as Then (fulfilled) or Catch (error)
- Code to take account what can proceed concurrently and what to wait for Promise return of API Call
- Example:

```
sendEmail(CustEmail, EmailSubject1, EmailBody1);
sendEmail(AgentEmail, EmailSubject2, EmailBody2);
```

Two Emails can be sent concurrently

```
UpdateCalendar(CalendarId, EventId, CustCommentM).then((response) ⇒
{console.log('Update Calendar function success ', response);
 agent.add(`Your comment has been passed to the Consultant. ` +
`You can go start or end call`);}).catch((err) ⇒
{console.log('Error in Update Calendar ', err);
 agent.add(`It seems that something is wrong.`);});
```

Update Calendar API Call, use of Then & Catch to await return of API Call

Async/Await, Promise.All, Map

- Async/Await is cleaner alternative to handle return from API Call
- Map is method to transform an array into another array based on a function
- Promise.All takes array of Promises & wait for all promises to return, with values of resolve/reject stored in array
- Example:

```
array1 = [1,2,3,4];
array2 = array1.map(elt ⇒ elt + 1);
```

array2 is [2,3,4,5]

```
const values = await Promise.all([promise1, promise2, ...]);
```

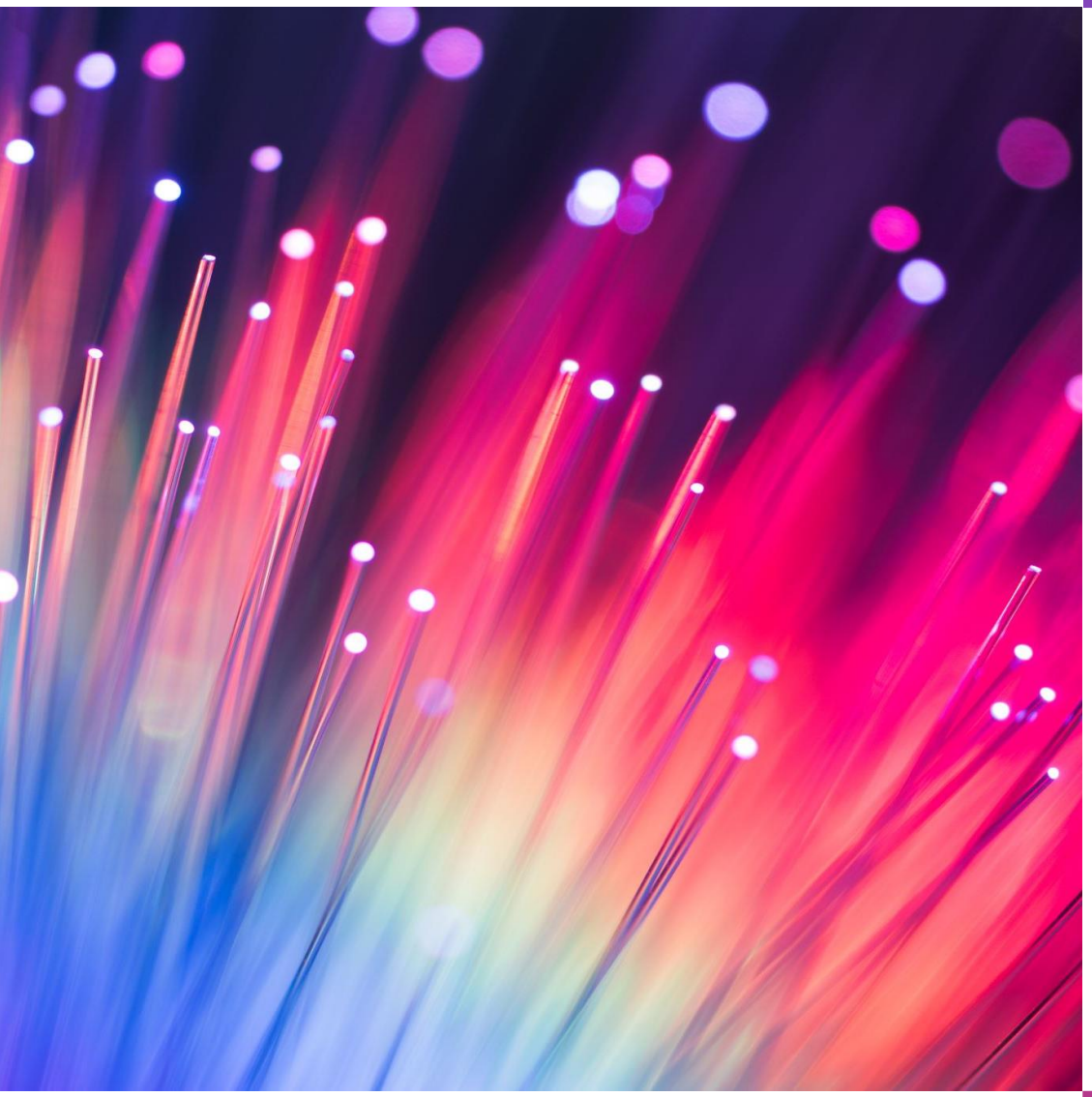
values is array of returned results of promise1, promise2,..

JS Maya Properties Bot V11 For Tutorial.js

```
async function CheckSchedule(agent) {
    const ApptDate = new Date(agent.parameters.ApptDate.split('T')[0] +
    'T' + '10:00:00+08:00');
    const ApptDateFormat = ApptDate.toLocaleString(
        'en-GB',
        { weekday: 'long', day: 'numeric', month: 'long', year: "numeric"
        }
    );

    const counter = [0, 1, 2, 3, 4, 5];
    var PromiseArray = counter.map(item =>
        CheckSlot(new Date(new Date(ApptDate).setHours(ApptDate.getHours
        () + (item * 2))))
    );
    const values = await Promise.all(PromiseArray);
    var OutputString = '';
    var AllNo = true;
    const TimeValues = ['10 am', '12 noon', '2 pm', '4 pm', '6 pm', '8
    pm'];
    for (let i = 0; i < 6; i++) {
        if (values[i] == 'Yes') {
            OutputString = OutputString + ' ' + TimeValues[i] + ',';
            AllNo = false;
        }
```

```
        pm'];
        for (let i = 0; i < 6; i++) {
            if (values[i] == 'Yes') {
                OutputString = OutputString + ' ' + TimeValues[i] + ',';
                AllNo = false;
            }
        }
        if (AllNo) {
            agent.add(`I am so sorry but there are no slots available on ${ApptDateFormat}. You may want to check another day`);
        } else {
            OutputString = OutputString.slice(0, -1);
            if (OutputString.includes(',')) {
                OutputString = OutputString.substring(0, OutputString.lastIndexOf(',')) + ' and ' + OutputString.substring(OutputString.lastIndexOf(',') + 1);
            }
            agent.add(`The available slots for ${ApptDateFormat} are: ${OutputString}. You can make appointment with the Consultant`);

        }
    }

    function CheckSlot(dateTimeStart) {
```

```
451
452     function CheckSlot(dateTimeStart) {
453         const dateTimeEnd = new Date(new Date(dateTimeStart).setHours
            (dateTimeStart.getHours() + 2));
454         return new Promise((resolve, reject) => {
455             calendar.events.list({                    READ
456                 auth: serviceAccountAuth, // List events for time period
457                 calendarId: CalendarId,
458                 timeMin: dateTimeStart.toISOString(),
459                 timeMax: dateTimeEnd.toISOString()
460             }, (err, calendarResponse) => {
461                 if (calendarResponse.data.items.length == 0) { resolve('Yes')
                    ; }
462                 if (calendarResponse.data.items.length > 0) { resolve('No'); }
463                 if (err) { reject('error'); }
464             }
465             );
466         });
467     }
468
469     let intentMap = new Map();
470     intentMap.set('MakeAppt', MakeAppt);
471     intentMap.set('LeaveComment', LeaveComment);
472     intentMap.set('FailureAppt', MakeAppt);
```



Custom Events & Context Parameter Setting from Backend Code

Frontend vs Backend: Intents & Context

Function	Frontend	Backend
Trigger Intents	Intents triggered by matching user input to trained phrases	Trigger specific intents through custom events Call from backend with **agent.setFollowupEvent()** method
Usage	Intents triggered to conduct appropriate actions	Provide means to trigger specific intents upon certain outcomes in backend code. Increases efficiency & better flow
Context Setting	Pass parameters from one intent to another through context	Implemented through **agent.context.set()** method
Usage	Further action needed on same parameters in subsequent intents	Pass parameters computed or obtained in backend code to frontend intents for further action

```
131
132     function MakeAppt(agent) {
133         const CustName = agent.parameters.CustName;
134         const CustEmail = agent.parameters.CustEmail;
135         const CustMobile = agent.parameters.CustMobile;
136         const CustComment = 'NIL';
137         const ApptDate = agent.parameters.ApptDate.split('T')[0];
138         const ApptTime = agent.parameters.ApptTime.split('T')[1].substr(0, 14)
            ;
139         const dateTimeStart = new Date(ApptDate + 'T' + ApptTime);
140         const dateTimeEnd = new Date(new Date(dateTimeStart).setHours
            (dateTimeStart.getHours() + 1));
141         const appointment_type = CustName + ' ' + CustMobile + ' ' +
            CustEmail;
142         const ApptDateFormat = ApptDate.split('-').reverse().join('-');
143         const ApptTimeFormat = ApptTime.substr(0, 5);
144
145         // Email Notification to Customer
146         const EmailSubject1 = 'Appointment Confirmation from Maya Properties
            Bot';
147         const EmailBody1 = 'Dear ' + CustName +
148             '<p> Thank you for using Maya Properties.' +
149             '<p> The appointment is confirmed as follows : ' +
150             '<p><strong> Customer Name: </strong>' + CustName +
```

```
144
145         // Email Notification to Customer
146         const EmailSubject1 = 'Appointment Confirmation from Maya Properties
            Bot';
147         const EmailBody1 = 'Dear ' + CustName +
148             '<p> Thank you for using Maya Properties.' +
149             '<p> The appointment is confirmed as follows : ' +
150             '<p><strong> Customer Name: </strong>' + CustName +
151             '<br><strong> Appointment Date & Time: </strong>' +
                ApptDateFormat + ' ' + ApptTimeFormat +
152             '<br><strong> Consultant Name: </strong>' + AgentName +
153             '<p> The consultant will be contacting you shortly on the details
                of the appointment.' +
154             '<p> Have a nice day!' +
155             '<p> From: Maya Properties Bot';
156
157         // Email Notification to Consultant
158         const EmailSubject2 = 'Appointment Confirmation from Maya Properties
            Bot';
159         const EmailBody2 = 'Dear ' + AgentName +
160             '<p> There is a new customer appointment from Maya Properties Bot.
                ' +
161             '<p> The appointment is confirmed as follows : ' +
162             '<p><strong> Customer Name: </strong>' + CustName +
```

JS Maya Properties Bot V11 For Tutorial.js

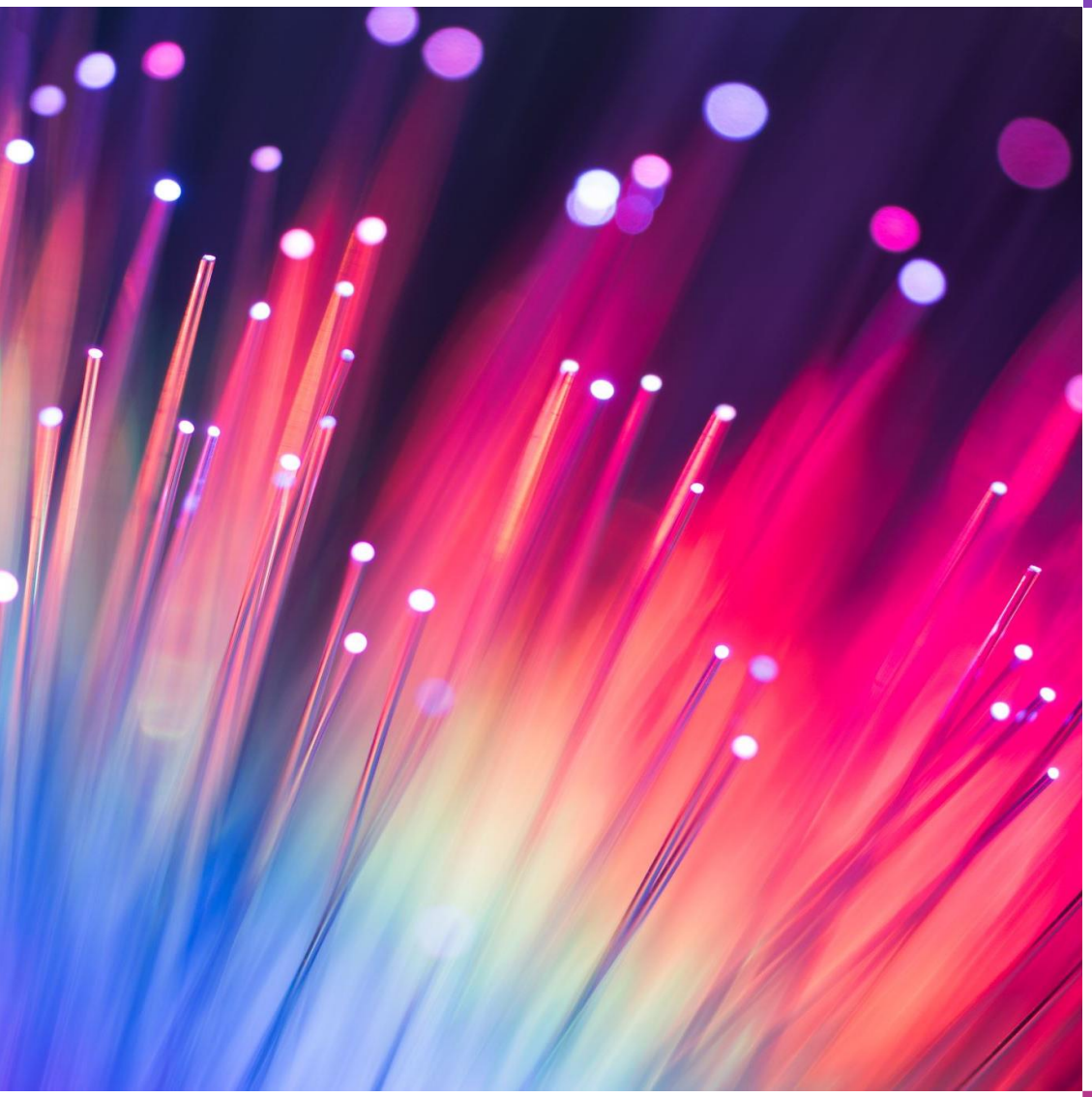
```
156
157         // Email Notification to Consultant
158         const EmailSubject2 = 'Appointment Confirmation from Maya Properties
            Bot';
159         const EmailBody2 = 'Dear ' + AgentName +
160             '<p> There is a new customer appointment from Maya Properties Bot.
                ' +
161             '<p> The appointment is confirmed as follows : ' +
162             '<p><strong> Customer Name: </strong>' + CustName +
163             '<br><strong> Customer Mobile: </strong>' + CustMobile +
164             '<br><strong> Customer Email: </strong>' + CustEmail +
165             '<br><strong> Consultant Name: </strong>' + AgentName +
166             '<br><strong> Appointment Date & Time: </strong>' +
                ApptDateFormat + ' ' + ApptTimeFormat +
167             '<p> Please contact the customer quickly on the details of the
                appointment.' +
168             '<p> Have a nice day!' +
169             '<p> From: Maya Properties Bot';
170
171         // Check the time availibility and make appointment
172         return createCalendarEvent(dateTimeStart, dateTimeEnd,
            appointment_type).then((calendarResponse) => {
173             sendEmail(CustEmail, EmailSubject1, EmailBody1);
174             sendEmail(AgentEmail, EmailSubject2, EmailBody2);
```

JS Maya Properties Bot V11 For Tutorial.js

```
170
171         // Check the time availibility and make appointment
172         return createCalendarEvent(dateTimeStart, dateTimeEnd,
            appointment_type).then((calendarResponse) => {
173             sendEmail(CustEmail, EmailSubject1, EmailBody1);
174             sendEmail(AgentEmail, EmailSubject2, EmailBody2);
175             const EventId = calendarResponse.data.id;
176             agent.context.set({
177                 'name': 'MakeAppt-followup',
178                 'lifespan': 2,
179                 'parameters': {
180                     'EventId': EventId,
181                     'CalendarId': CalendarId
182                 }
183             });
184             CreateDBEntry(AgentName, ApptDateFormat, ApptTimeFormat,
            CustName, CustMobile, CustEmail, CustComment, EventId);
185             agent.add(`The appointment with consultant ${AgentName} on $
            {ApptDateFormat} at ${ApptTimeFormat} has been confirmed. ` +
186                 `The consultant will be contacting you shortly on the details
                of the appointment. Would you like to leave a comment for the
                consultant? ` +
187                 `If no, you can end call`);
188         }).catch(() => {
```

JS Maya Properties Bot V11 For Tutorial.js

```
176             agent.context.set({
177                 'name': 'MakeAppt-followup',
178                 'lifespan': 2,
179                 'parameters': {
180                     'EventId': EventId,
181                     'CalendarId': CalendarId
182                 }
183             });
184             CreateDBEntry(AgentName, ApptDateFormat, ApptTimeFormat,
                CustName, CustMobile, CustEmail, CustComment, EventId);
185             agent.add(`The appointment with consultant ${AgentName} on $
                {ApptDateFormat} at ${ApptTimeFormat} has been confirmed. ` +
186                 `The consultant will be contacting you shortly on the details
                of the appointment. Would you like to leave a comment for the
                consultant? ` +
187                 `If no, you can end call`);
188         }).catch(() => {
189             agent.add(`I'm sorry, there are no slots available for $
                {AgentName} on ${ApptDate} at ${ApptTimeFormat}`);
190             agent.setFollowupEvent('FailureAppt');
191         });
192     }
193
194     function createCalendarEvent(dateTimeStart, dateTimeEnd,
```



Google Assistant Integration

Integrations

Try it now

Google Assistant

Try Actions Builder NEW

Actions Builder visualizes the conversational flow and gives you more control over your users' experience. Try Actions Builder by migrating your Dialogflow project. Learn more

PREVIEW MIGRATION

Not ready yet? Continue with the integration

Please use test console above to try a sentence.

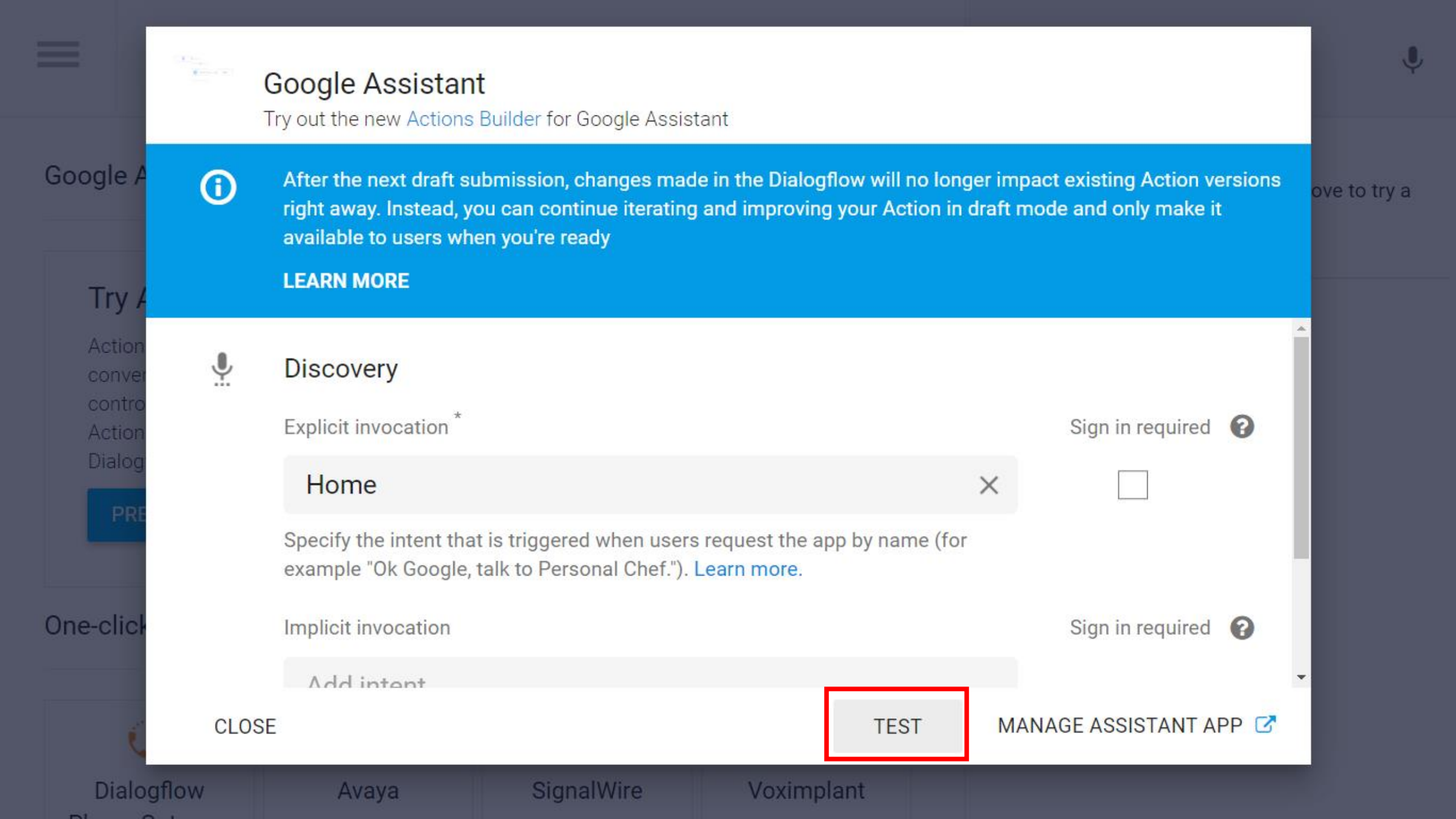
One-click telephony BETA

Dialogflow

Avaya

SignalWire

Voximplant


Google Assistant
Try out the new Actions Builder for Google Assistant
After the next draft submission, changes made in the Dialogflow will no longer impact existing Action versions right away. Instead, you can continue iterating and improving your Action in draft mode and only make it available to users when you're ready
LEARN MORE
Discovery
Explicit invocation *
Sign in required
Home
Specify the intent that is triggered when users request the app by name (for example "Ok Google, talk to Personal Chef."). Learn more.
Implicit invocation
Sign in required
Add intent
CLOSE
TEST
MANAGE ASSISTANT APP

Talk to Maya Properties Bot

Smart Display | English (United States) | Settings

Suggestions

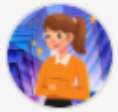
Talk to Maya Properties Bot

Test URL : None

Reset Test | Logs in Google Cloud | Automated tests

Test your Actions

- Use the Simulator to view event logs and session storage
- Type or say "Talk to Maya Properties Bot" on any Assistant enabled device you're logged into with "chungrpa9891@gmail.com"



Maya Properties Bot

Last modified today at 3:29 PM

English

Modify languages in Settings

Quick setup

You've finished all the steps. Well done!

Build your Action

You have finished building Actions. Good job!

Get ready for deployment

You have finished all the steps. Hooray!

Release

You modified Actions, but have not created a release for the new draft.

Manage releases

Manage Alpha testers Manage Beta testers

Create a new release

Officially launch your Action to Google Assistant users. All new beta and production releases go through a review process.

Start new release

Manage Releases

CURRENT RELEASES PAST RELEASES

All channels

Release name	Channel	Review status	Modified by	Last modified
Version 7	Alpha	Not requested	[redacted]	Oct 16, 2021, 9:14:53 PM

Overview Develop Test Deploy Analytics

Manage releases

Manage Alpha testers Manage Beta testers

Create a new release

Officially launch your Action to Google A

Start new release

Manage Releases

CURRENT RELEASES PAST REL

All channels

Release name	Ch			odified
Version 7	Alpha	Not requested	chungrpa9891@gmail.com	Oct 16, 2021, 9:14:53 PM

Create a new release

Version	Channel	Release name
8	Alpha	Version 8

Cancel Submit

[redacted]

Manage releases

Manage Alpha testers | Manage Beta testers

Create a new release

Officially launch your Action to Google Assistant users. All new beta and production releases go through a review process.

Start new release

Manage Releases

CURRENT RELEASES | PAST RELEASES

All channels

Release name	Channel	Review status	Modified by	Last modified
Version 8	Alpha	Not requested	[redacted]	Oct 25, 2021, 3:31:04 PM

Manage releases

Manage alpha testers

Save

Add Alpha testers

Add tester emails separated by commas, then send them the opt-in link below. It may take a few hours for the link to work.

email1@example.com, email2@example.com

3 members

[redacted]

Manage alpha testers

Save

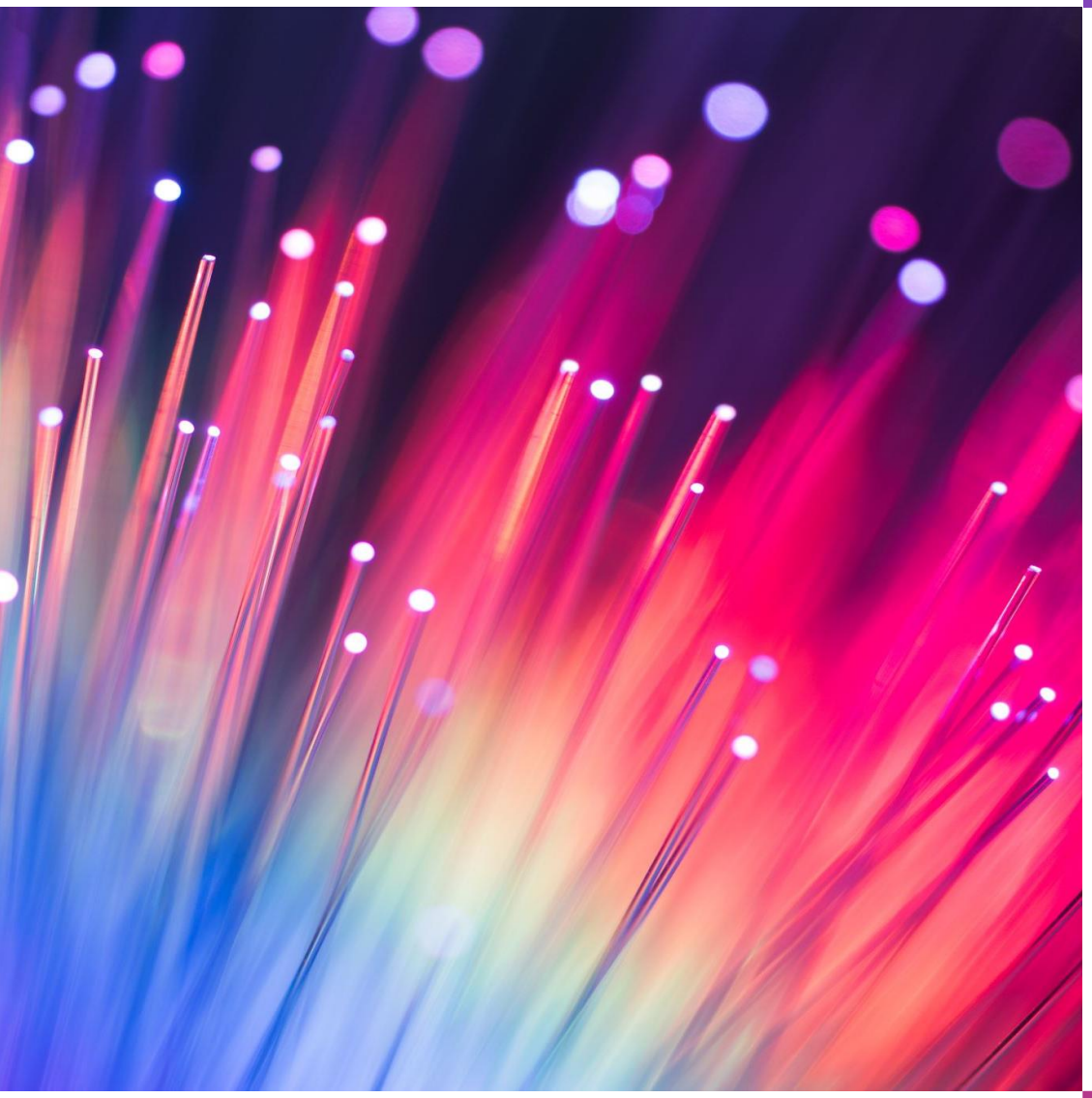
[redacted]

Opt-in link

Share the opt-in link below with your tester

https://assistant.google.com/services/a/uid/00000094d73131b6?hl=e

Please note that your Actions need to be approved before your testers can opt-in and they can opt out anytime. Testers must be signed into Chrome on their mobile device using the email account provided.



Reference Documentation

Intents & Parameters

No	Intent	Intent Type	Parameters	Description
1	Ask FAQ	Main	Nil	Placeholder to FAQ
2	Make Appt	Main	Date, Time, CustName, CustMobile, CustEmail	
3	Failure Appt	Custom	Date, Time, [CustName, CustMobile – Pass by Context]	Trigger by Event follow-up only at Make Appt
4	Leave Comment	Follow-up	Comment, Date, Time, [CustName, CustMobile – Pass by Context]	'Yes' follow-up intent to Make Appt
5	Check Schedule	Main	Date	Multiple API calls to Google Calendar at different time slots to check availability
6	Delete Appt	Main	CustName, CustMobile	
7	Confirm Delete	Follow-up	CustMobile – Passed by Frontend Context Event Id – Passed by Backend Context	
8	Check Home Value	Main	Locality, Property Type, Area Type, Tenure Type, Floor Area	
9	Home	Main	Nil	Upon invocation at Google Assistant
10	End	Main	Nil	Final greetings & end conversation

Parameters/Entities

No	Parameters	Entity Type	Elements	Description
1	Date	@sys.date		Customer date of appointment
2	Time	@sys.time		Customer time of appointment
3	CustName	@sys.any		Customer name
4	CustMobile	@sys.phonenumber		Customer mobile
5	CustEmail	@sys.email		Customer email
6	Locality	Custom Entity	Changi, Orchard, Tampines…	Localities in Singapore [39 Variants]
7	Property Type	Custom Entity	Condo, Apt, Terrace..	6 Variants
8	Area Type	Custom Entity	Landed, Strata	2 Variants
9	Tenure Type	Custom Entity	Freehold, Leasehold	2 Variants
10	Size	@sys.number-int		Floor Area size
11	Event Id			Google Calendar unique identifier string for each appointment, required for update & delete functions