**DOCUMENTATION ON THE CREATION OF INTENTS FOR JASMINE - PAU ADMISSIONS BOT**

Jasmine

She is an agent of Pan-Atlantic University’s Admissions office that aims to help prospective candidates and parents find answers to pressing issues concerning the admission process of the school. Activities that Jasmine can help you with include:  
1. Answering questions

1. Schedule appointments
2. Schedule Tours of the campus
3. Register for scholarships
4. Apply to the school
5. Show the courses available
6. Show accommodations on campus

**Creation of the Agent**

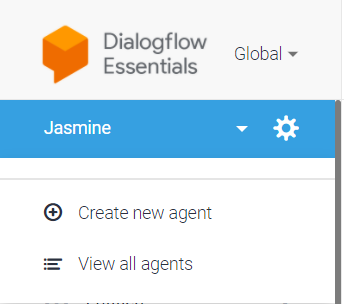
Agents created on Dialogflow platform are easy to start and manage.

1. Log in to dialogflow at <https://dialogflow.cloud.google.com/> and create an account (preferably with a Google account
2. Go to “create new agent” and type your preferred name for your agent. This name can be changed as you go on, but it is more preferred that you stick to the name.

1. You can find your new agent on the top left hand side of the page

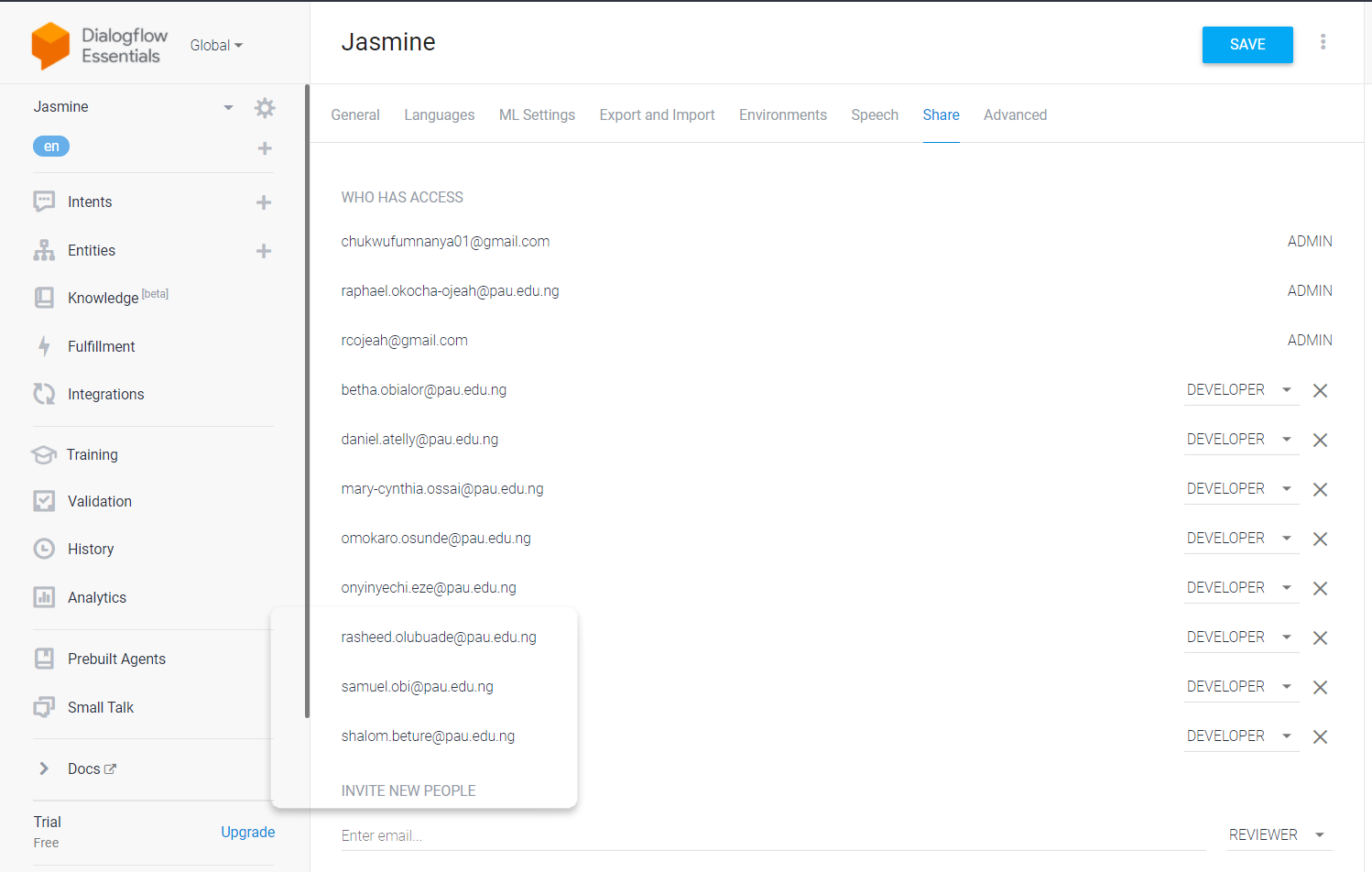
Addition of viewers and developers

To add viewers and developers (group members):



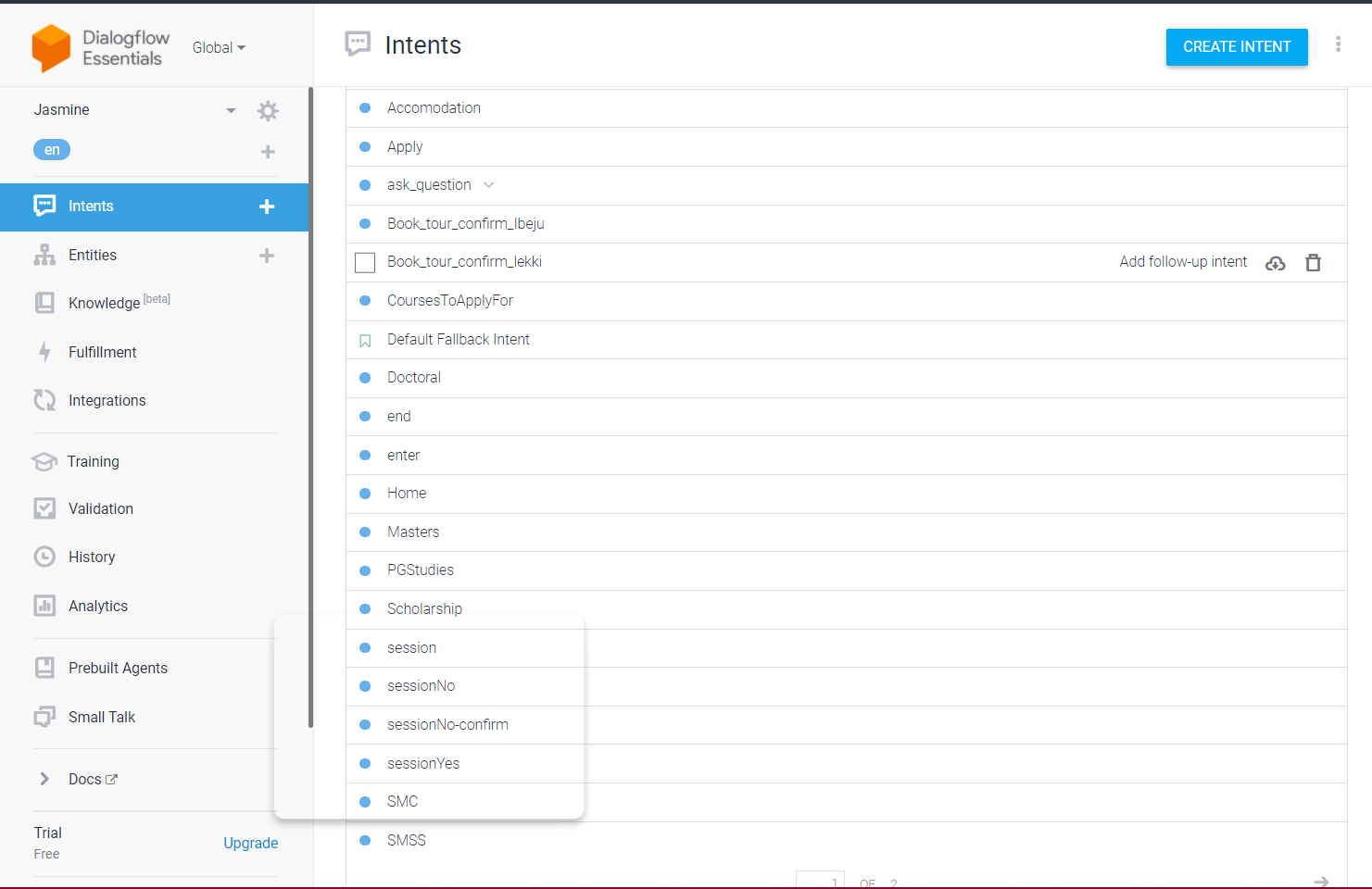
1. Go to the gear icon beside the name of the agent to go to the agent settings
2. Go to “share” and use the emails of the users to add them
3. Indicate whether you would like them as viewers (unable to edit data) or as developers (able to edit some data)
4. Click on “save” to save your work.

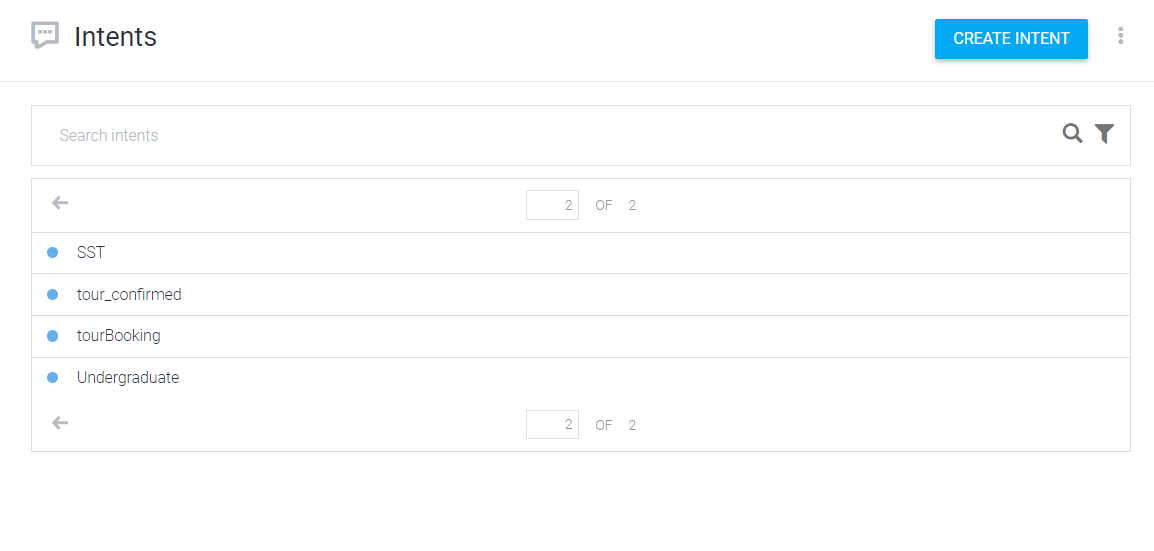
OWNERS NOTE: Here is the list of developers and viewers on the JASMINE project:



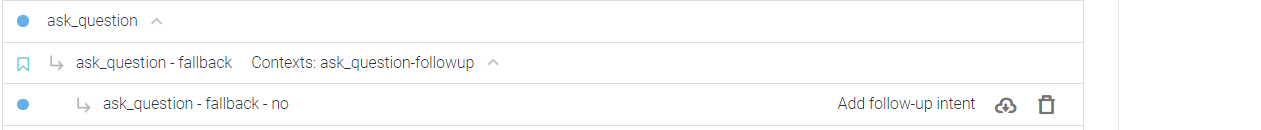
**Intents created**

In all, 24 intents were created by the developers, with 2 fallback intents under one intent ( the question intent. Here is a list of the intents created:



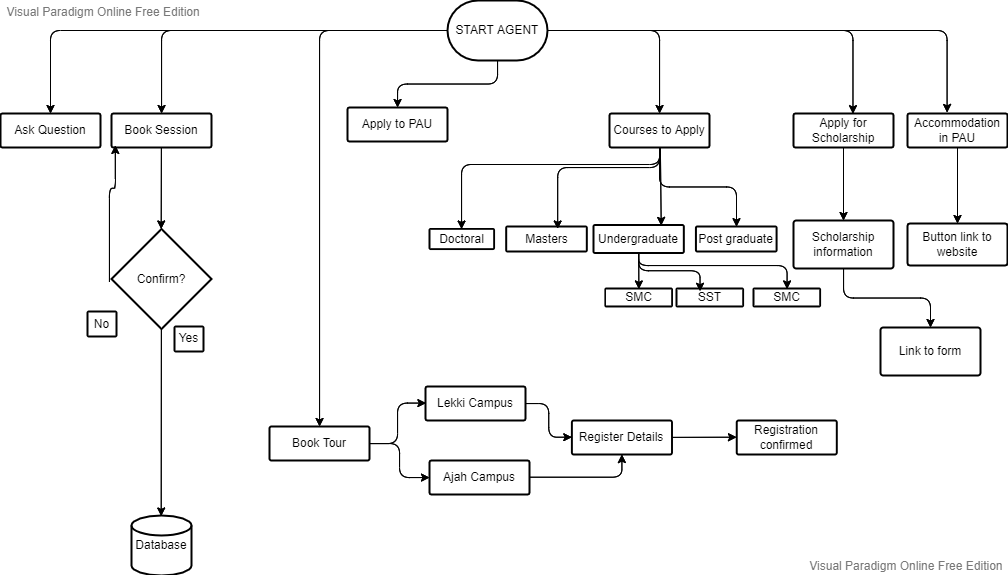


The question intent:

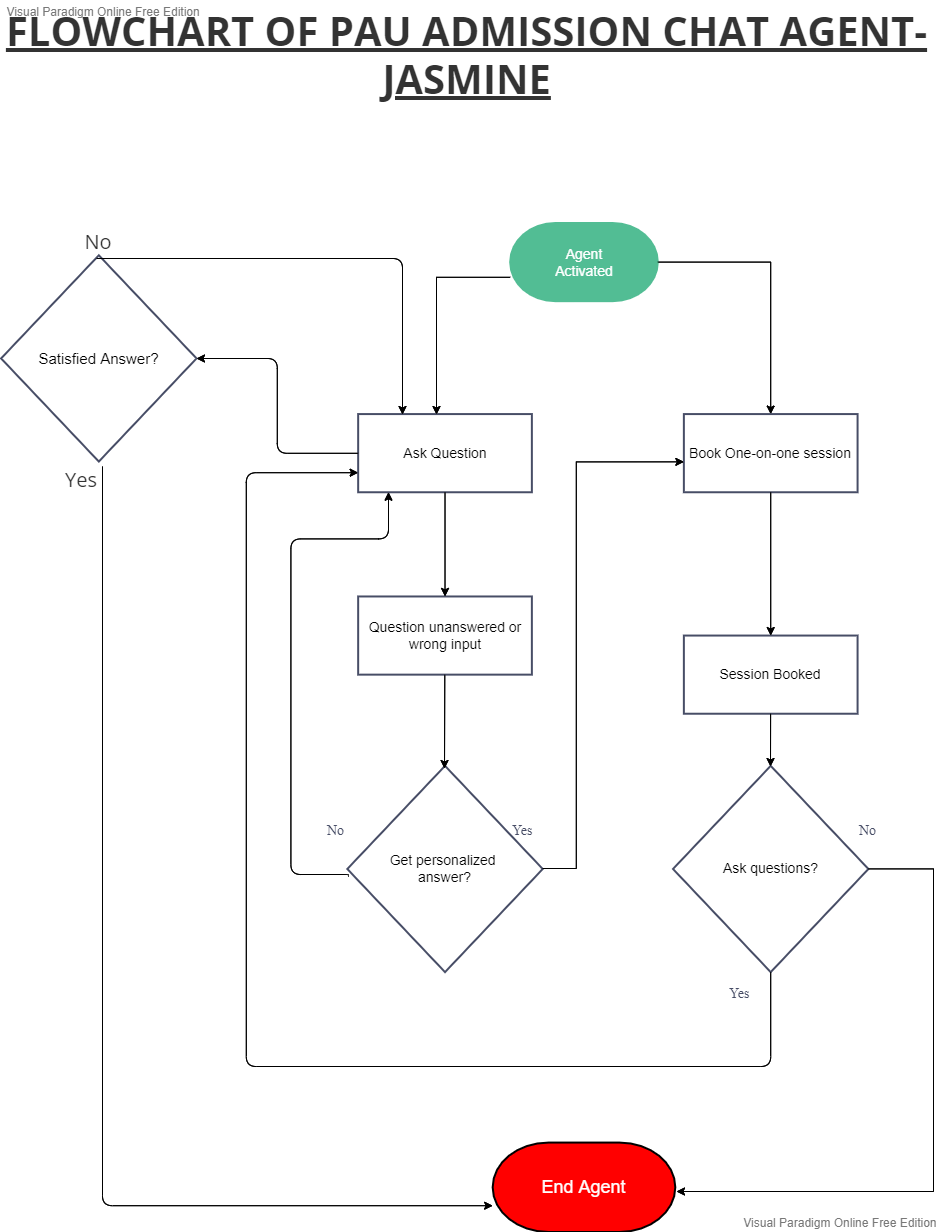


**The relationship between intents:**

The chatbot is built to be an assistant to any new/prospective candidate and their families who would like to know more about the school and engage with different levels of the school. The relationships between intents is as follows:



**Flow Chart Showing “basic intents”**



**Flow Chart Showing “Ask question” and Book session intents**

Other intents used:

1. Home: Attached to every intent, this intent brings the user to a “main menu”, giving the user the ability to get other options after the conclusion of an intent
2. Confirmation intents: Used in the booking tour and booking session intents, these were used to make sure that the user is sure about details sent to us before registering it in the database. If a mistake is made, the agent takes them back to the registration stage.

**User experience enhancement using rich response messages**

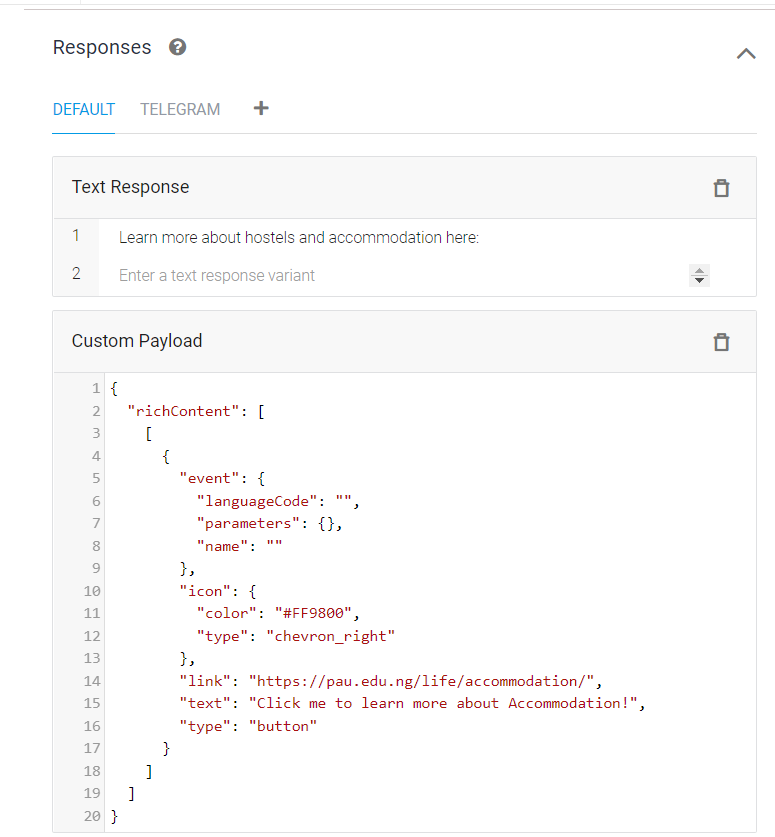
Users (especially new ones) like to be in the know of every process they would like to carry out, especially when it involves chatbots. It is easy to assume that the user knows every process and every response, but that is not the case. With the use of rich response messages, users can see options to move to the next intent in real time - in a stylish manner, of course. These were used in support of the text-based responses, and were placed in the “custom payload” section of the response area in each intent.

**Rich responses used in this agent**

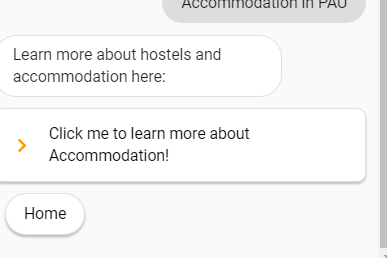
1. **Buttons**

This was used for items that linked the user to an external site.

Code snippet for the button (Accommodation example):



**Implementation of the button:**



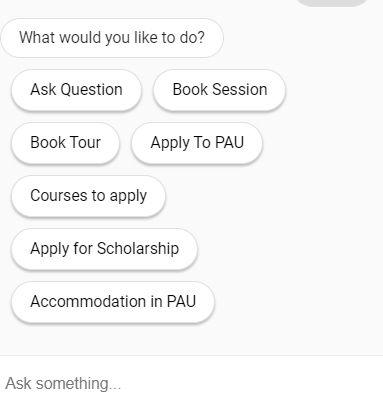
1. **Chips**

This was used to itemize the possible responses that a user could give optimally at that particular intent:

Code snippets showing chips used for the home menu intent



**Implementation:**

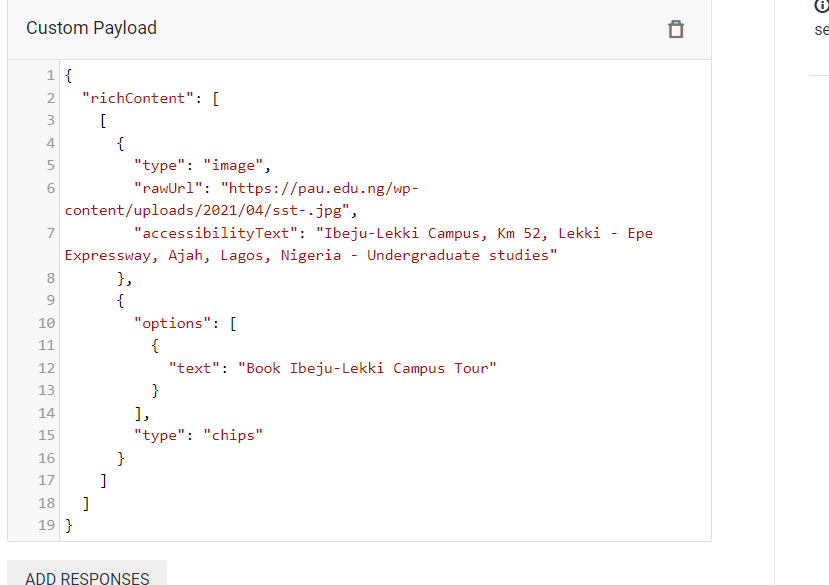


**Clicking on any of the buttons would trigger a response based on the option clicked.**

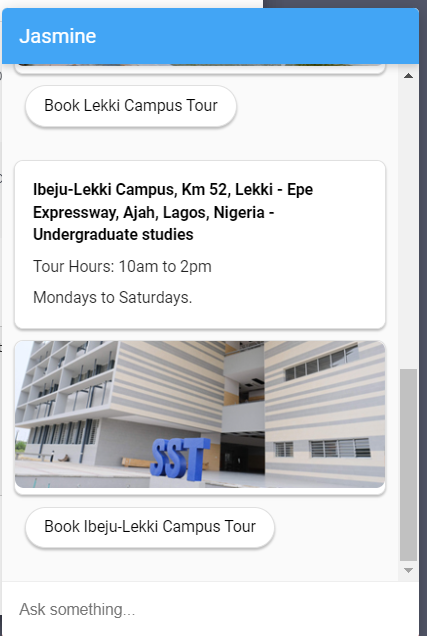
1. **Cards**

These are used to itemize information with images

**Code snippets**



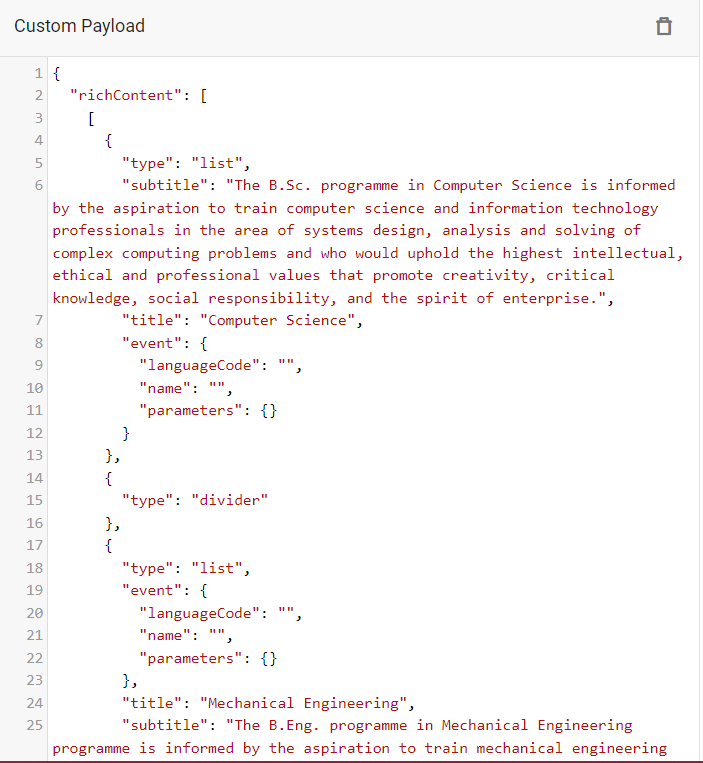
**Implementation:**



1. **Lists**

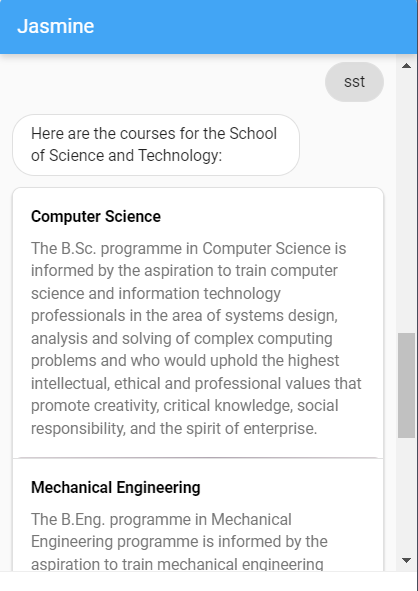
Lists were used to itemize some information regarding courses organized by certain schools.

**Code snippet (For the page on SST):**





**Implementation:**



1. **Logo Image as seen at the agent start**



**Implementation of rich messages on telegram**

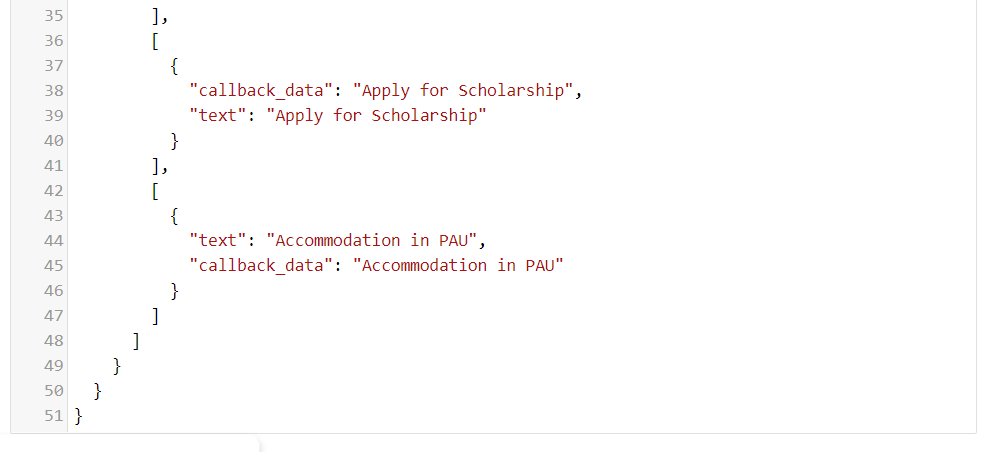
For each implementation to a different platform, the codes have to be replicated in a different format as different platforms have different protocols. For telegram, it cannot read the rich message format of the dialogflow messenger, so new lines of codes have to be written. The different rich messaging styles used on this agent for telegram include:

1. **Buttons**

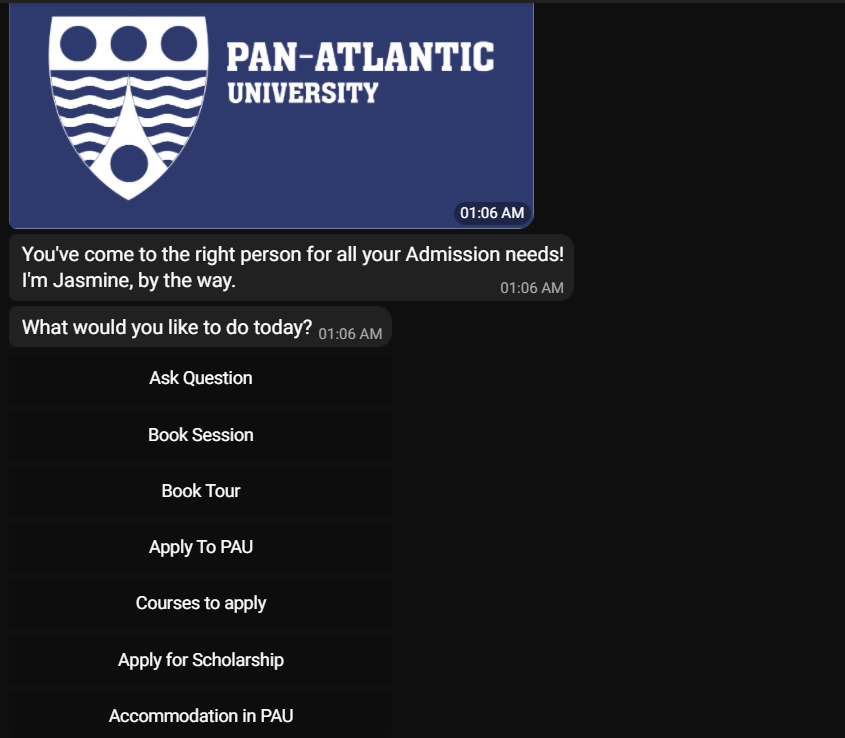
Inline keyboard feature that allows the same functionality as chips

**Code snippet as shown for the start agent stage**

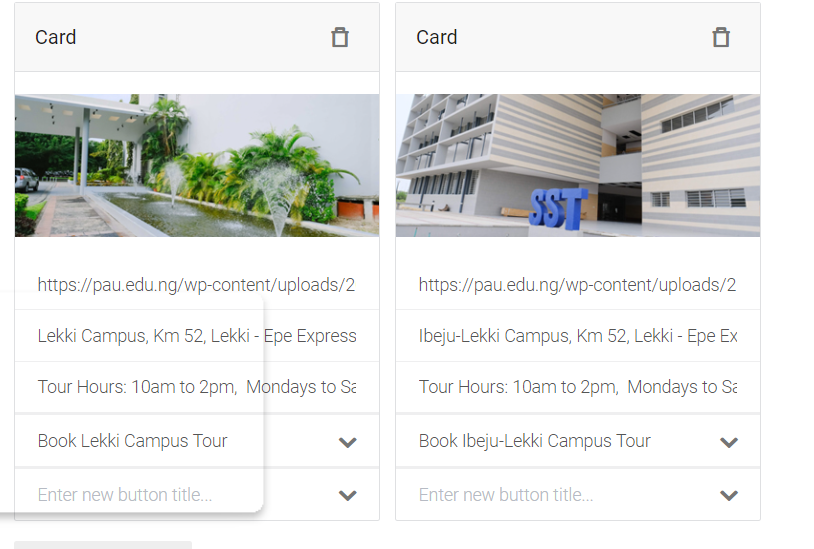




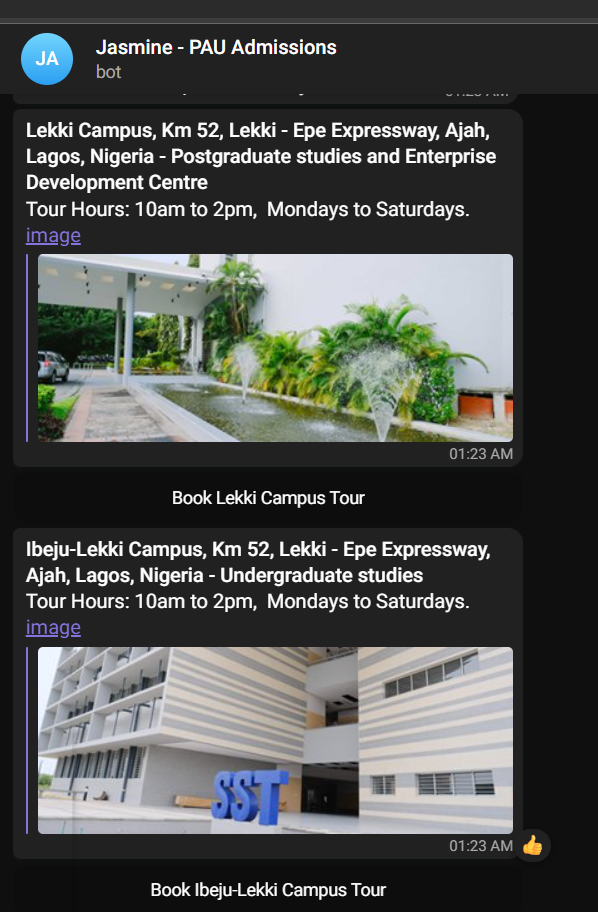
**Implementation:**



1. **Cards**

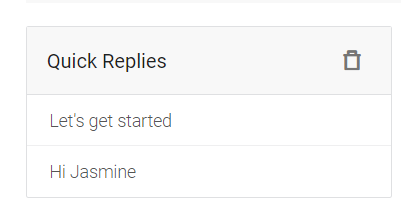


**Implementation:**

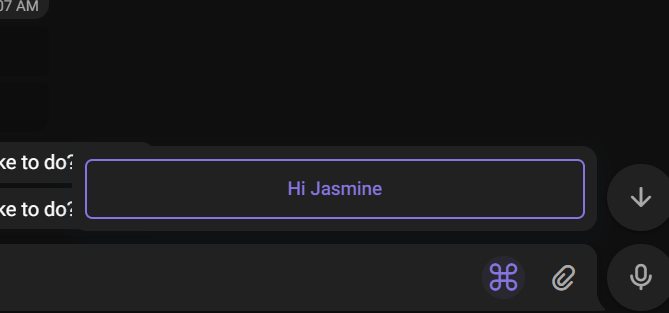


1. **Quick replies**

This was used in this case to ensure that the user can always start the agent from the beginning if a point of confusion arises



**Implementation:**



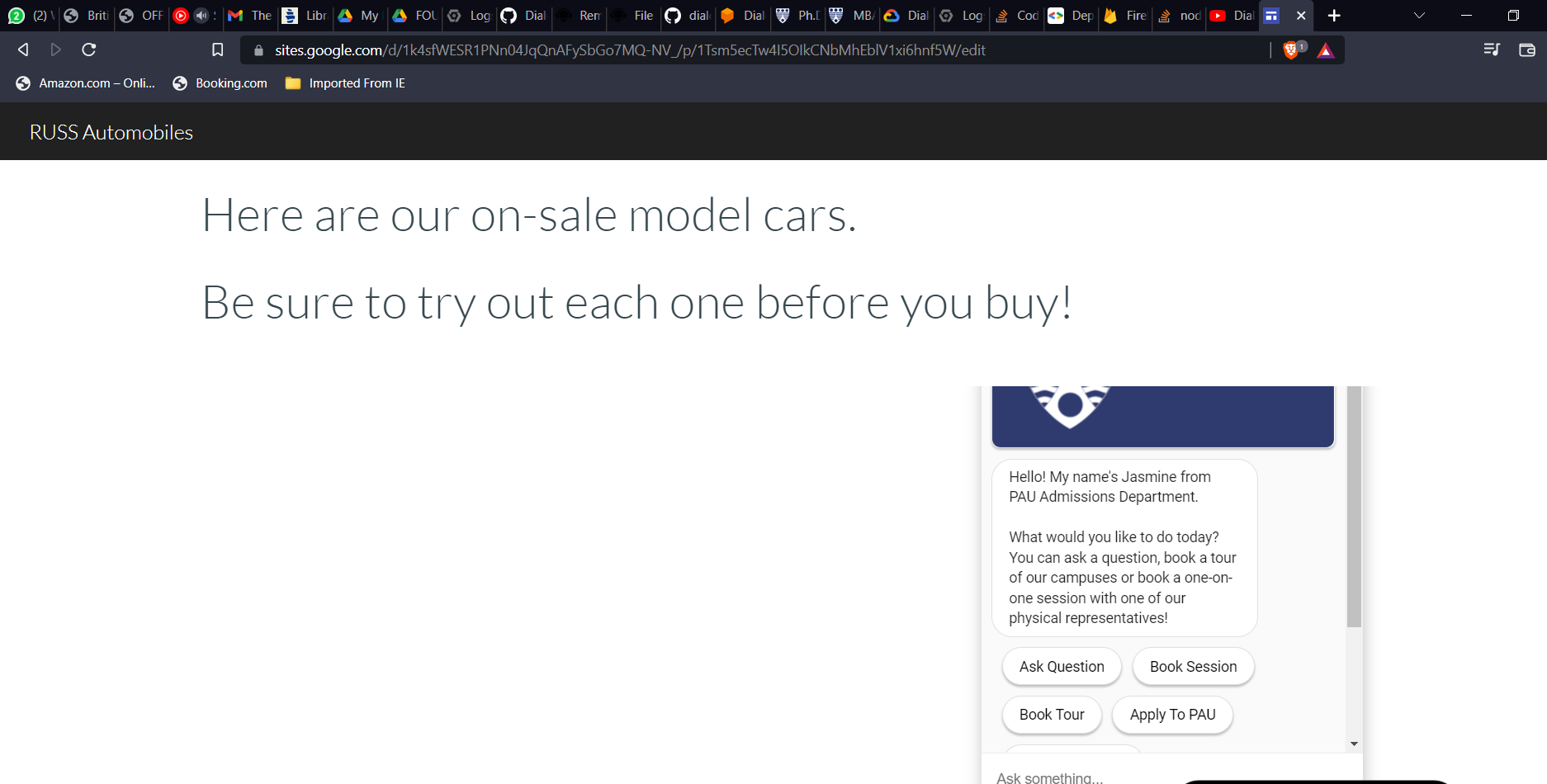
Points to note for telegram responses:

1. Buttons that originally would send users to external links on dialogflow messenger were replaced by hyperlinks
2. Because of limited functionality, the interface on telegram is slightly different from dialogflow messenger.

**Integration**

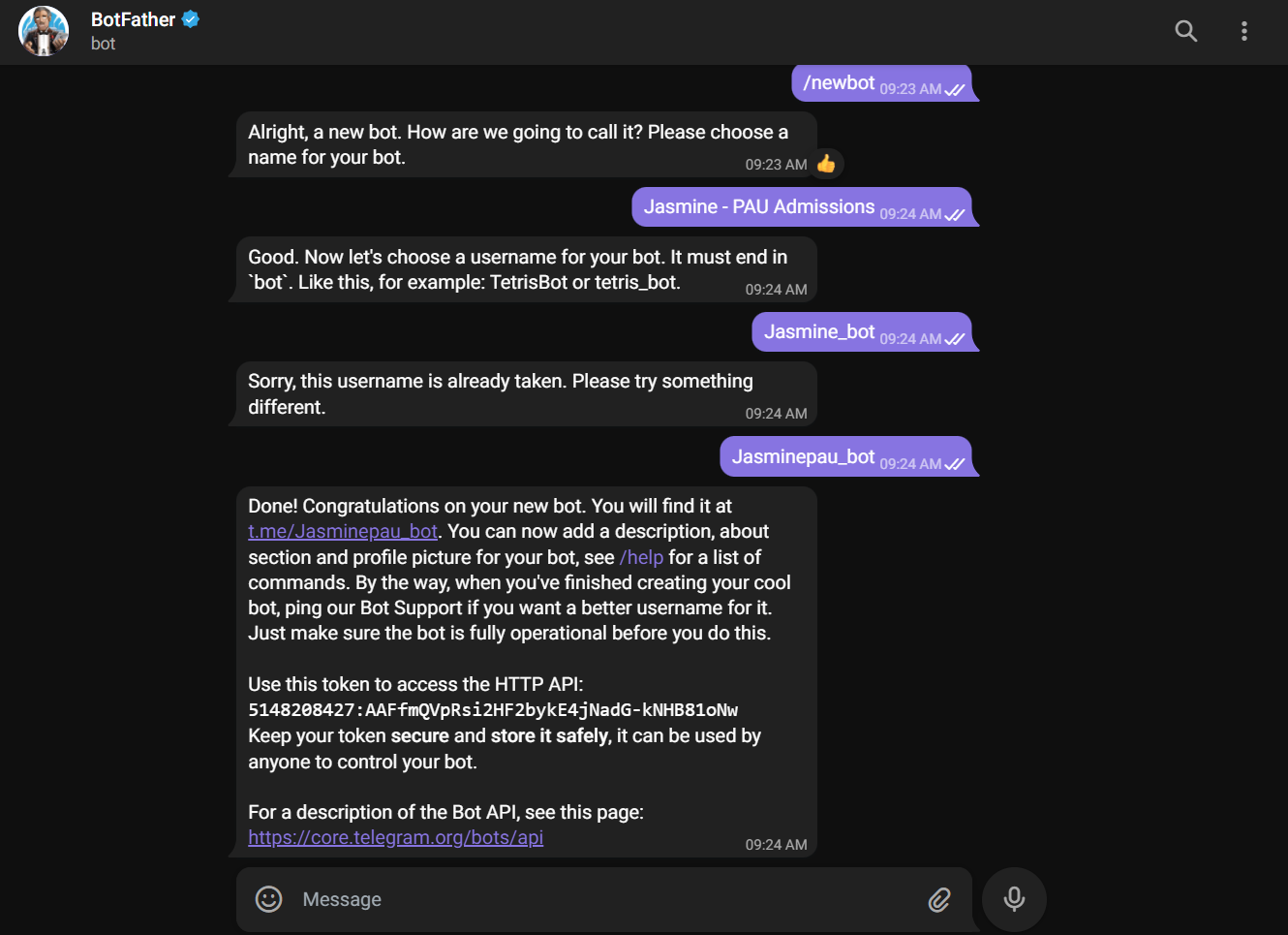
The Agent was integrated on three platforms

1. Dialogflow Messenger
2. Web demo
3. Telegram
4. Dialogflow messenger and the web demo can be accessed and embedded in any website of choice as shown below:

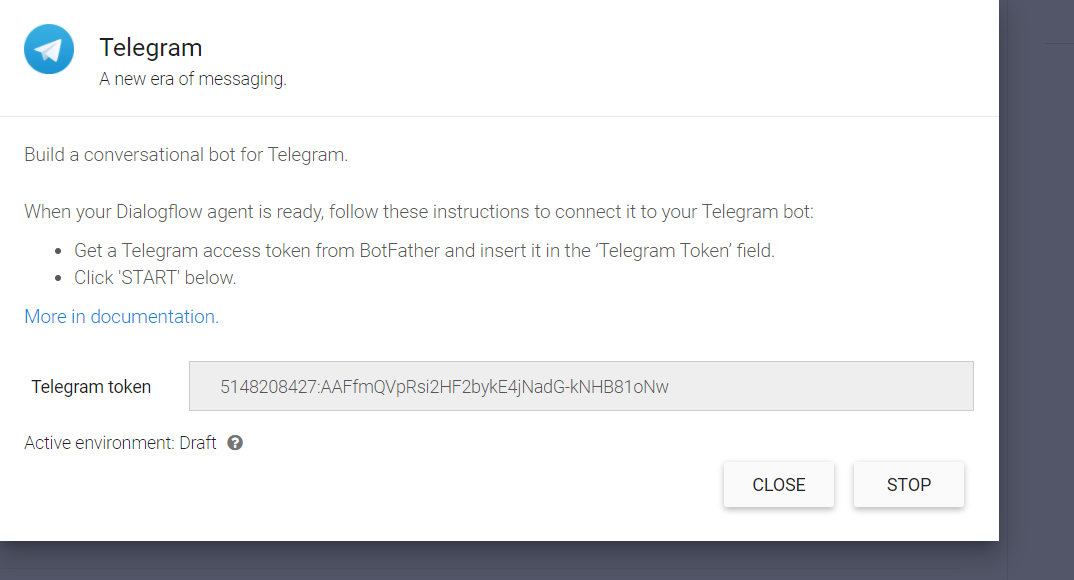


**Image of the agent being used in a model website (Embedded)**

1. **For telegram:**
2. We integrated by registering with the Bot Father:



1. Then used the token to connect to dialogflow:



The username for his bot on telegram is: @Jasminepau\_bot

The link to the bot is: t.me/Jasminepau\_bot

**Fulfillment:**

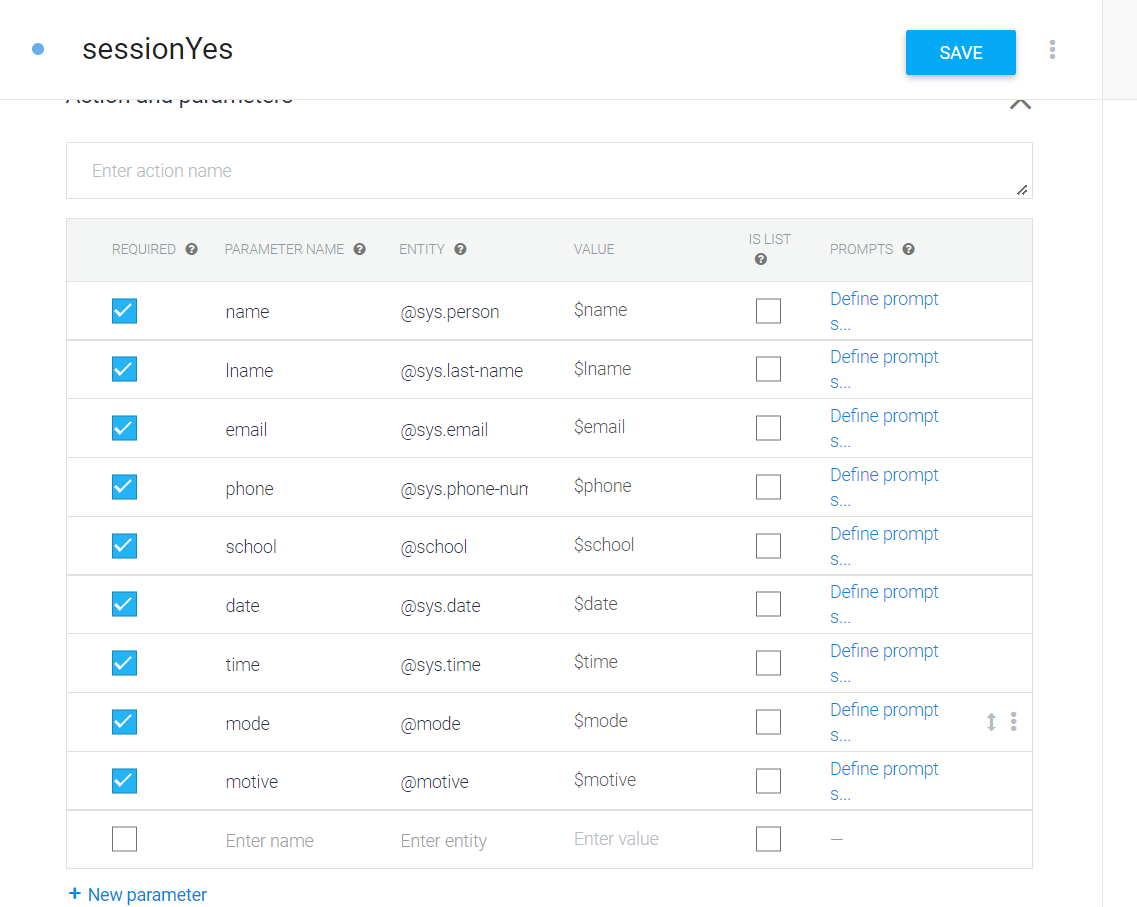
The attribute connected to a database is the “Book session” program in which the applicant or user books a one on one session with an admissions representative. He or she is required to provide some information, after which a confirmation will be sent via email once successful. The parameters asked for is as follows:

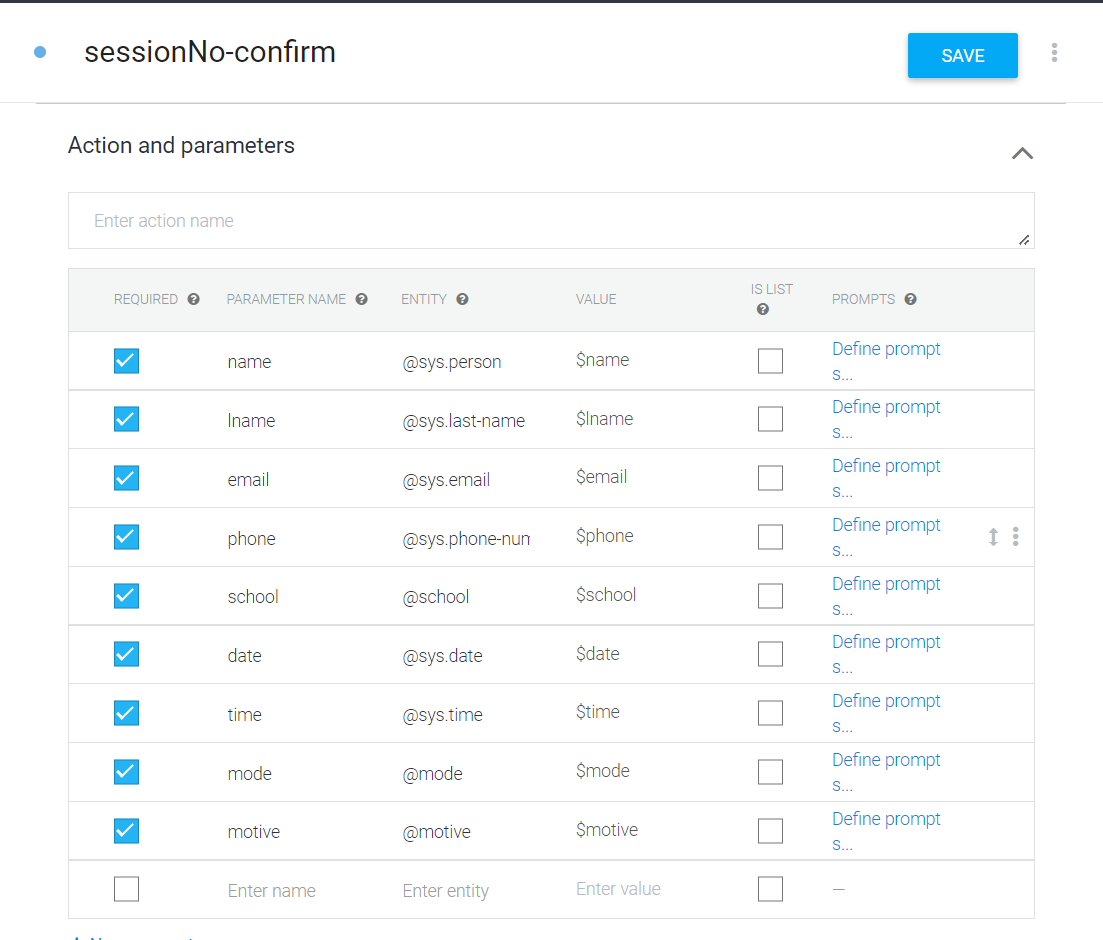
1. Name: This would include the first and last name of user
2. Email
3. Phone number
4. School (current or immediate past)
5. Date and time the user would like to have the meeting
6. Mode of the meeting (online or physical)
7. Motive of the meeting (submission, enquiry or otherwise)

**Setting up parameters in intents**

The connection of the database to the agent happened on the intents where the user confirms the details stated above. This would then prompt the agent to save the details to the database. The two intents connected are:

1. sessionYes, the initial confirmation
2. sessionNoConfirm, the redirected intent in case the user would like to make some adjustments

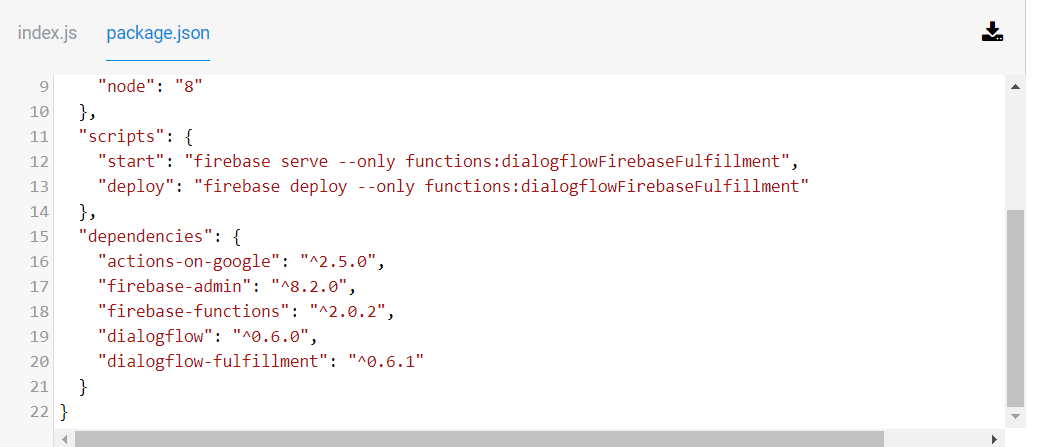




**Parameters used**

After the parameters had been set up, the connection to the database was established through the fulfillment page, where dependencies and parameters were set up in the in-line editor’s index.js and package.json. These would be deployed to the database of our creation, after which we would define the parameters on that database and deploy again so that the agent and the database would be linked successfully.





**Codes for the package.json**

The codes for the index.js are as follows:

***'use strict';***

***const functions = require('firebase-functions');***

***const {WebhookClient} = require('dialogflow-fulfillment');***

***const admin = require('firebase-admin');***

***admin.initializeApp();***

***const ab = admin.firestore();***

***process.env.DEBUG = 'dialogflow:debug'; // enables lib debugging statements***

***exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {***

***const agent = new WebhookClient({ request, response });***

***function newSession(agent) {***

***let name = agent.parameters.name;***

***let lname = agent.parameters.lname;***

***let email = agent.parameters.email;***

***let phone = agent.parameters.phone;***

***let school = agent.parameters.school;***

***let date = agent.parameters.date;***

***let time = agent.parameters.time;***

***let mode = agent.parameters.mode;***

***let motive = agent.parameters.motive;***

***ab.collection ('session') .add ({name:name, lname:lname, email:email, phone:phone,school:school, date:date, time:time, mode:mode, motive:motive });***

***agent.add(`Duly noted, ${name}. You'll receive an email in the next 24 hours on how the session will be conducted.Please try to be available at the selected time, and be open to a reschedule if the time picked is not conducive for our representative.Would you like to do something else, or we can call it a day?`);***

***}***

***function noSession(agent) {***

***let name = agent.parameters.name;***

***let lname = agent.parameters.lname;***

***let email = agent.parameters.email;***

***let phone = agent.parameters.phone;***

***let school = agent.parameters.school;***

***let date = agent.parameters.date;***

***let time = agent.parameters.time;***

***let mode = agent.parameters.mode;***

***let motive = agent.parameters.motive;***

***ab.collection ('session') .add ({name:name, lname:lname, email:email, phone:phone,school:school, date:date, time:time, mode:mode, motive:motive });***

***agent.add(`Duly noted, ${name}. You'll receive an email in the next 24 hours on how the session will be conducted.Please try to be available at the selected time, and be open to a reschedule if the time picked is not conducive for our representative.Would you like to do something else, or we can call it a day?`);***

***}***

***let intentMap = new Map();***

***intentMap.set('sessionYes', newSession);***

***intentMap.set('sessionNo-confirm', noSession);***

***agent.handleRequest(intentMap);***

***});***

**Connected to the Database:**

