**Build a Restaurant bot**

[Template Bot Builder](http://blog.gupshup.io/index.php/2016/08/31/introducing-the-sme-bot-builder/) lets you create a bot in a jiffy on Facebook messenger. It has pre-defined templates designed specific to a business needs. For example - A restaurant template, which lets you create a bot for your restaurant. You can edit the elements in the template according to your need.

There is no coding involved as you can just edit the template and your bot will be ready. These templates are also pre-approved by Facebook and hence you need not send an approval request to Facebook. Once saved and published your bot will be live to all the users.

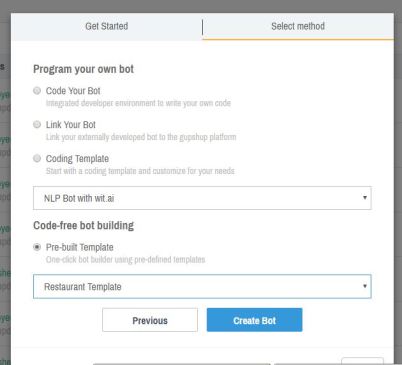
Let's now understand how to use the Template Bot Builder tool.

Pre-requisite of using this tool -

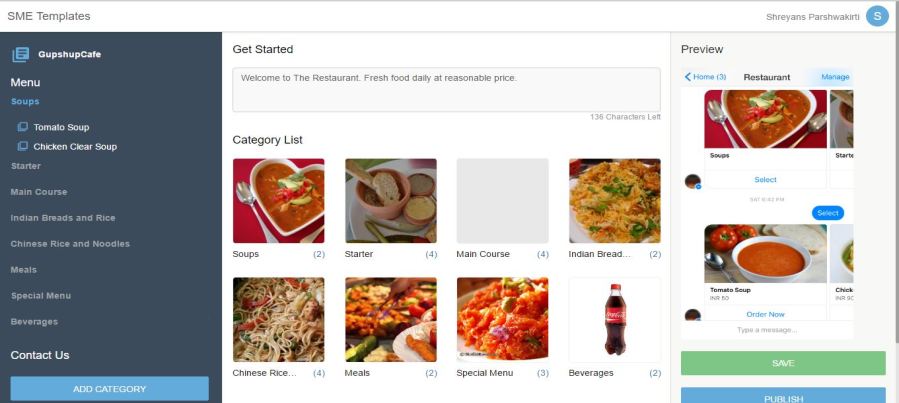
1. Facebook account.
2. Facebook page.

Steps to use the tool for bot creation -

1. **Create the bot and select the template** - Go to “[My Bots](https://www.gupshup.io/developer/dashboard)” section and click on the “+” sign. Give a name to the bot,then on the next step check the “Pre-built Template” radio button and select the desired template from the drop-down.



2. **Edit the template** - After you have selected the template, you will be navigated to a page where you can edit the templates as per your need.



Things you can edit in the template -

1. ***Menu*** - You can add/edit/delete the main menu.   
To add a new menu click on the “ADD CATEGORY” button.Adding a new menu needs you to add a picture,a name and a description to it.   
To delete a menu just hover over that menu and you will find the delete button or else you can go inside the menu and then delete it.

2. ***Menu items*** - You can choose to add/edit/delete the items in the main menu as well. Navigate inside the main menu and then click on the “+” sign to add an item to that menu.

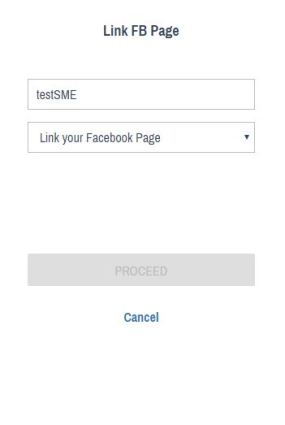
You can have maximum of 10 menus and each menu can have maximum of 10 menus items.

3. ***Contact form*** - Click on the “Contact Us” link to edit your contacts. You can provide your address, phone number, email address and website link(if any). Make sure to provide a valid email address and phone number so that users can get in touch with you.

An email will be triggered to the provided email address when user tries to order food and provide his number. This email will go into your spam folder.

4. ***Get started message*** - You can even edit the getting started message which is sent to the user for the first time he starts interacting with your bot.

3. **Save and Publish** - Once you are done with editing the template click on “Save” button on the right to save the template. Click on the “Publish” button to publish the template on your Facebook page. To publish your bot you will need to link the Facebook page to which you want the bot to be linked.



If you are not logged into Gupshup using Facebook login then you will be asked to login into Facebook at this step.

Other features -

1.**Unpublish** - You can also unpublish the template from the page in case you want to remove the bot from your page.

As of now we have Restaurant template and we will be adding more templates to the Template Bot Builder soon.

# Building a bot outside Bot Builder tool.

Gupshup allows you to write the bot code outside Gupshup's Bot Builder tool in any programming language of your choice.You can then associate a callback to the bot on Gupshup's platform.This callback will be the location of your bot code on the server where you have hosted the code. Whenever the bot is called by the user, Gupshup will send the relevant details to your bot through the callback URL using which you can respond back to the user.

Let's see how this can be achieved through an example using Java as the programming language.

Follow these steps below to build, deploy and test a weatherbot that responds to user's query about the weather in any city :

1.  **Create a bot on Gupshup platform** - Go to [My Bots](https://www.gupshup.io/developer/dashboard) section and then click on the '+' sign on the bottom right and give a name to the bot.

2.  **Write the bot's code in your IDE** -   
When a user interacts with the bot a GET call is made to the callback URL with additional parameters.The below table illustrates all the parameters which are sent to the bot.

|  |  |  |  |
| --- | --- | --- | --- |
| **Object Name** | **Sample Value** | **Description** | **Remarks** |
| contextobj | {"botname":"demobot", "channeltype":"twitter", "contextid":"john", "contexttype":"p2p"} | JSON object defining various parameter and way of conversation. | **channeltype** - the messaging channel name from where the user interacts with the bot. For list of channels refer [appendix](https://www.gupshup.io/developer/docs/bot-platform/guide/appendix).  **contextid / channelid** - The user ID of a user on the particular messaging channel. **contexttype** - The type of conversation. For the list of probable values refer [appendix](https://www.gupshup.io/developer/docs/bot-platform/guide/appendix). **display** - The display name of a user corresponding to the user ID on the messaging channel **botname** - The bot name. **text** - The message from the user. **type** - The type of message sent by the user. For the list of probable values refer [appendix](https://www.gupshup.io/developer/docs/bot-platform/guide/appendix). |
| senderobj | {"channelid":"john", "channeltype":"twitter", "display":"John Doe"} | JSON object identifying the user details. This is useful in case of a group on a messaging channel. |
| messageobj | {"text":"Hi", "type":"msg"} | JSON object for messages from the user. |
| channel | twitter | The channel from where the user is interacting with the bot. |  |
| botname | demobot | The name of the bot. |  |

Now handle these in your code to define your bot logic. For Java let's create a Java Servlet.

**Pseudo code -**

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

String contextobj = request.getParameter("contextobj");

String senderobj = request.getParameter("senderobj");

String messageobj = request.getParameter("messageobj");

}

When you map your bot to the Proxy Bot for testing, a “botmappedevent” event is triggered. You can read more about Proxy Bots [here](https://www.gupshup.io/developer/docs/bot-platform/guide/testing-your-bot). So you should handle this event to send a text to the user asking him/her to enter the city or you can also send this text if someone says 'Hi' to your bot.

**Pseudo code -**

JSONObject messageobjObj = JSONObject(messageobj);

String conversationType = messageobjObj.getString("type");

String message = messageobjObj.getString("text");

if((conversationType.equalsIgnoreCase("event")&& message.equalsIgnoreCase("botmappedevent"))|| (message.equalsIgnoreCase("hi")) )

{

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.print("Enter the city name");

out.flush();

out.close();

}

Once user provides the city name, call an external API to get the weather details

**Pseudo code -**

JSONObject messageobjObj = JSONObject(messageobj)

String conversationType = messageobjObj.getString("type");

String cityName = messageobjObj.getString("text");

if(conversationType.equalsIgnoreCase("msg"))

{

String URL = "https://api.worldweatheronline.com/free/v2/weather.ashx?q="+cityName+"&format=json&num\_of\_days=1&key={Your apikey}";

okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();

okhttp3.Request request = new okhttp3.Request.Builder()

.url(URL)

.get()

.build();

okhttp3.Response response = client.newCall(request).execute();

}

In the example we have used the API of World Weather Online but you can choose any other API available. The apikey for this API is provide by World Weather Online after registration.

Once you have the response from the external API you can send the message to the user. There are two ways to do so -

  a.  **Immediate response** - In this you return a HTTP response back to the callback without any delay.

**Pseudo code -**

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.print("Weather in"+cityName+" is -"+responseofWeatherAPI);

out.flush();

out.close();

  b.  **Delayed response** - In this you can use the Gupshup send message API. For details please read this [guide](https://www.gupshup.io/developer/docs/bot-platform/guide/sending-multiple-messages).

**Pseudo code -**

String botname = "weatherbot";

String botmessage = "Weather in"+cityName+" is -"+responseofWeatherAPI";

String URL = "http://api.gupshup.io/sm/api/bot/weatherbot/msg";

String body = "apikey={Your Gupshup API key}&botname="+botname+"&context="+contextobj+"&message="+botmessage;

okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();

okhttp3.MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");

RequestBody body = RequestBody.create(mediaType,body);

okhttp3.Request request = new okhttp3.Request.Builder()

.url(URL)

.post(body)

.build();

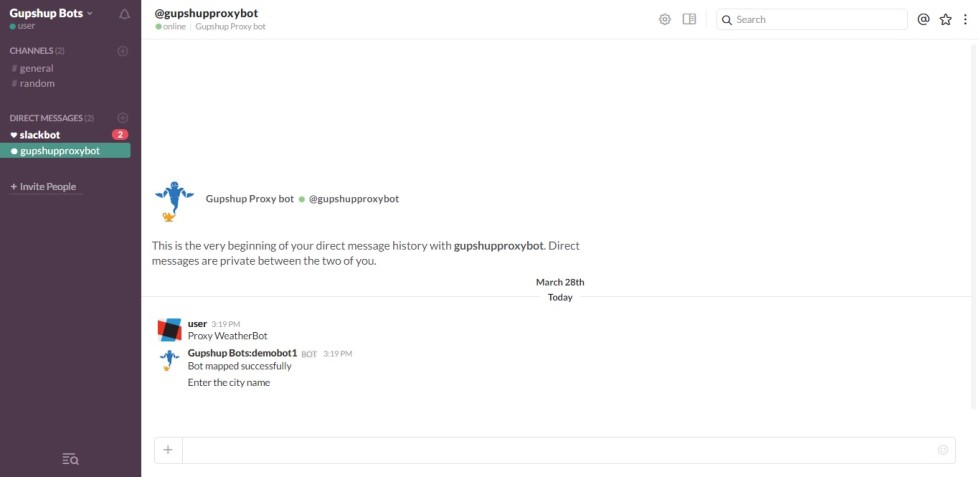
okhttp3.Response response = client.newCall(request).execute();

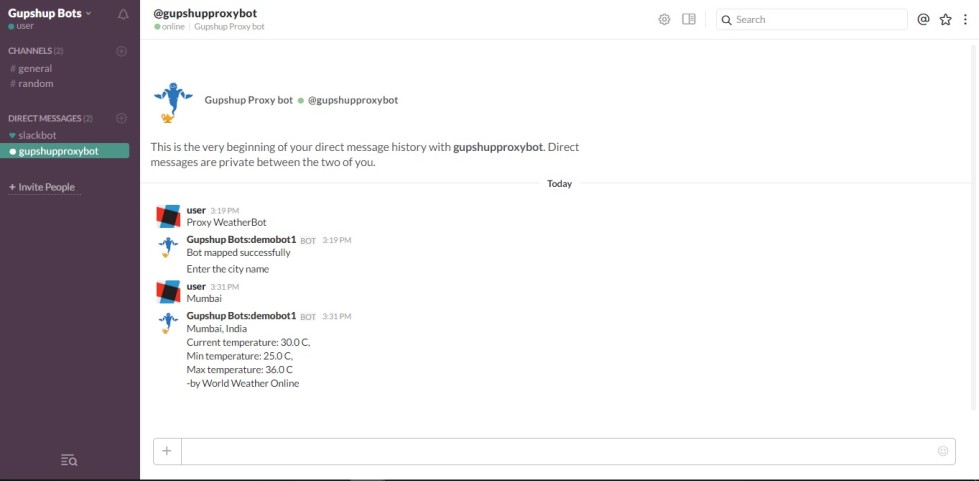
Once your code is ready, run it on the local server.

3.  **Deploy locally and associate the callback** - To deploy the bot locally and associate the callback please read this [guide](https://www.gupshup.io/developer/docs/bot-platform/guide/deploying-your-bot-locally)

4.  **Testing your bot** - You can test your bot either using the [Proxy Bots](https://www.gupshup.io/developer/docs/bot-platform/guide/testing-your-bot) or you can [publish](https://www.gupshup.io/developer/docs/bot-platform/guide/publish-bot-to-channels) your bot to the specific messaging channel and test.

Testing flow for the bot which we have built above -

    1.  Mapping the bot to Proxy Bot and getting the help text for the mapping event.   


    2.  Entering the city name and getting the result.   


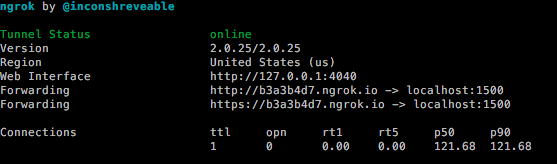
[Here is a sample code](https://github.com/GupshupBot/sampleCode/tree/master/currencyConversionBot)

# Deploying your bot locally

Gupshup allows you to write the bot code outside Gupshup's Bot Builder tool in any programming language of your choice.You can then associate a callback to the bot on Gupshup's platform.This callback will be the location of your bot code on the server where you have hosted the code. Whenever the bot is called by the user, Gupshup will send the relevant details to your bot through the callback URL using which you can respond back to the user.

If you are building your bot and are using Gupshup's callback URL feature you can test your bot locally. To do this we recommend this nifty tool developed by [@inconshreveable](https://twitter.com/inconshreveable) called [ngrok](https://ngrok.com/). This tool allows you to test your bot locally and provide a secure, public-facing URL that you can use as your callback URL. Here's a quick guide on how you can start host botcode using ngrok.

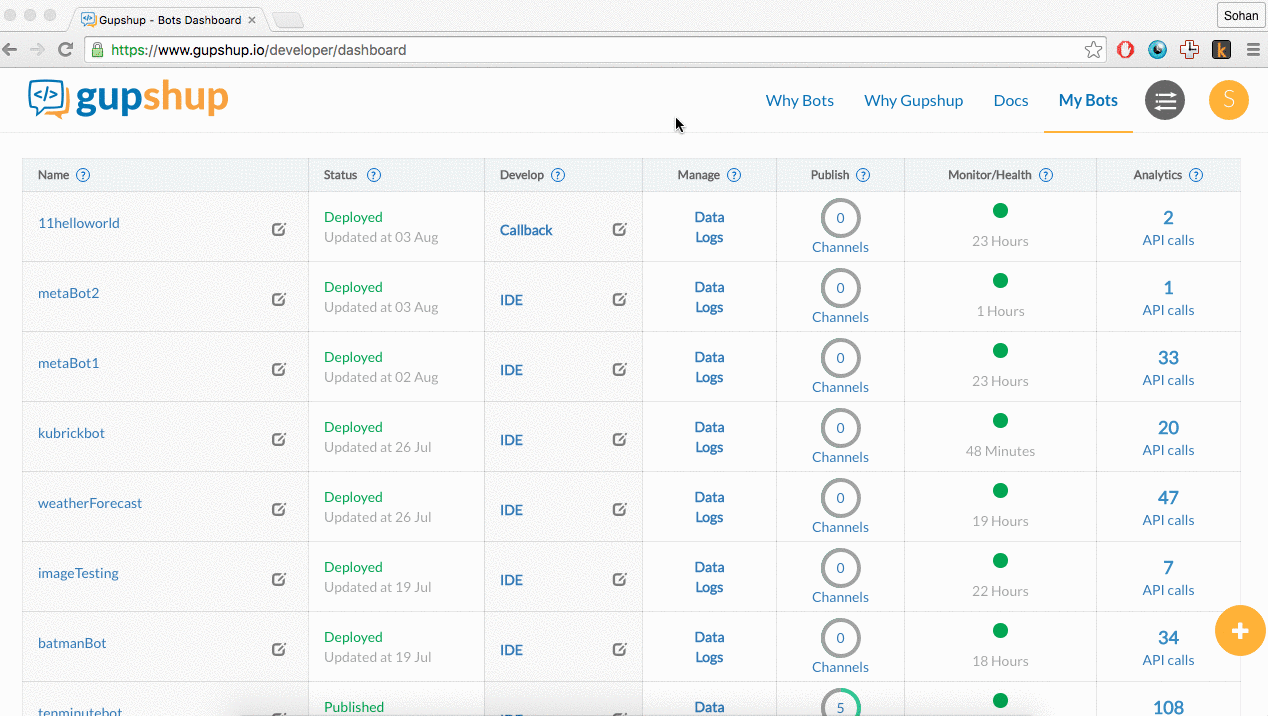
Download ngrok from [here](https://ngrok.com/) and follow the instructions to install it.   
Now we shall create a secure URL to your localhost. Type `ngrok http 1500` in your Terminal. You will see an ngrok window like this.



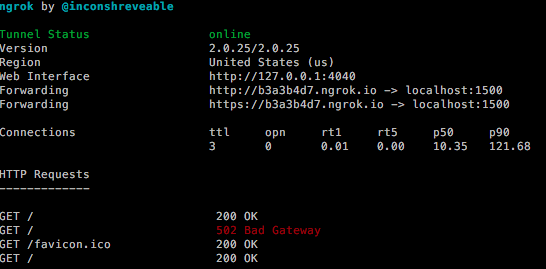
Ports 1-1024 are restricted so use any port number > 1024.

Ngrok has thus created a secure public-facing URL (http://b3a3b4d7.ngrok.io in this instance) to localhost:1500, that you can access from anywhere. Type this URL in your browser and you will see the 'Hello World' message from server.js.

Paste this URL as your callback URL in your Gupshup dashboard.



Your ngrok window will also inspect and display the HTTP requests.



That is how you can test your bot locally using ngrok. Once you're done testing, you can host this file on your production server.