



Building a Ticket Booking Bot using Google's Dialogflow

Open Lab | Digital Summit 2019



Build a ticket booking Bot using Google's Dialogflow

Introduction

This document contains the step-by-step process of creating a chat bot with Google's Dialogflow (previously it was called as api.ai) NLP and will teach you how to create a bot and integrate the Chatbot with Facebook Messenger using one click integration in Dialogflow.

This guide was prepared by [Miracle's Innovation Labs](#)

Pre-Requisites

All attendees must have their workstation (with Internet) to participate in the lab (both PC and Mac are compatible). The following pre-requisites will help you to make the Hands-on Lab experience easier.

- A Google account is required for Dialogflow
- Access for Facebook and Facebook for Developers
- Facebook account to create a page for Chatbot integration
- Download and install Node JS and ngrok
- Text Editor such as Visual Studio (or) Notepad++

Technology Involved

- NLP - Google's Dialogflow
- Server Side - Node JS

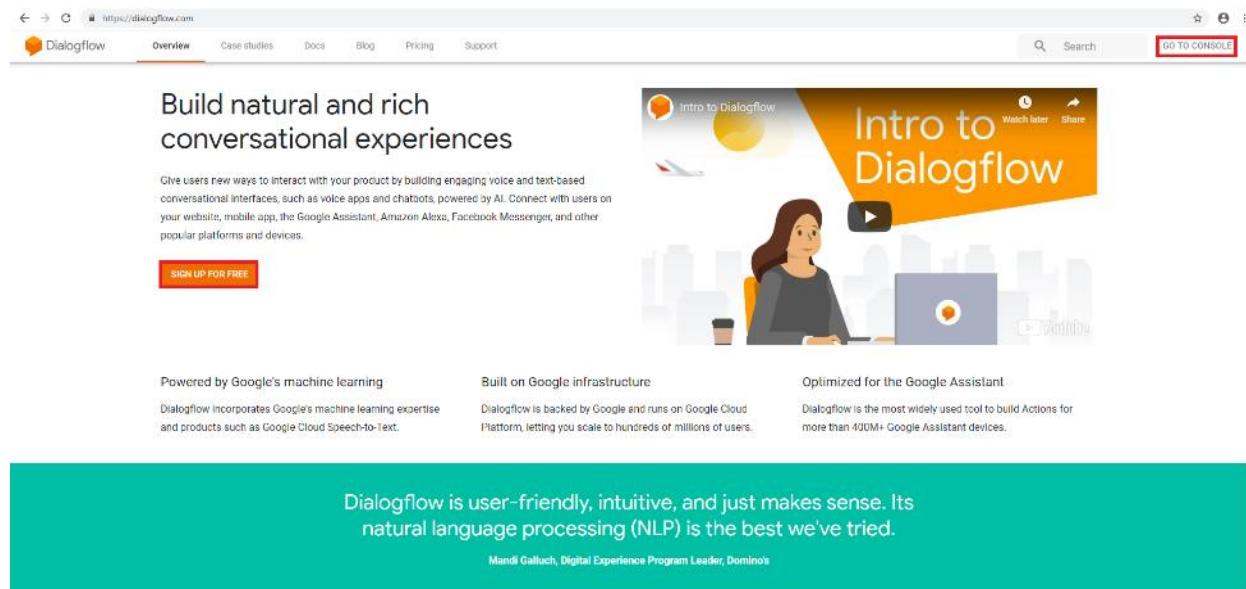
Labs Steps

So, let us get started with the bot!

The following steps will outline how you can create a Ticket Booking Bot using Google's Dialogflow and integrate it with Facebook Messenger. Users will be able to directly message your bot through Facebook Messenger to book your flight ticket.

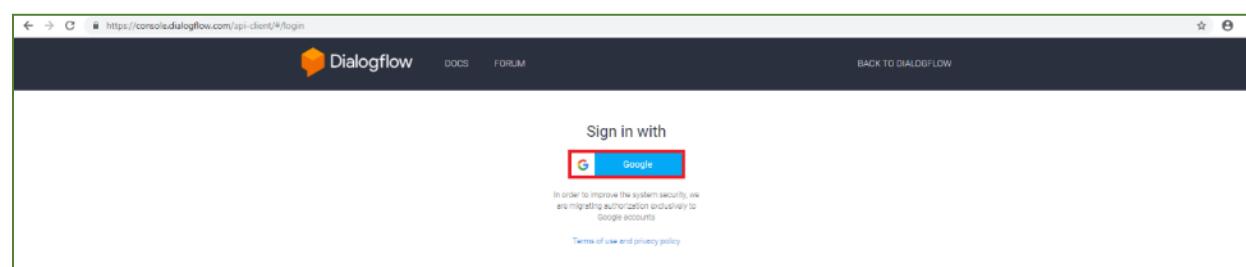
Step #1 | Create Dialog Model in Dialogflow

Create an account for Dialogflow. Go to <https://dialogflow.com/> and click on **GO TO CONSOLE** or **SIGN UP FOR FREE**.



The screenshot shows the official Dialogflow website at <https://dialogflow.com/>. The header includes the Dialogflow logo, navigation links for Overview, Case studies, Docs, Blog, Pricing, and Support, and a search bar. A prominent red button labeled "GO TO CONSOLE" is located in the top right corner. The main content area features a large yellow banner with the text "Intro to Dialogflow" and a video thumbnail. Below the banner, there's a section titled "Build natural and rich conversational experiences" with a "SIGN UP FOR FREE" button. At the bottom, there are three columns: "Powered by Google's machine learning", "Built on Google infrastructure", and "Optimized for the Google Assistant". A testimonial from Mandi Galluch, Digital Experience Program Leader at Domino's, is displayed in a teal box: "Dialogflow is user-friendly, intuitive, and just makes sense. Its natural language processing (NLP) is the best we've tried." The quote is attributed to Mandi Galluch, Digital Experience Program Leader, Domino's.

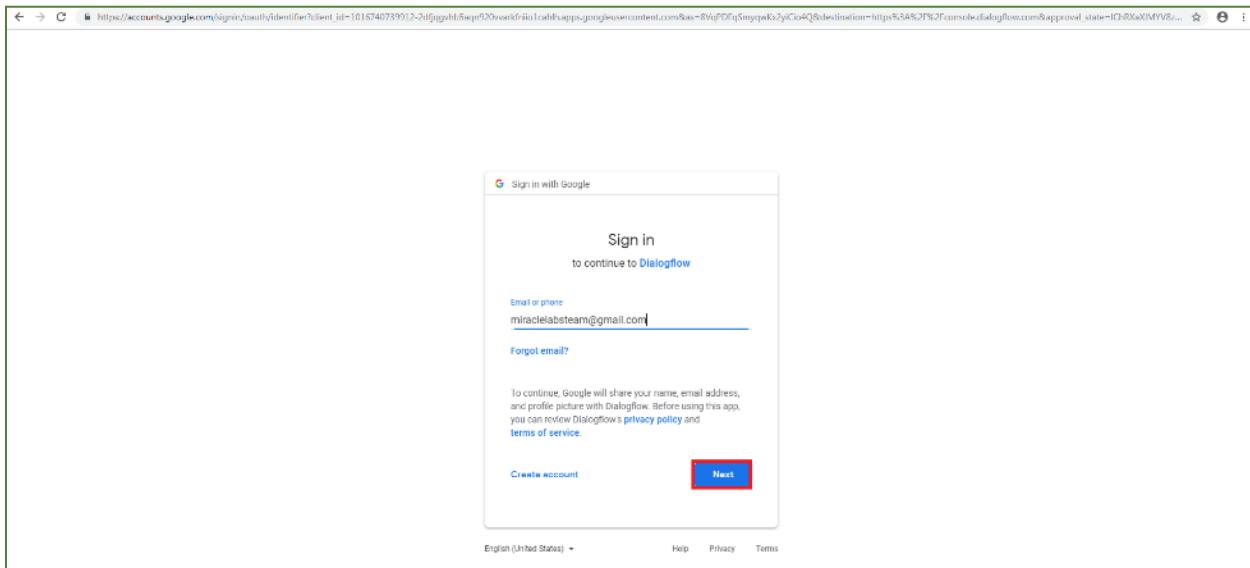
After clicking on any one of the button as shown above, it will be redirected to the console of Dialogflow for login. Now, click on **Google** button.



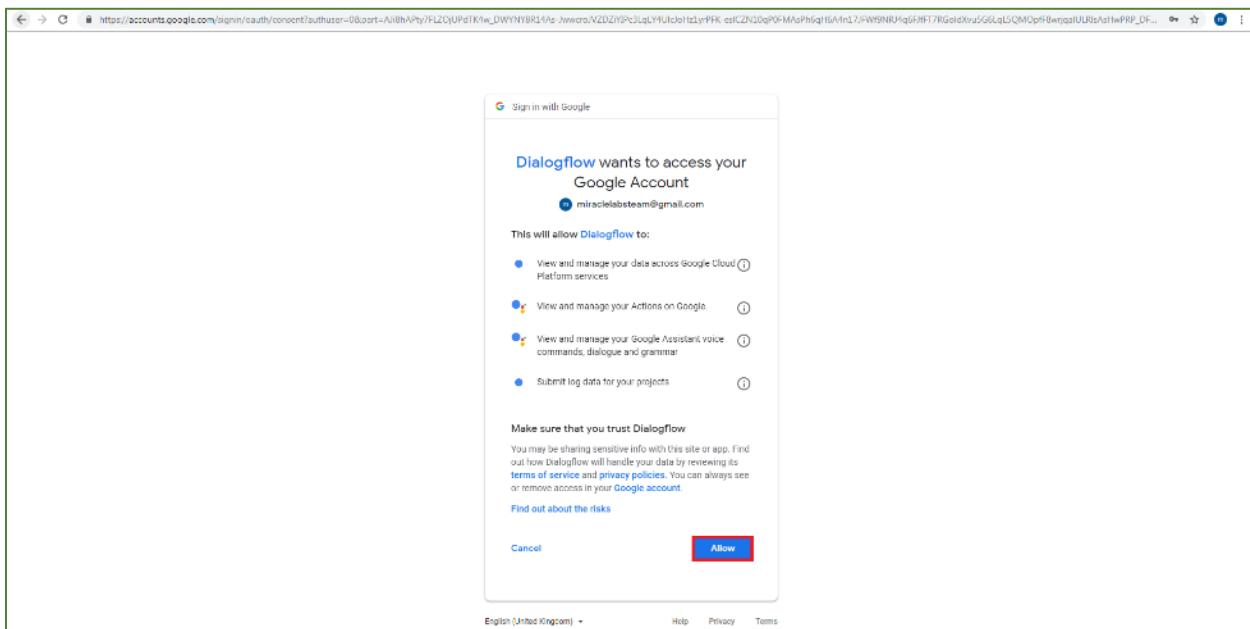
The screenshot shows the "Sign in with" page of the Dialogflow API Client at <https://console.dialogflow.com/api-client/#/login>. It features a "Google" button for sign-in. A note at the bottom states: "In order to improve the system security, we are migrating authorization exclusively to Google accounts." There are also links for "Terms of use" and "Privacy policy". A "BACK TO DIALOGFLOW" link is visible in the top right.

Dialogflow account can be created with Google account credentials. If you have a Google account, use those credentials and authorize yourself.

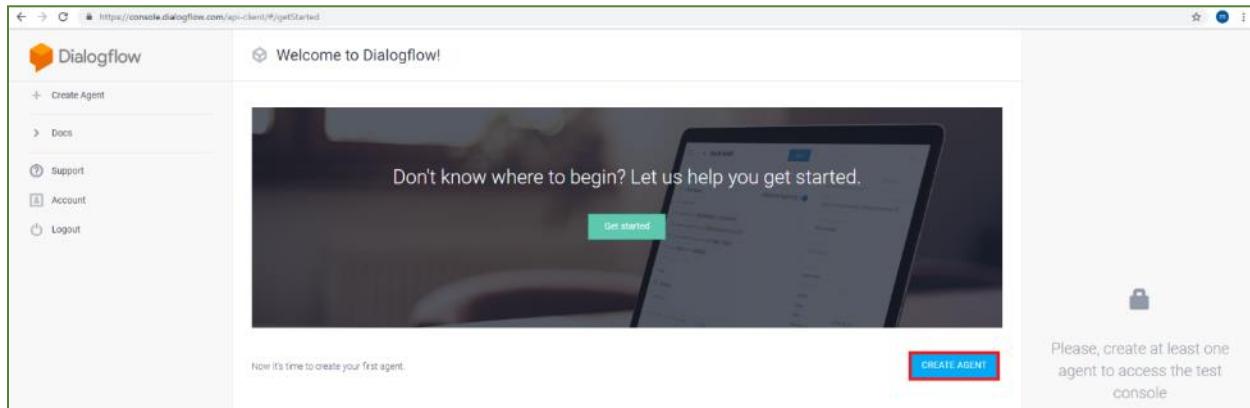
If you don't have a Google account, create it using the below link,
<https://accounts.google.com/SignUpWithoutGmail?hl=en/>



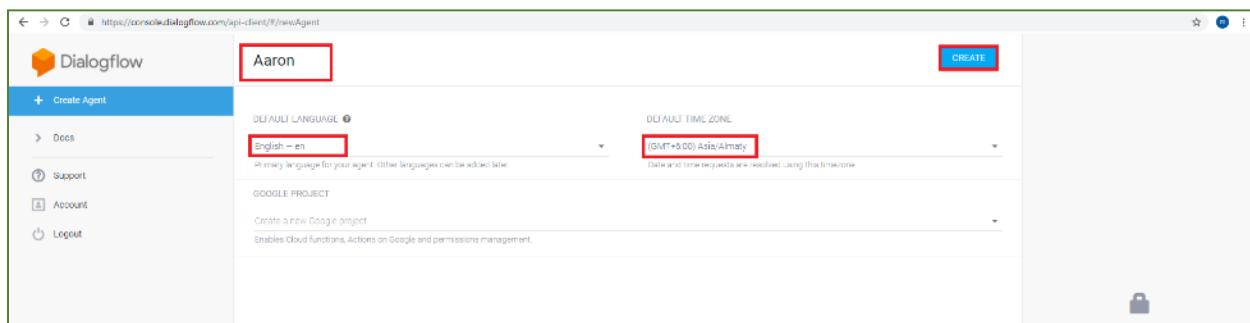
Once you get signed in with your credentials, you will be asked to provide permissions for accessing Dialogflow with your Google Account. To provide access click on **Allow** as shown in the below image.



The Dialogflow home page is opened, where you have to create an Agent/Bot. Click on **CREATE AGENT** (if a pop up opens stating don't have correct permissions just Authorize).



Once you click on **CREATE AGENT**, you will be requested to provide an Agent name. In our case it is **Aaron**, choose the language, and time zone from the drop down respectively and click on **CREATE** button.



Once you are done with the creation of your agent you will get the dashboard with lot of different options on the left hand side which are used for bot creation.

You will also have two Default Intents initially,

- **Default Welcome Intent** - This intent will initially respond to the users, when users engage to the service
- **Default Fallback Intent** - This intent responds back to the user, when input is not recognized

The screenshot shows the Dialogflow interface. On the left, there's a sidebar with a user profile (Aaron, en), navigation tabs (Intents, Entities, Knowledge [beta], Fulfillment, Integrations, Training, Validation [beta], History, Analytics, Prebuilt Agents), and a 'CREATE INTENT' button. The main area is titled 'Intents' and shows a search bar, a list of intents ('Default Fallback Intent', 'Default Welcome Intent'), and a message: 'No regular intents yet. [Create the first one.](#)'. Below it, there's information about intents and a link to 'Prebuilt Agents'.

You can read about these in detail through [Dialogflow docs](#).

By now, we have a basic setup of Dialogflow. Now you have to Create Intent, Entities and Text Response for bot. Let's see one by one.

Step #2 | Creation of Intent

Click on + sign on **Intents** tab on left hand side or click on **CREATE INTENT** button for creating new Intent.

This screenshot is identical to the one above, but the 'Intents' tab in the sidebar is highlighted with a red box, and the 'CREATE INTENT' button at the top right is also highlighted with a red box.

Now you need to provide the **Intent name** and click on **Add Training Phrases** which is located in the **Training Phrases** section for creating the Intent.

The screenshot shows the Dialogflow web interface. On the left, a sidebar menu includes 'Intents' (selected), 'Entities', 'Knowledge [beta]', 'Fulfillment', 'Integrations', 'Training', 'Validation [beta]', 'History', 'Analytics', and 'Prebuilt Agents'. The main area shows an intent card with the title 'Intent name' (highlighted with a red box) and a 'SAVE' button. Below it are sections for 'Contexts', 'Events', and 'Training phrases'. A sub-section titled 'Train the intent with what your users will say' contains a 'ADD TRAINING PHRASES' button (highlighted with a red box). To the right, there's a 'Try it now' button and a note: 'Please use test console above to try a sentence.' Below that is a link to 'See how it works in Google Assistant.'

Note - Under User Expressions, as many expressions you provide the Agent will become that matured.

This screenshot shows the 'Training phrases' section for the 'Greetings' intent. The 'Add user expression' input field (highlighted with a red box) contains the text 'Hello'. Below it, three examples are listed: 'Hola', 'Hey', and 'Hello'. The rest of the interface is similar to the previous screenshot, including the sidebar menu and the 'Try it now' section.

Now, provide the response in the **Responses** section which is used to respond back to the intent when it detects same kind of user examples. For providing response for an intent, click on **Add Response**.

The screenshot shows the Dialogflow web interface. On the left, there's a sidebar with navigation links like Intents, Entities, Knowledge, Fulfillment, Integrations, Training, Validation, History, Analytics, and Prebuilt Agents. The main area is titled 'Greetings' and contains sections for 'Extract the action and parameters' and 'Responses'. The 'Responses' section has a sub-section titled 'Execute and respond to the user' with a 'ADD RESPONSE' button. This 'ADD RESPONSE' button is highlighted with a red box.

Now you can provide multiple responses for the same Intent. Once it is done, click on **SAVE** for saving the created intent.

This screenshot is similar to the previous one but shows the 'Responses' section after adding responses. It includes a 'Text Response' list with two items: '1 Hello, I am Aaron Tourister Bot!' and '2 Enter a text response variant'. Both the 'Text Response' list and the 'SAVE' button are highlighted with red boxes.

Once you provide the response in the Text response section, if your responses from all the intents need to send to the Facebook channel, go to every intent and click on **FACEBOOK MESSENGER** and enable the button to use your text responses as default responses as shown below.

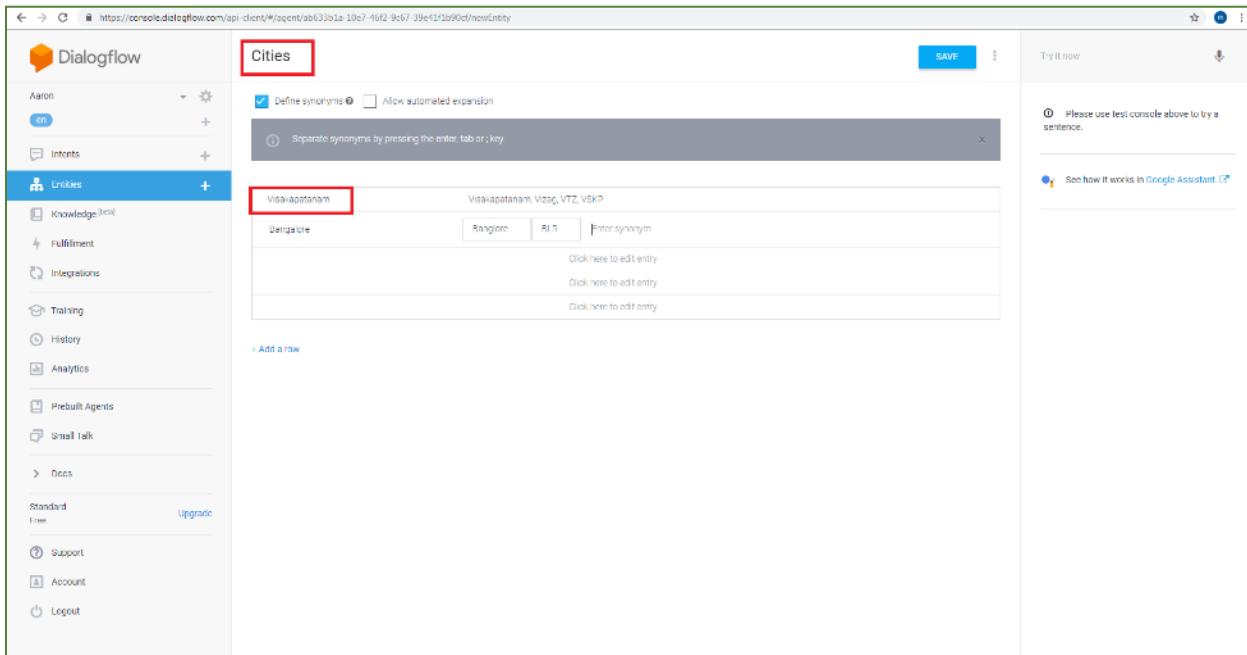
The screenshot shows the Dialogflow web interface. On the left, there's a sidebar with options like Intents, Entities, Knowledge, Fulfillment, Integrations, Training, Validation, History, Analytics, and Prebuilt Agents. The main area shows an intent named 'Greetings'. Under the 'Responses' tab, the 'DEFAULT' tab is selected, and a 'FACEBOOK MESSENGER' button is visible. A red box highlights both the 'FACEBOOK MESSENGER' button and the toggle switch below it, which is used to enable responses for the DEFAULT tab.

Step #3 | Creation of Entities

Click on + sign on **Entities** tab on left hand side (or) click on **CREATE ENTITY** button, which is at top right corner for creating the Entity.

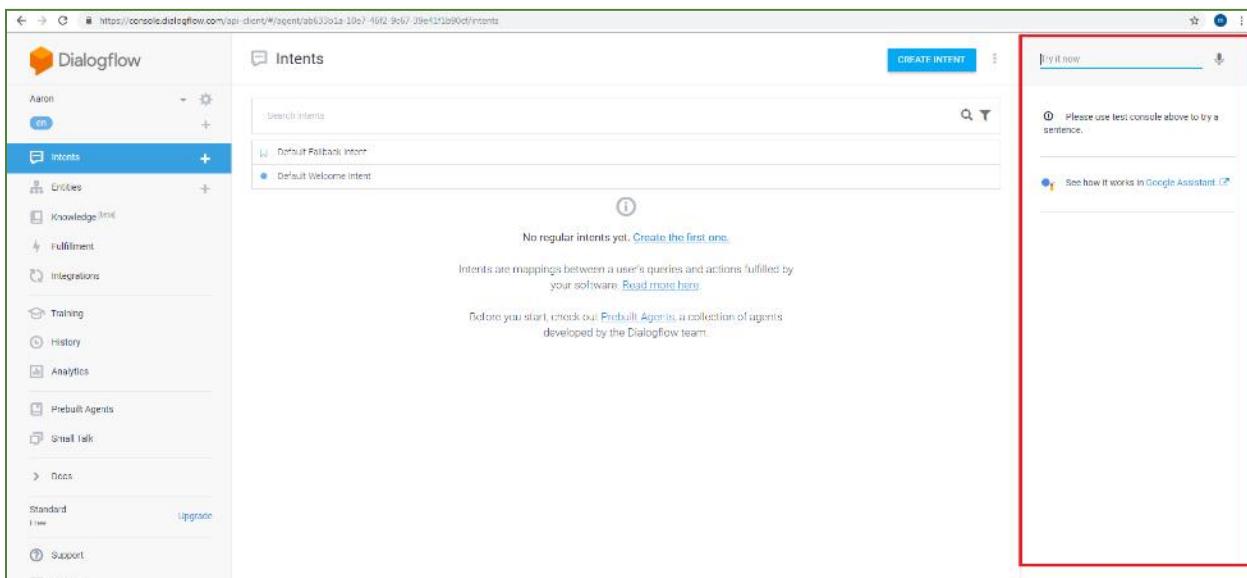
The screenshot shows the Dialogflow Entities page. The sidebar on the left has the 'Entities' tab selected, indicated by a red box. At the top right of the main area, there is a 'CREATE ENTITY' button, also highlighted with a red box. The main content area displays a message stating 'No entities yet. Create the first one.' and information about entities being objects your app or device takes action on.

You need to give the entity name and also you have to provide the reference value and synonyms for it while creating the Intent as following,



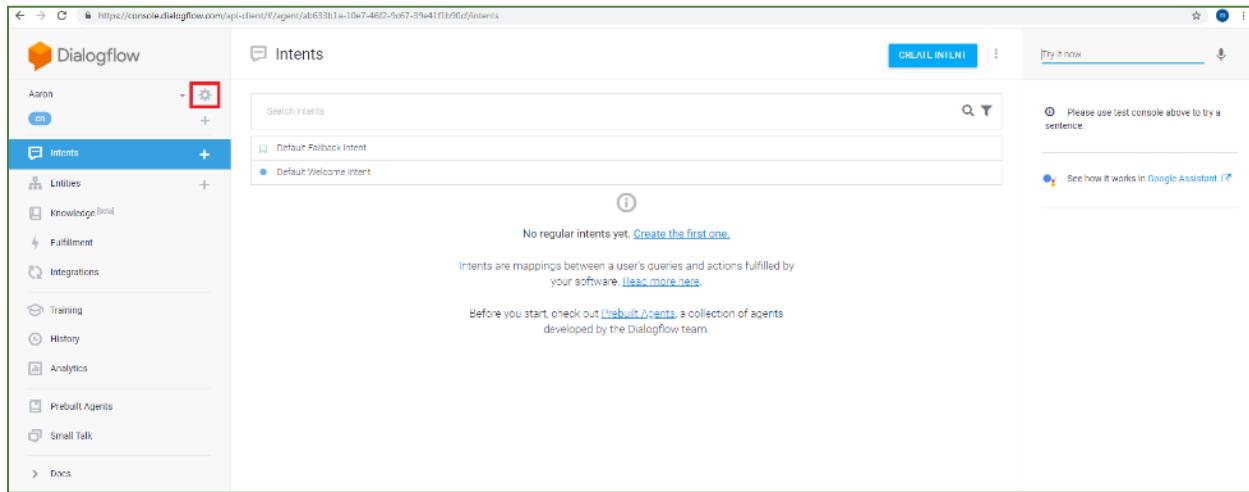
Then click on **SAVE** button on the top right to save the created entity.

You can test your Bot in the right-side test console provided by Dialogflow.

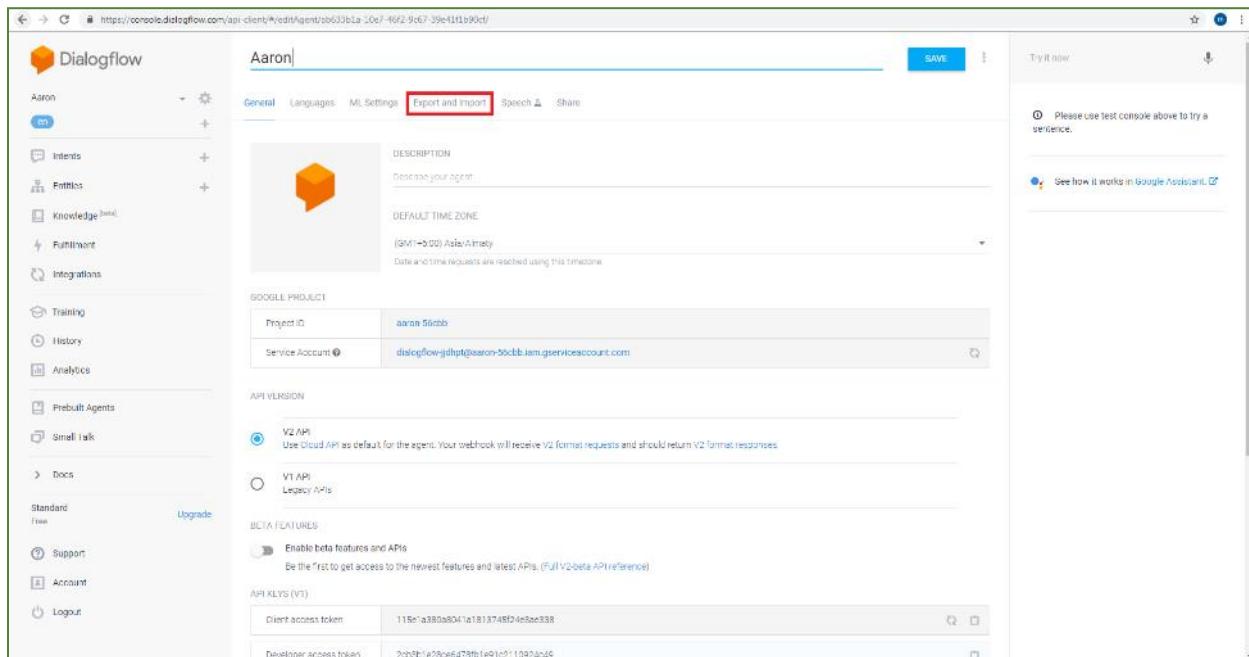


From our GitHub repository, download the zip file in your system. You can find the sample Agent which is created with the name **dialog_model.zip**

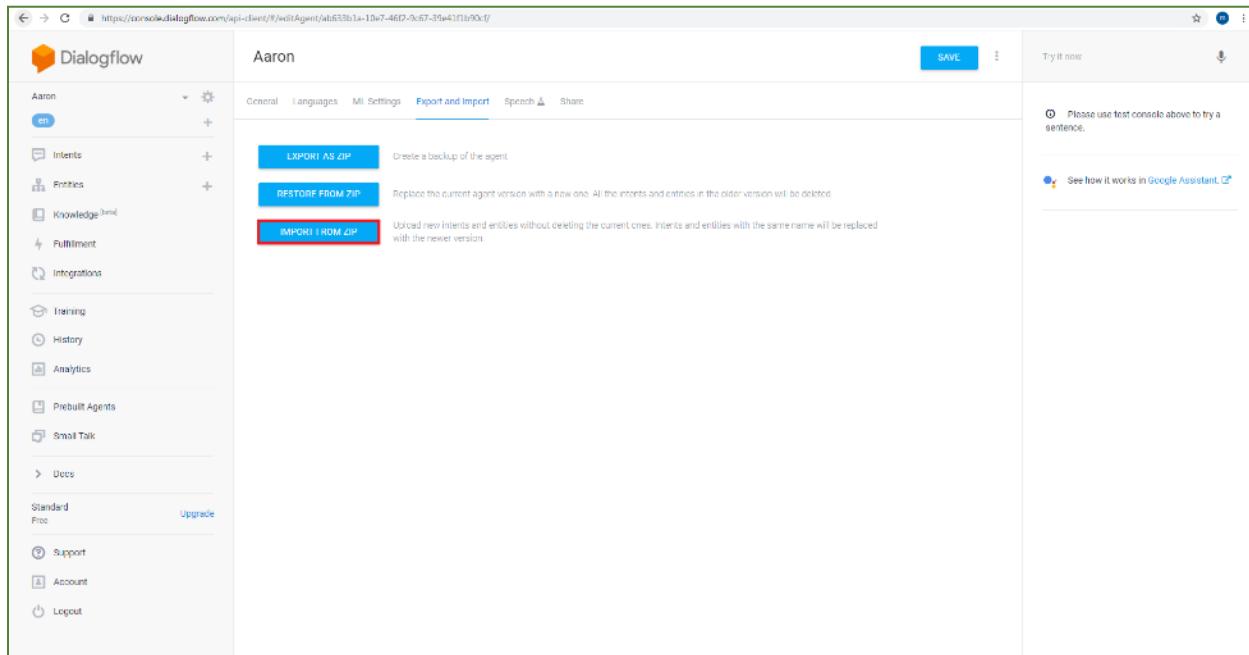
Now go to Dialogflow console and click on gear icon as shown below to import Intents and Entities.



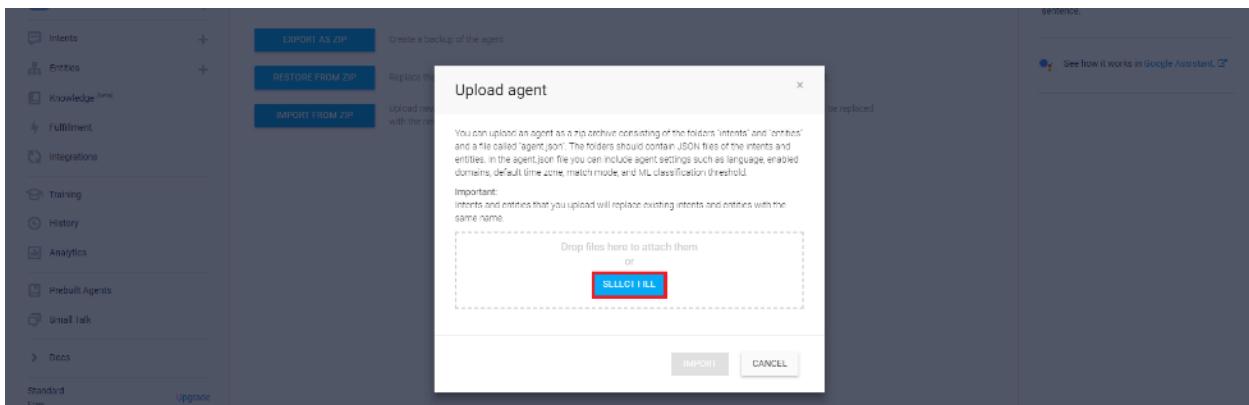
By default, you will be landed on General session page. Select **Export and Import** tab.



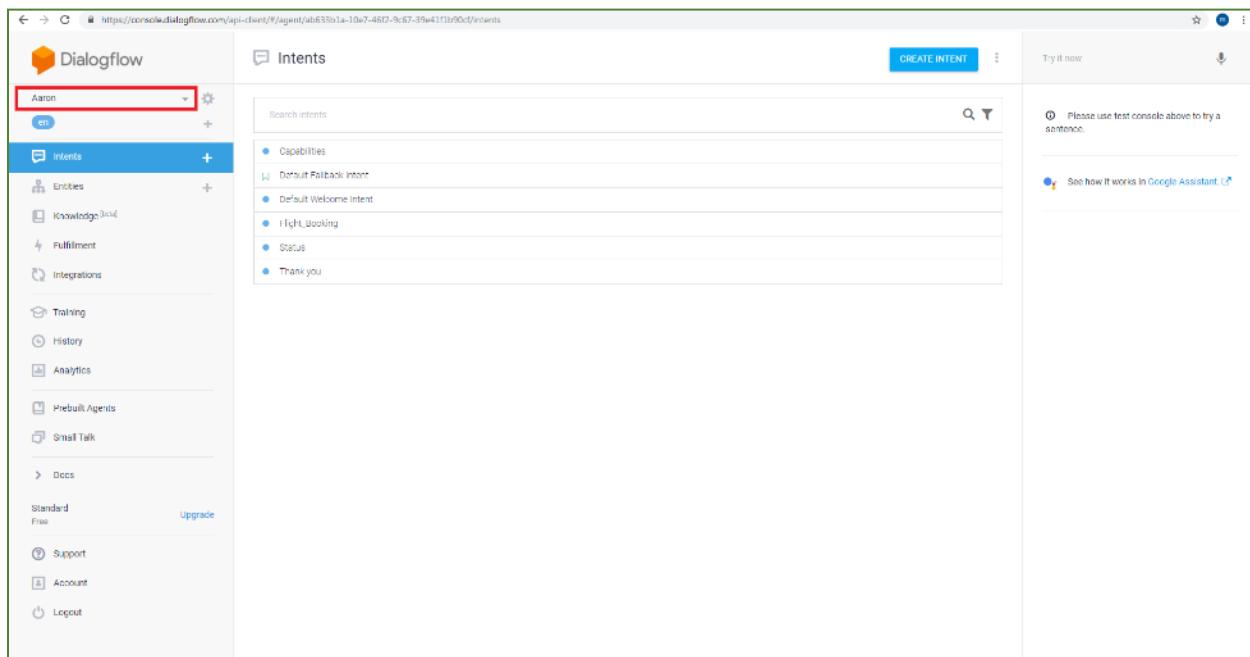
Click on **IMPORT FROM ZIP** tile, which will prompt you to select the zip file from your system.



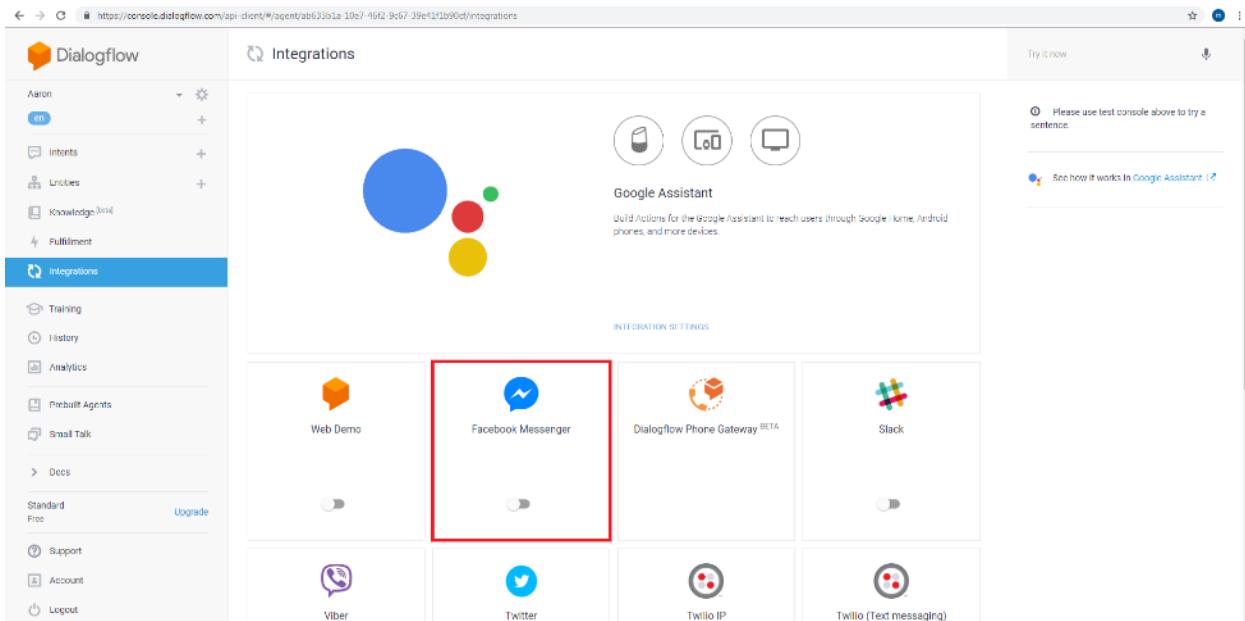
A pop-up is displayed asking you to **SELECT FILE**, you have to upload the zip file which you downloaded previously from GitHub repository. Once the upload is done click on **IMPORT** button.



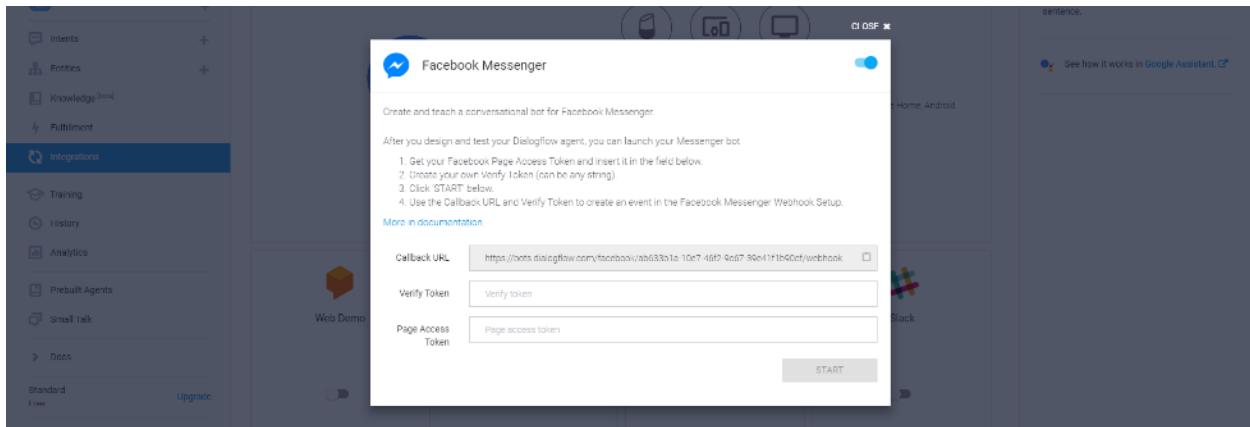
Now, you will find the agent of **Aaron Bot** which you imported.



In your agent, go to **Integrations** tab on the left side menu in Dialogflow and you will see a bunch of one click integrations. In that, go ahead and enable the Facebook Messenger integration.



Once you click on Facebook Messenger integration, you will get a default Callback URL in a pop-up. Copy that Callback URL.

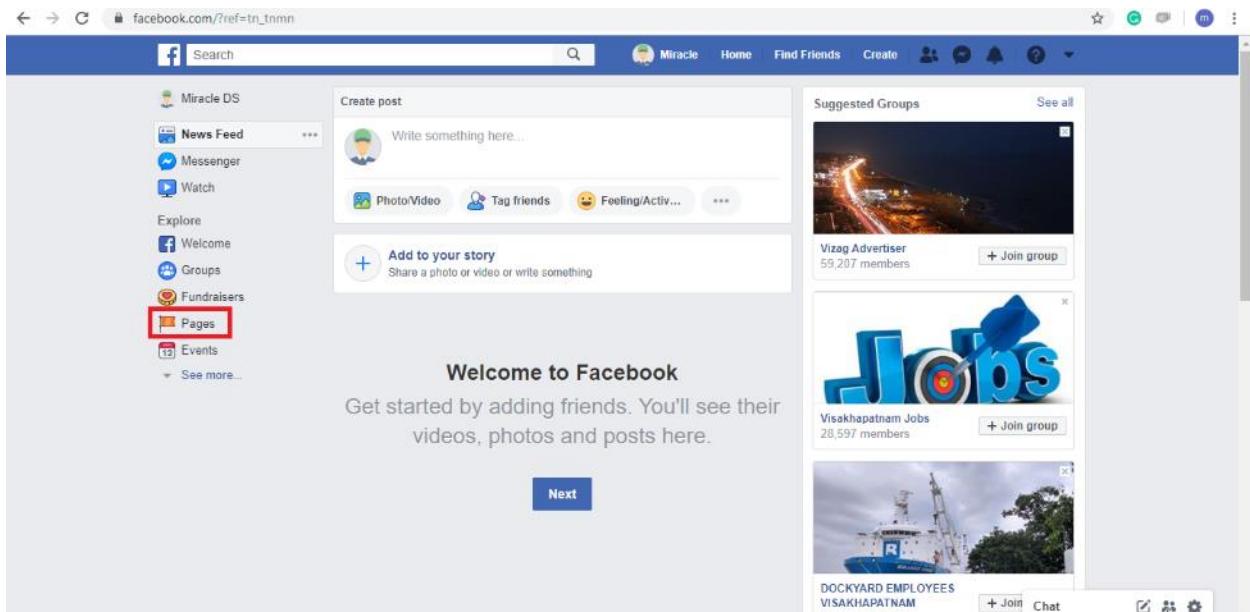


Step #4 | Create Facebook Page to Access Chatbot

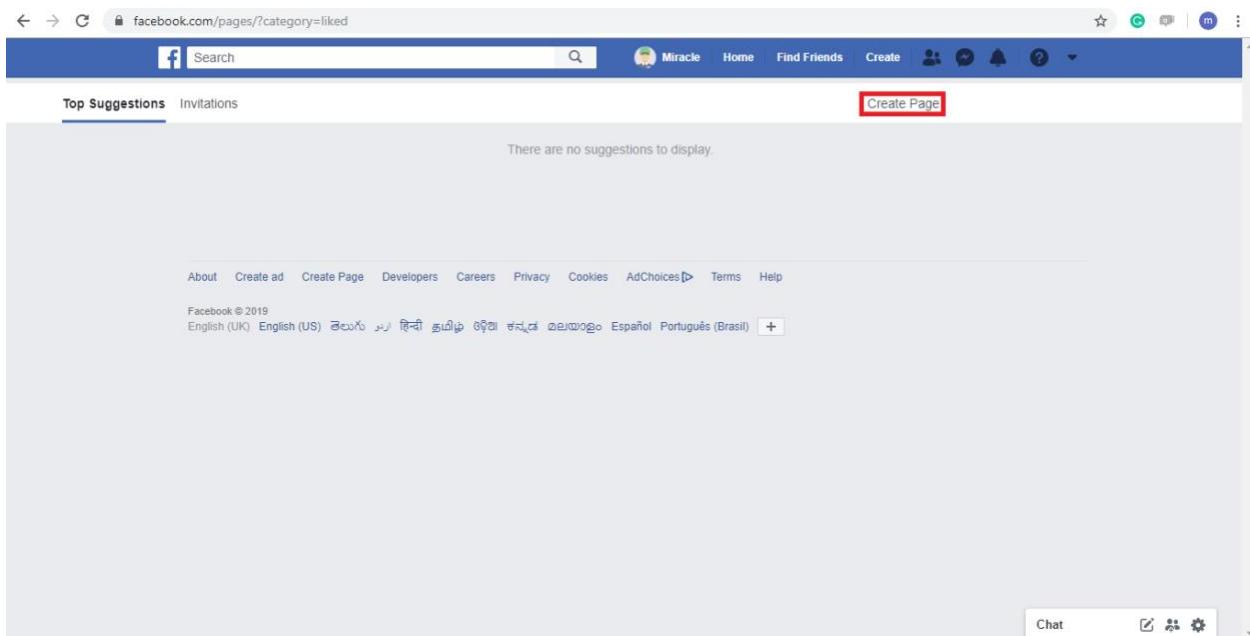
The first step will be to create a Facebook account and **Log In** by providing Email or Phone and Password at, <https://www.facebook.com/>



Once you login into Facebook, you need to create a new page by opening the below link in a new tab, <https://www.facebook.com/pages/create/> or else click on **Pages** as shown below.



If you are creating a page for very first time, you will not find any existing pages. Then click on **Create Page** button as shown below,



Now, provide a **Page name** and select **Category** of that page and click on **Continue** button. Note that page name should be unique.

The screenshot shows the 'Create a Page' wizard on the Facebook website. On the left, under the 'Business or brand' section, the page name is set to 'Aaron' and the category is 'Transport system'. A note at the bottom states that Pages, Groups, and Events Policies apply. A red 'Continue' button is visible. On the right, there's a preview of a page with a yellow flag icon and the text 'Community or public figure'. A 'Get Started' button is also present.

After clicking on Continue button, your page will be created and looks like below,

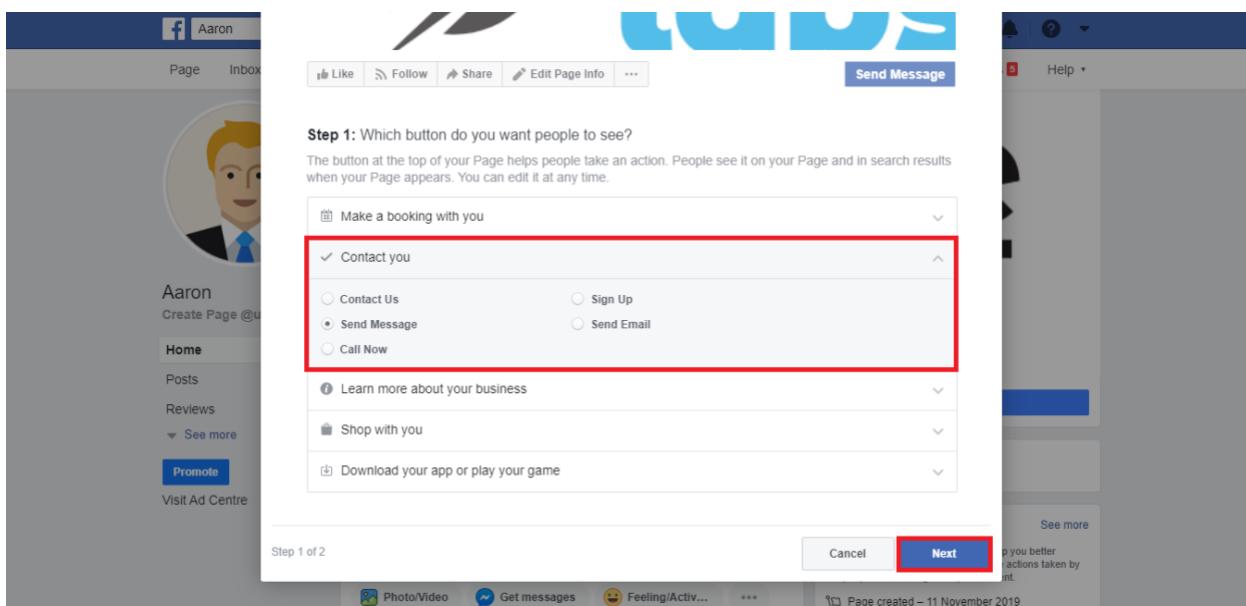
The screenshot shows the newly created Facebook page for 'miracle labs'. The page header includes the profile picture of Aaron, the page name 'Aaron', and a 'Create Page @username' link. Below the header, there are tabs for Home, Posts, Reviews, and Promote. The main content area features the large, bold page title 'miracle labs' in black and blue. Below the title are standard Facebook controls for Like, Follow, Share, Edit Page Info, and Add a Button. There are also buttons for Create, Live, Event, Offer, and a 'Write a post...' input field. A 'Page transparency' section is visible at the bottom.

Once you are done with the page creation, click on + Add a Button.

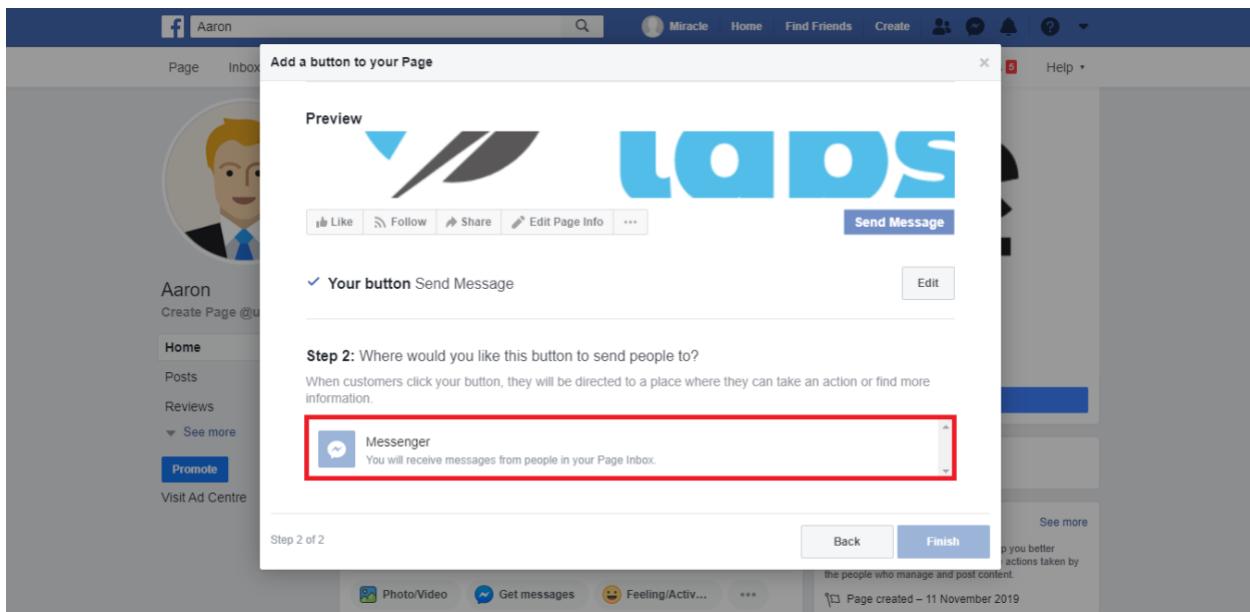


We need to get a **Send Message** button to start the conversation with bot.

To get that, click on **+ Add a Button**. You will get the options to select the button category. Please select the **Contact you** option and choose **Send Message** radio button as below and click on **Next**.

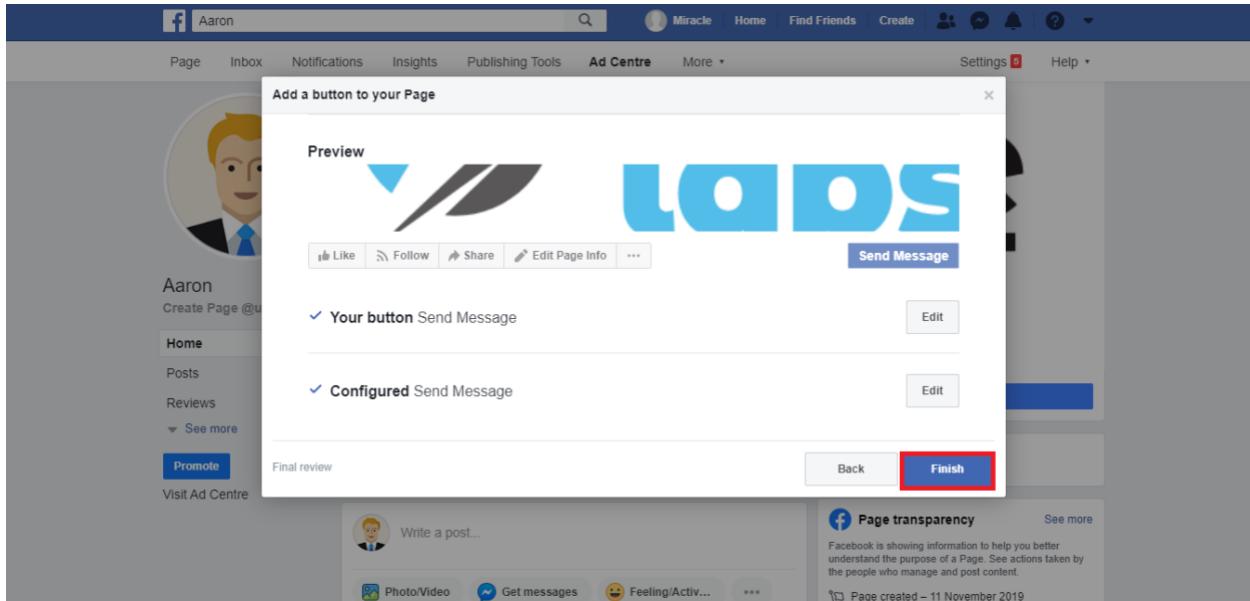


After clicking on **Next** button, it will ask for another option - **Where would you like this button to send people to?** Place your cursor over **Messenger** and click on it.



Once you complete the above steps, **Finish** button will be enabled and click on that button.

Note - If you want to edit any option that you selected previously, click on **Edit** button which is available above the **Finish** button.

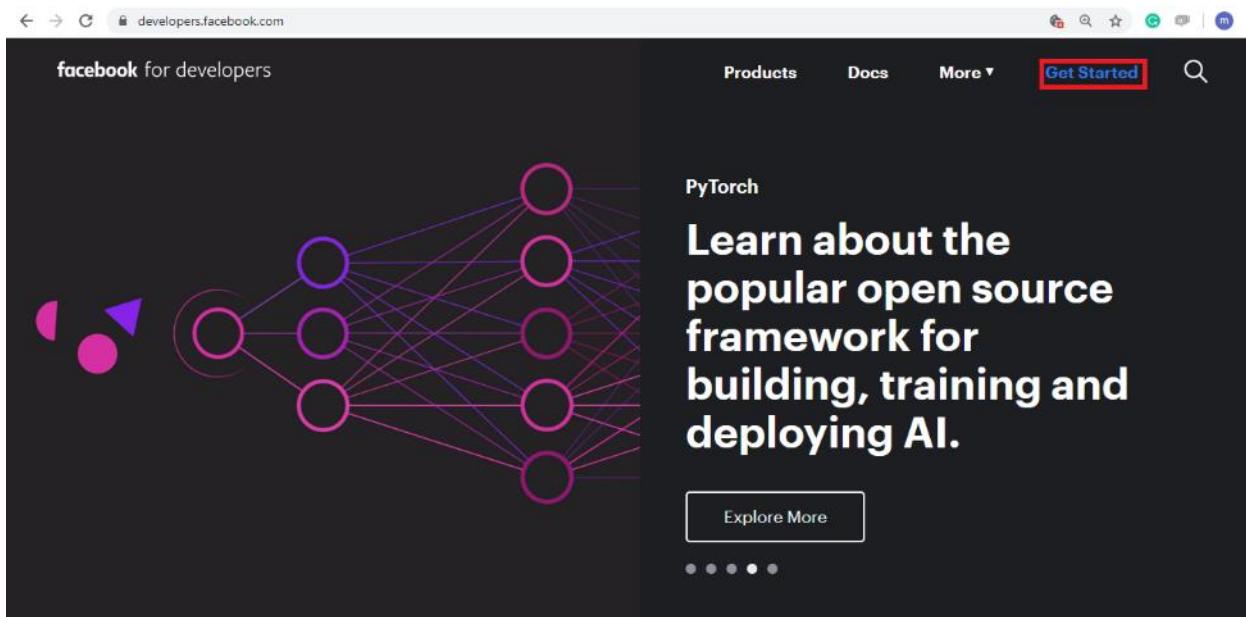


Now, you can see that the option **+ Add a Button** on your page is replaced with **Send Message**.

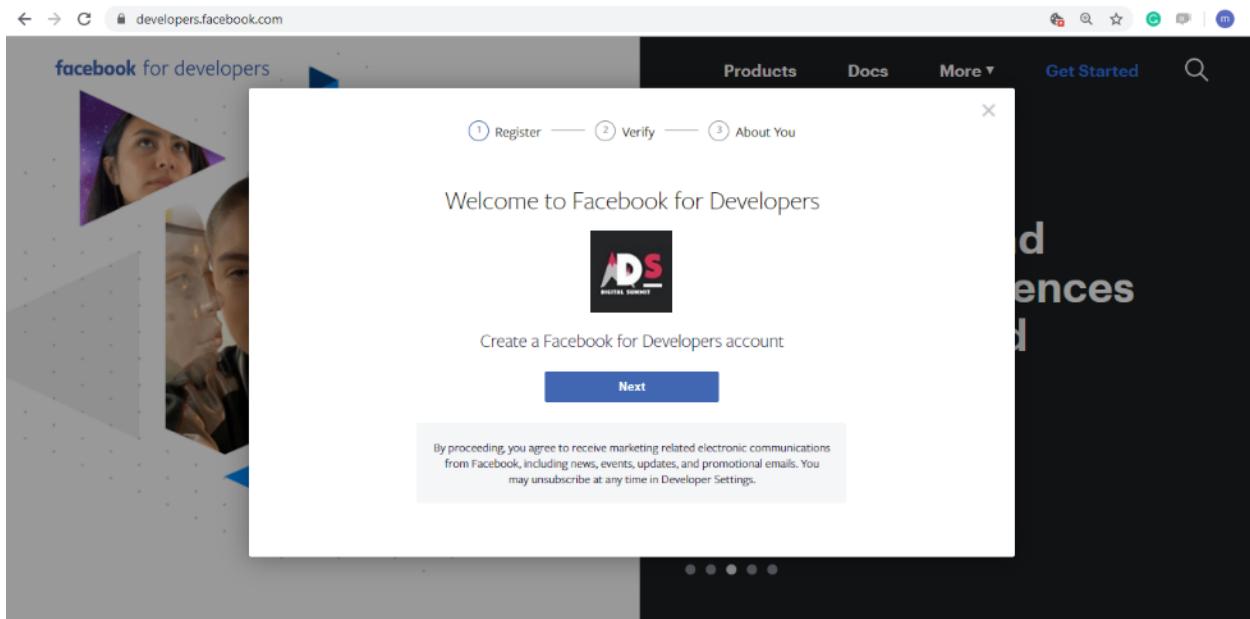


Step #5 | Accessing Facebook for Developers

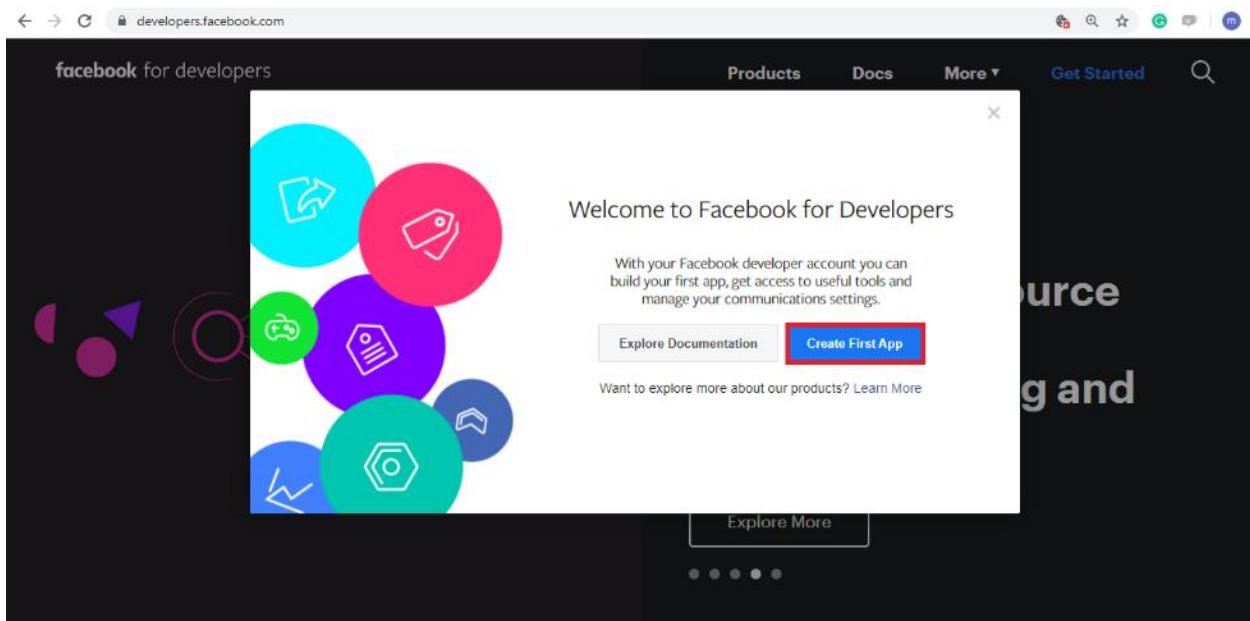
Login into Facebook for Developers <http://developers.facebook.com/> for creating apps in order to generate page access token. If you had already logged in, click on **Get Started** as shown in the image.



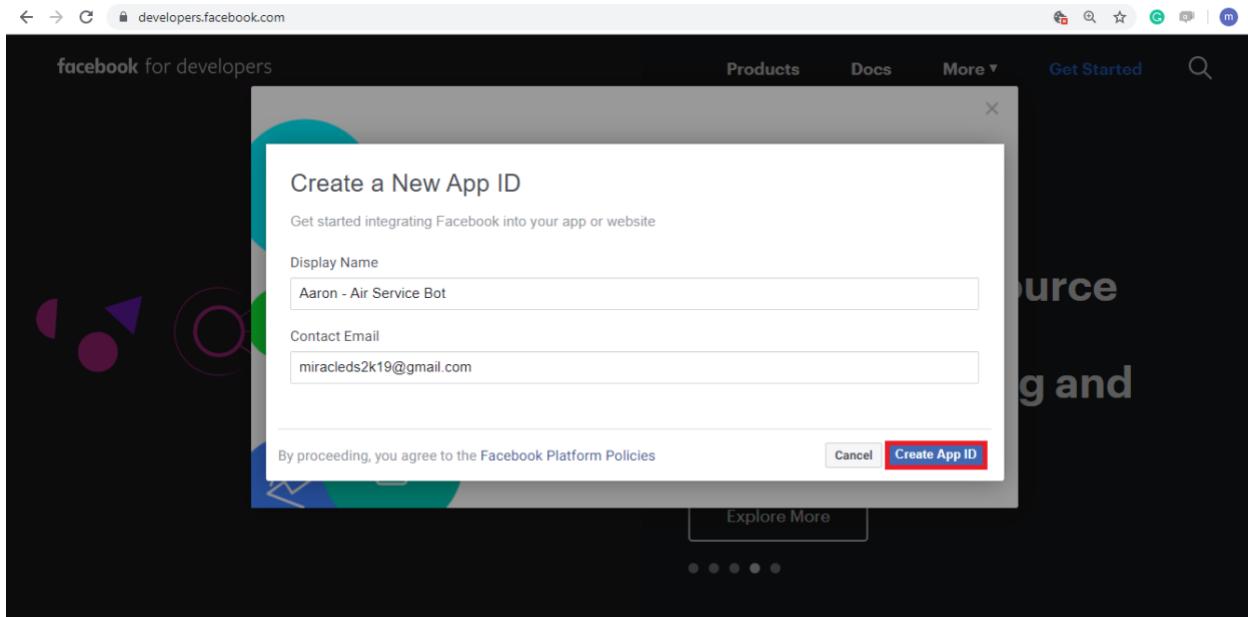
By clicking on Get Started button, it will ask for verification process to create a developers account in Facebook. Let's complete all 3 steps - **Register, Verify** and **About You**. To complete the above steps click on next for creating developers account and authenticate yourself by providing Phone number to get verified and select your role in the next step.



After completion of the above verification process, we need to create a First App in developers account as shown below.



Now it will ask you to provide the Display name and contact details for the App. Please provide all the necessary details for your respective page and click on **Create App ID** button.



Note - If you have already created the app and you want to use it, select it from the **My Apps** menu.

Product	Description	Action
Account Kit	Seamless account creation. No more passwords.	Read Docs Set Up
Facebook Login	The world's number one social login product.	Read Docs Set Up
Audience Network	Monetize your mobile app or website with native ads from 3 million Facebook advertisers.	Read Docs Set Up
Analytics	Understand how people engage with your business	Read Docs
Messenger	Customize the way you interact with people on	Read Docs
Webhooks	Subscribe to changes and receive updates in real	Read Docs

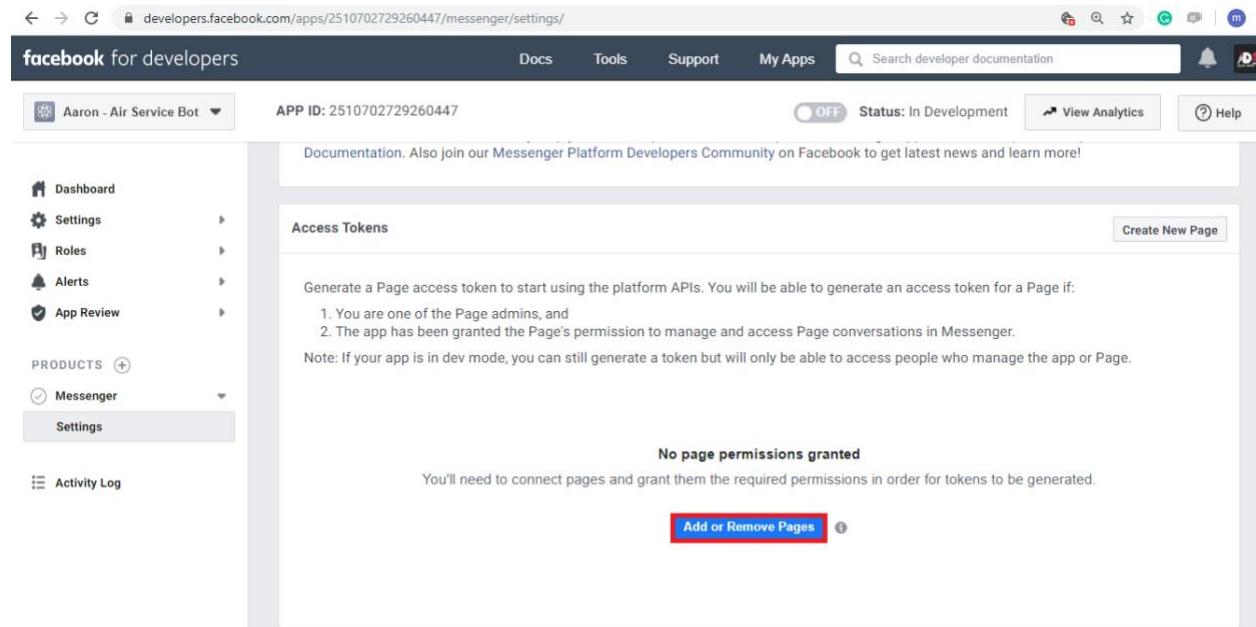
Select **Messenger** option and click on **Set Up** to customize the way, to interact with people on Messenger through your app.

The screenshot shows the Facebook for Developers dashboard for an app with ID 2510702729260447. The left sidebar includes options like Dashboard, Settings, Roles, Alerts, App Review, and a Products section with a plus sign. Under Products, 'Messenger' is listed with a red box around its 'Settings' link. The main content area displays several tiles: Account Kit, Facebook Login, Audience Network, Analytics, Messenger (with a red box around its 'Set Up' button), and Webhooks. Each tile has a 'Read Docs' button and either a 'Set Up' or 'View Analytics' button.

Once you click on the Set Up option, you will get the Messenger settings. In left side panel under the Product, go to Messenger tile and select **Settings**.

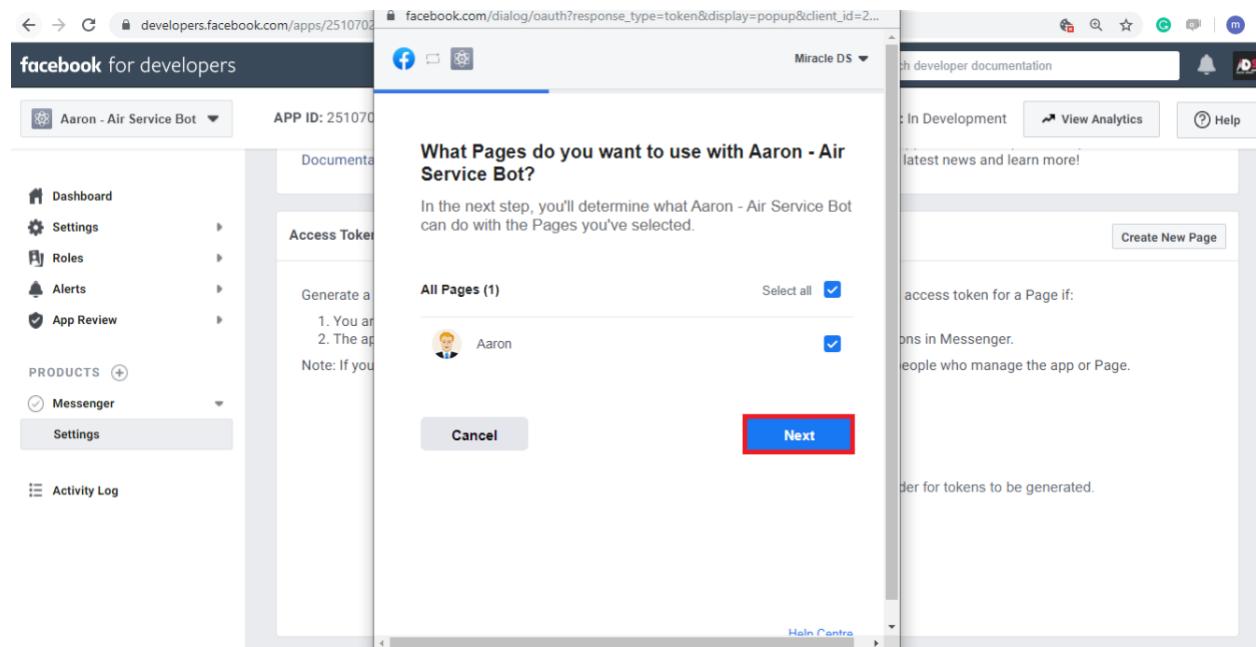
The screenshot shows the Messenger Platform settings page. The left sidebar highlights the 'Messenger' product and its 'Settings' link, which is also enclosed in a red box. The main content area features a 'Messenger Platform' section with a welcome message about the platform's APIs and webview. It also includes a 'Increase Traffic to Messenger' section with a 'Learn more' button and a note about creating ads to help discover the experience in Messenger.

Scroll down to **Access Tokens** and click on **Add or Remove Pages** to generate token for your pages.



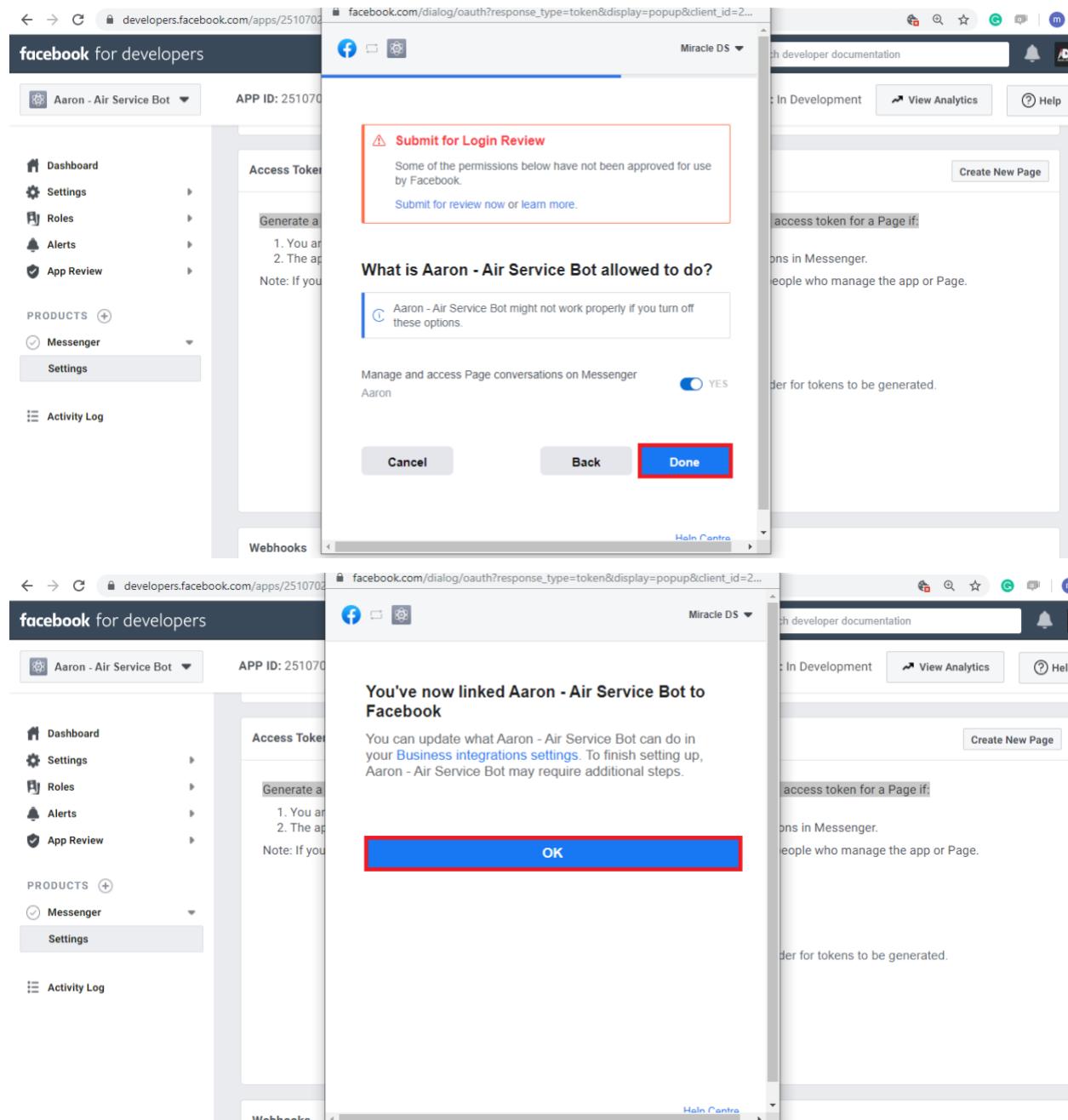
The screenshot shows the Facebook for Developers interface. On the left, there's a sidebar with options like Dashboard, Settings, Roles, Alerts, App Review, Products (Messenger), and Activity Log. The main area is titled 'Access Tokens'. It contains instructions to generate a Page access token for the platform APIs. A note says: 'If your app is in dev mode, you can still generate a token but will only be able to access people who manage the app or Page.' Below this, a section titled 'No page permissions granted' states: 'You'll need to connect pages and grant them the required permissions in order for tokens to be generated.' A prominent red button labeled 'Add or Remove Pages' is centered at the bottom of this section.

Now, you will get all the pages that you are created in a new window. Select the page that you want to submit for review and click on **Next**.



This screenshot shows a modal dialog box from Facebook's OAuth interface. The title is 'What Pages do you want to use with Aaron - Air Service Bot?'. It asks: 'In the next step, you'll determine what Aaron - Air Service Bot can do with the Pages you've selected.' Below this, it lists 'All Pages (1)' with a checkbox labeled 'Select all' which is checked. A small profile picture of a person named 'Aaron' is shown next to the list. At the bottom of the dialog are two buttons: 'Cancel' and a large red 'Next' button.

Once you click on **Next**, you need to submit your page for review by clicking on done button. Complete the review process for getting all permissions to your page as shown below.



The image consists of two vertically stacked screenshots from the Facebook for Developers platform, specifically the App Review section. Both screenshots show a browser window with a URL like `facebook.com/dialog/oauth?response_type=token&display=popup&client_id=2...`.

Screenshot 1 (Top): This screenshot shows a 'Submit for Login Review' dialog. It contains a warning message: 'Some of the permissions below have not been approved for use by Facebook.' Below this is a link 'Submit for review now or learn more.' At the bottom of the dialog are three buttons: 'Cancel', 'Back', and a large blue 'Done' button which is highlighted with a red rectangle.

Screenshot 2 (Bottom): This screenshot shows a confirmation message: 'You've now linked Aaron - Air Service Bot to Facebook'. It includes a note: 'You can update what Aaron - Air Service Bot can do in your [Business integrations settings](#). To finish setting up, Aaron - Air Service Bot may require additional steps.' At the bottom is a single blue 'OK' button, which is also highlighted with a red rectangle.

Once you are done with the above process you will get the list of pages that have been submitted for the review. Now click on the **Generate Token** button to get the access token for the respective page.

developers.facebook.com/apps/2510702729260447/messenger/settings/

facebook for developers Docs Tools Support My Apps Search developer documentation

Aaron - Air Service Bot APP ID: 2510702729260447 Status: In Development View Analytics Help

Access Tokens

Generate a Page access token to start using the platform APIs. You will be able to generate an access token for a Page if:

1. You are one of the Page admins, and
2. The app has been granted the Page's permission to manage and access Page conversations in Messenger.

Note: If your app is in dev mode, you can still generate a token but will only be able to access people who manage the app or Page.

Pages ↑	Tokens
Aaron 104978990961853	—

Webhooks

To receive messages and other events sent by Messenger users, the app should enable webhooks integration.

Add Callback URL

Now click on the checkbox '*I understand*', copy the access token and click on done.

developers.facebook.com/apps/2510702729260447/messenger/settings/

facebook for developers Docs Tools Support My Apps Search developer documentation

Aaron - Air Service Bot View Analytics Help

Token Generated

Aaron
104978990961853

To protect your security, ONLY share this token with app developers you trust.

This token will only be shown once, so keep it safe. If it gets lost, you'll need to create a new one. Anyone could potentially use this token to impersonate this page, depending on the privacy settings of your app. If you wish to revoke all previously generated tokens from a page, you can remove this page from the app using the button below the table.

I Understand

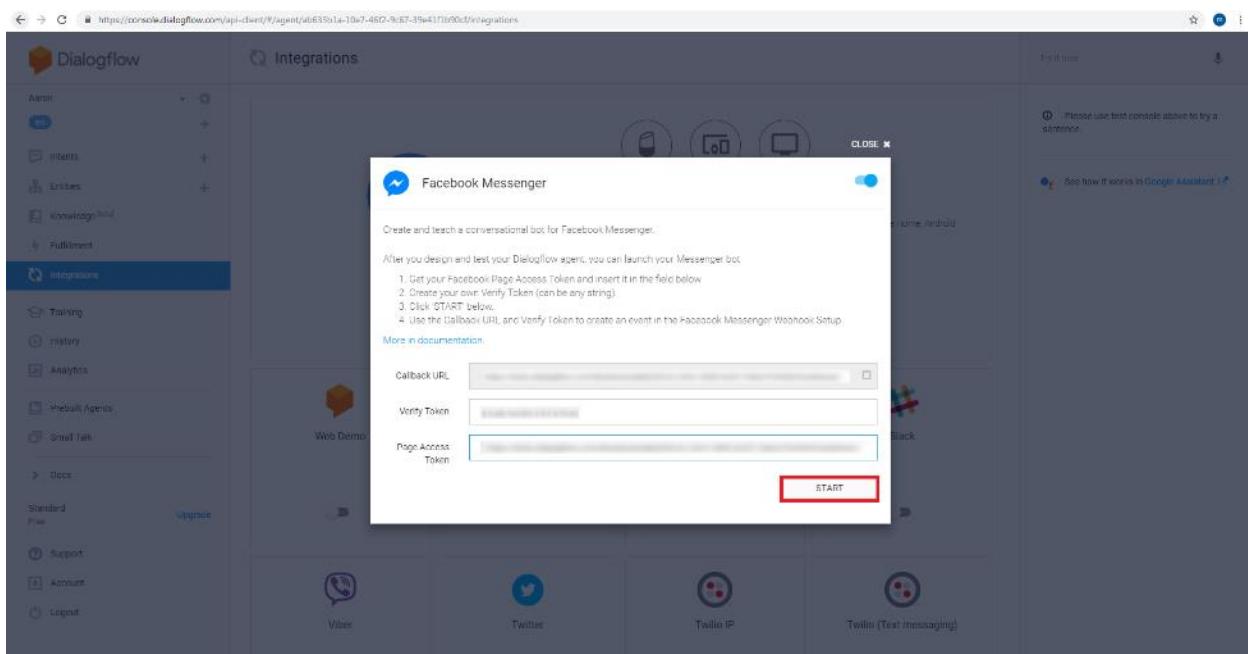
EAAjrelhQbZA8BAOB1A1t6nSuWgZBDUWk3PhvFLH3eWDezWZAQTjkGXSZBT8eEfoZALV8NW...

To receive messages and other events sent by Messenger users, the app should enable webhooks integration.

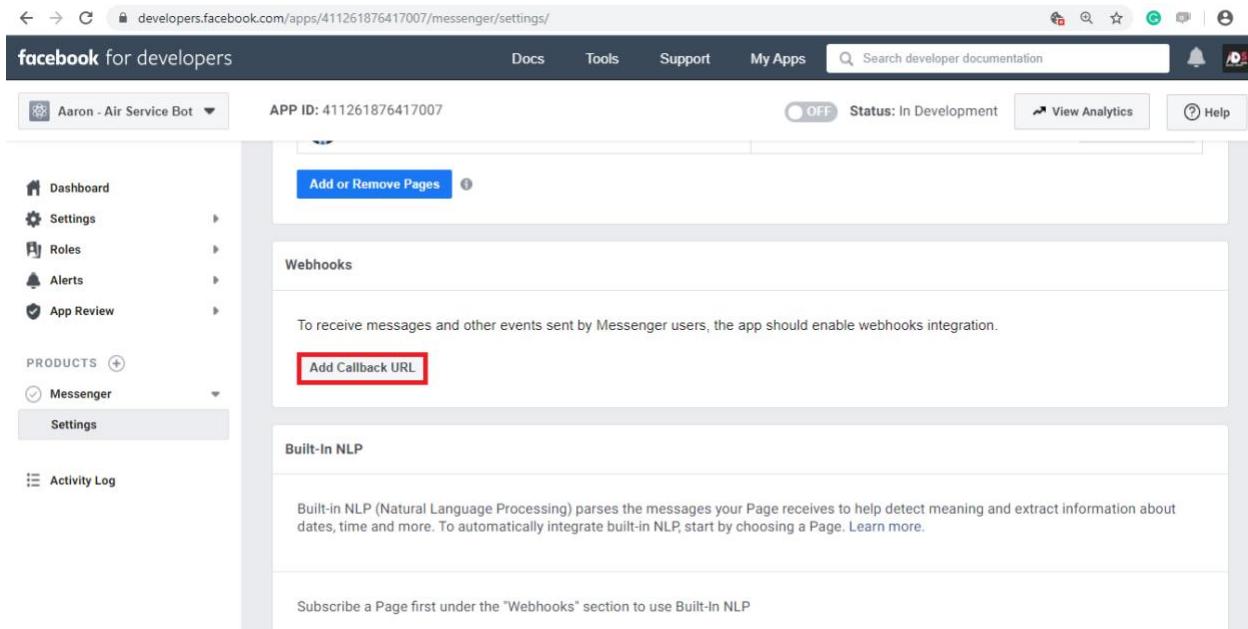
Add Callback URL

Step #6 | Integration with Facebook Messenger and Dialogflow

Navigate to Dialogflow Facebook Integrations and paste it over the **Page Access Token** section. Provide your own **Verify Token** and click on **Start** button.

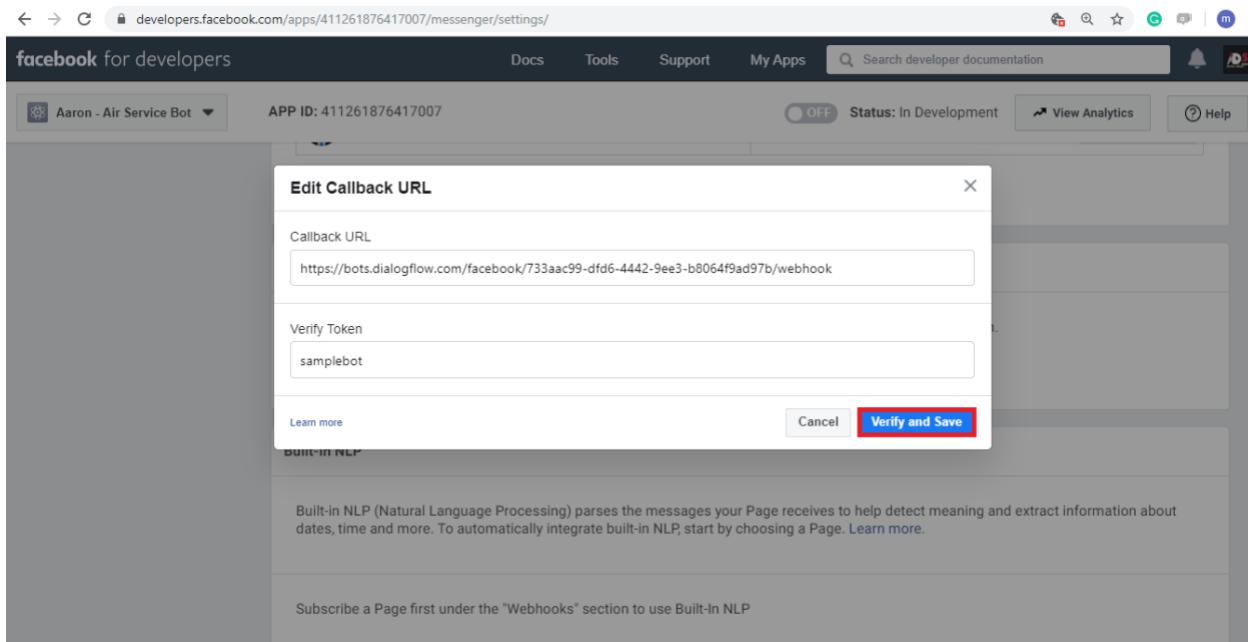


After clicking on start button in Dialogflow Facebook Integration, navigate to the app that you created in your Facebook Developer Page. Go to the Messenger settings and scroll down to the Webhooks section. Click on **Add Callback URL**.

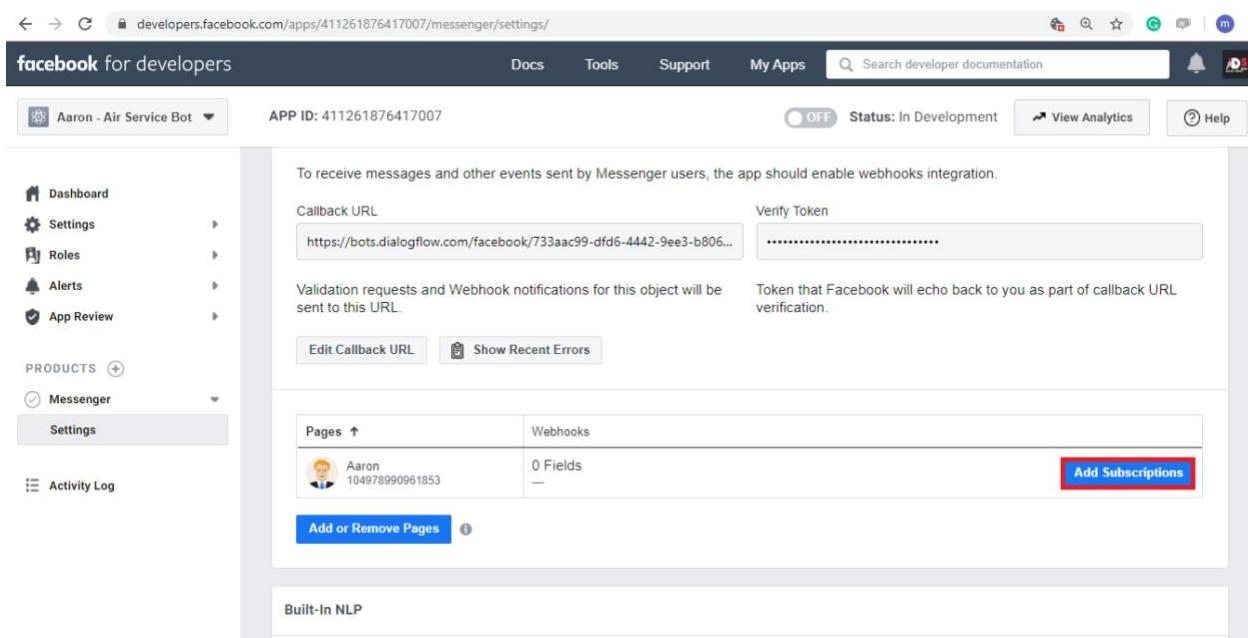


You will be navigated to **Edit Callback URL** window, paste the callback URL that was copied previously in the specified field. In the **Verify Token** field, specify the same

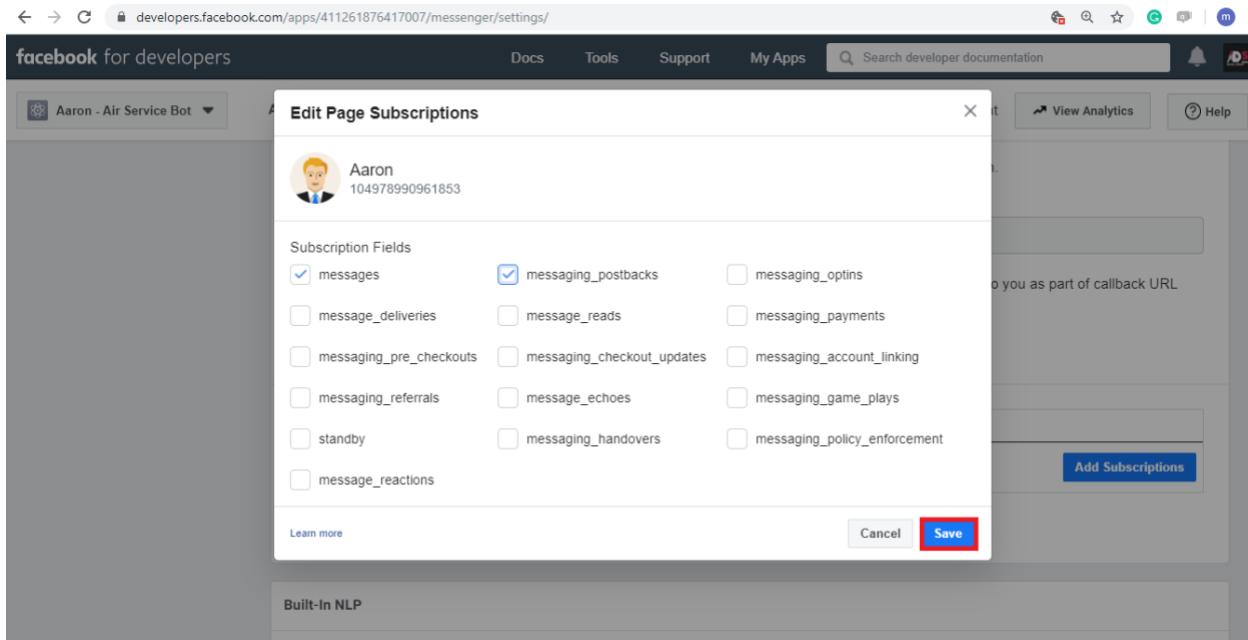
Facebook verify token that you provided in Dialogflow as below and click on **Verify and Save**.



Under the Webhook section you, will find **Add Subscription** button besides the page that you had selected during token generation, and click on **Add Subscription**.



Once you click on the **Add Subscription** then you will navigate to **Edit Page Subscriptions** Window. In this window, you need to select the subscription fields like **messages** and **messaging_postbacks**. Now, click on **Save**.



Once you are done with the Webhook and Page Subscriptions setup, you can integrate the Node JS code with the chatbot you have created.

Step #7 | Integration with Node JS and Google's Dialogflow

Once you are done with the integration of Dialogflow with Facebook Messenger, navigate to the workspace folder where the code exists, and open command prompt.

Run **node app.js** to run your application.

```
C:\Windows\System32\cmd.exe - node app.js
C:\Users\avennela\Desktop\DS'18 node app.js
running on 8000
```

Application is **running on 8000** port. Remember this for later use.

Open the browser and download ngrok by using the below link, <https://ngrok.com/download> (ngrok helps your app to get secure URL to your localhost server).

You need to install the ngrok based on the system requirement as provided below,

The screenshot shows the official ngrok download page. It features a navigation bar with links for 'ngrok', 'HOW IT WORKS', 'PRICING', 'DOWNLOAD', 'DOCS', 'LOGIN', and 'SIGN UP'. The main content is titled 'Download & setup ngrok' with the sub-instruction 'Get started with ngrok in just a few seconds.' Below this, there are four numbered steps:

- 1 Download ngrok**: A red box highlights this step. It contains instructions to download the ngrok client binary and a link to 'Download for Windows' with options for Mac OS X, Linux, Mac (32-bit), Windows (32-bit), Linux (64-bit), Linux (ARM), Linux (ARM64), FreeBSD (64-bit), and FreeBSD (32-bit).
- 2 Unzip to install**: Instructions for unzipping the ngrok archive on Linux or OSX, or double-clicking the executable on Windows.
- 3 Connect your account**: Instructions to run the command '\$./ngrok auth token <YOUR_AUTH_TOKEN>' to add an auth token to the ngrok.yml file. It also mentions connecting an account to the dashboard for longer tunnel timeouts and more.
- 4 Fire it up**: Instructions to run '\$./ngrok http 80' to start an HTTP tunnel on port 80. It also provides a link to documentation for more ideas.

After downloading the ngrok, extract the downloaded folder and run the **ngrok.exe** file. Now, give the following command and click on the Enter button.

ngrok http <Your-Application-Port>

Example : ngrok http 8000

Once you run the application successfully with the above command, you will get the URL as, ***https://<random_code>.ngrok.io***. Copy this for later use.

Note - You should not close this window until you stop running your Node JS application

A terminal window titled 'C:\Users\avennela\Desktop\Desktop\ngrok.exe - ngrok http 8000' with the instruction '(Ctrl+C to quit)' at the top right. The window displays the following output:

```

ngrok by @inconshreveable
Session Status      online
Session Expires    7 hours, 59 minutes
Update             update available (version 2.2.8, Ctrl-U to update)
Version            2.2.3
Region             United States (us)
Web Interface     http://127.0.0.1:4040
Forwarding         http://2f558d60.ngrok.io -> localhost:8000
Forwarding         https://2f558d60.ngrok.io -> localhost:8000
Connections        ttl     opn     rt1     rt5     p50     p90
                  0       0     0.00   0.00   0.00   0.00

```

The URL **https://2f558d60.ngrok.io** is highlighted with a red box.

Now, navigate to the Dialogflow Fulfillment section and enable the button which is showing as **DISABLED** initially as shown below,

The screenshot shows the Dialogflow Fulfillment interface. On the left, there's a sidebar with navigation links like 'Integrations', 'Training', 'History', 'Analytics', 'Prebuilt Agents', 'Small Talk', 'Docs', 'Standard Pro', 'Support', 'Account', and 'Logout'. The main area has tabs for 'Fulfillment' (which is selected and highlighted in blue), 'Webhook', and 'Inline Editor'. The 'Webhook' tab contains instructions about receiving POST requests from Dialogflow and a note that the webhook requirements apply to the API version enabled in the agent. It features a 'Disabled' switch (which is currently off) and a 'Try it now' button. Below these are sections for 'Basic Auth' (with fields for 'Enter username' and 'Enter password') and 'Headers' (with a 'Add header' button). The 'Inline Editor' section is titled '(Powered by Cloud Functions for Firebase)' and contains a code editor with Node.js code. The code handles intents for 'GREETING', 'NAME', and 'WELCOME'. It uses Cloud Functions for Firebase to interact with Google Assistant and performs database operations.

```

1 // This function is called when Dialogflow receives a POST request to your endpoint
2 // It takes the intent and sends a response back to Dialogflow
3 // See https://cloud.google.com/functions/docs/writing/http#nodejs
4
5 const functions = require('firebase-functions');
6 const dialogflow = require('dialogflow');
7 const db = require('firebase-admin').database();
8
9 process.env.NODE_ENV = 'staging';
10
11 exports.dialogflowFulfillment = functions.https.onRequest((request, response) => {
12   const agent = new dialogflow.Agent();
13   const dialogflowResponse = agent.detectIntent(request);
14   const dbRef = db.ref(`users/${request.queryResult.context.params.user_id}`);
15
16   dialogflowResponse.promise.then(response => {
17     dbRef.set(response.queryResult.response);
18   });
19
20   function welcome(agent) {
21     agent.add(`Welcome to my agent!`);
22   }
23
24   function greeting(agent) {
25     agent.add(`Hello! How can I help you today?`);
26   }
27
28   function name(agent) {
29     agent.add(`What is your name?`);
30     agent.add(`Please tell me your name.`);
31     agent.add(`I would like to know your name.`);
32     agent.add(`Please tell me your first name.`);
33     agent.add(`I would like to know your first name.`);
34     agent.add(`Please tell me your last name.`);
35     agent.add(`I would like to know your last name.`);
36     agent.add(`Please tell me your full name.`);
37     agent.add(`I would like to know your full name.`);
38   }
39
40   function unknown(agent) {
41     agent.add(`I'm sorry, I don't understand what you said.`);
42   }
43
44   function endConversation(agent) {
45     agent.add(`Goodbye!`);
46   }
47
48   function fallback(agent) {
49     agent.add(`I'm sorry, I don't understand what you said.`);
50   }
51
52   function error(agent, err) {
53     console.error(`Error: ${err}`);
54   }
55
56   const intentMap = new dialogflow.IntentMap();
57   intentMap.set('GREETING', greeting);
58   intentMap.set('NAME', name);
59   intentMap.set('WELCOME', welcome);
60   intentMap.set('UNKNOWN', unknown);
61   intentMap.set('FALLBACK', fallback);
62
63   dialogflowResponse.promise.then(response => {
64     const result = response.queryResult;
65     const message = result.fulfillmentText || result.speech;
66     const user_id = result.context.params.user_id;
67
68     dbRef.set(message);
69   });
70
71   response.json({
72     fulfillmentText: dialogflowResponse.promise.value().queryResult.fulfillmentText
73   });
74 });

```

Once the Fulfilment section is enabled, you will need to provide the ngrok URL and it acts as a webhook which is an endpoint to your Node JS application.

Now, click on **Save** button.

This screenshot shows the same Dialogflow interface as the previous one, but with the 'Enabled' switch turned on (indicated by a blue circle). The 'URL' field is populated with 'https://255050.ngrok.io/test'. The rest of the interface remains the same, including the 'Basic Auth' and 'Headers' sections, the 'Inline Editor' with its Node.js code, and the 'Try it now' button.

Now, click on the **SAVE** button.

```

// This file contains Cloud Functions for Firebase fulfillment code.
// To learn more, see https://firebase.google.com/docs/functions/fulfillment

// Import the Dialogflow module
const functions = require('firebase-functions');
const dialogflow = require('actions-on-google').Dialogflow;
const agent = new dialogflow.Agent();
const sessionClient = dialogflow.createSessionClient();
const intentClient = dialogflow.createIntentClient();
const response = require('express').Response;
const request = require('express').Request;
const util = require('util');

// Function to handle messages
exports.handleMessage = (request, response) => {
    const session = sessionClient.session(request.session);
    const intent = intentClient.intent(session);
    const parameters = intent.parameters;
    const fulfillmentText = util.format(`You said: ${parameters.text}`);
    const messageText = `I heard you say: ${parameters.text}`;
    const message = agent.addText(messageText);
    message.ask(fulfillmentText);
    sessionClient.close();
};

// Function to handle session openings
exports.handleSessionOpen = (request, response) => {
    const session = sessionClient.session(request.session);
    const intent = intentClient.intent(session);
    const parameters = intent.parameters;
    const fulfillmentText = util.format(`Welcome to ${parameters.name}!`);
    const messageText = `Welcome to ${parameters.name}!`;
    const message = agent.addText(messageText);
    message.ask(fulfillmentText);
    sessionClient.close();
};

// Function to handle slot filling
exports.handleSlotFilling = (request, response) => {
    const session = sessionClient.session(request.session);
    const intent = intentClient.intent(session);
    const parameters = intent.parameters;
    const fulfillmentText = util.format(`You said: ${parameters.text}`);
    const messageText = `I heard you say: ${parameters.text}`;
    const message = agent.addText(messageText);
    message.ask(fulfillmentText);
    sessionClient.close();
};

// Function to handle end of conversation
exports.setEndOfConversation = (request, response) => {
    const session = sessionClient.session(request.session);
    const intent = intentClient.intent(session);
    const parameters = intent.parameters;
    const fulfillmentText = util.format(`Thank you for your visit to ${parameters.name}. You can return soon!`);
    const messageText = `Thank you for your visit to ${parameters.name}. You can return soon!`;
    const message = agent.addText(messageText);
    message.end();
    sessionClient.close();
};

// Function to handle webhook calls
exports.webhookCall = (request, response) => {
    const session = sessionClient.session(request.session);
    const intent = intentClient.intent(session);
    const parameters = intent.parameters;
    const fulfillmentText = util.format(`You said: ${parameters.text}`);
    const messageText = `I heard you say: ${parameters.text}`;
    const message = agent.addText(messageText);
    message.ask(fulfillmentText);
    sessionClient.close();
};

// Function to handle webhook calls for slot filling
exports.webhookCallForSlotting = (request, response) => {
    const session = sessionClient.session(request.session);
    const intent = intentClient.intent(session);
    const parameters = intent.parameters;
    const fulfillmentText = util.format(`You said: ${parameters.text}`);
    const messageText = `I heard you say: ${parameters.text}`;
    const message = agent.addText(messageText);
    message.ask(fulfillmentText);
    sessionClient.close();
};

```

To get responses from your Node JS application to Dialogflow enable the fulfillment in all the required intents as shown below,

Parameter	Type	Value	Description
fromCity	Location	Los Angeles	From City
toCity	Location	Los Angeles	To City
date	Date	09/09/09	Date
Enter name	Text		Enter entity
	Text		Enter value

Responses

Fulfillment

- Enable webhook call for this intent
- Enable webhook call for slot filling

Step #8 | Testing Aaron Chat bot in Facebook Messenger

If you want to test the bot that you created, place your cursor over the **Send Message** button.



To start the conversation with chatbot that you created, click on **Test button** from the options.



Now, a chat window will be popped up to start the conversation with your bot. For starting the conversation with the chatbot click on **Get Started**.





Hurrah! With this lab you were able to create your first chat bot with Google's Dialogflow using Facebook Messenger.

For any questions regarding the lab please feel free to reach out to innovation@miraclesoft.com. We hope you enjoyed creating bots with us!