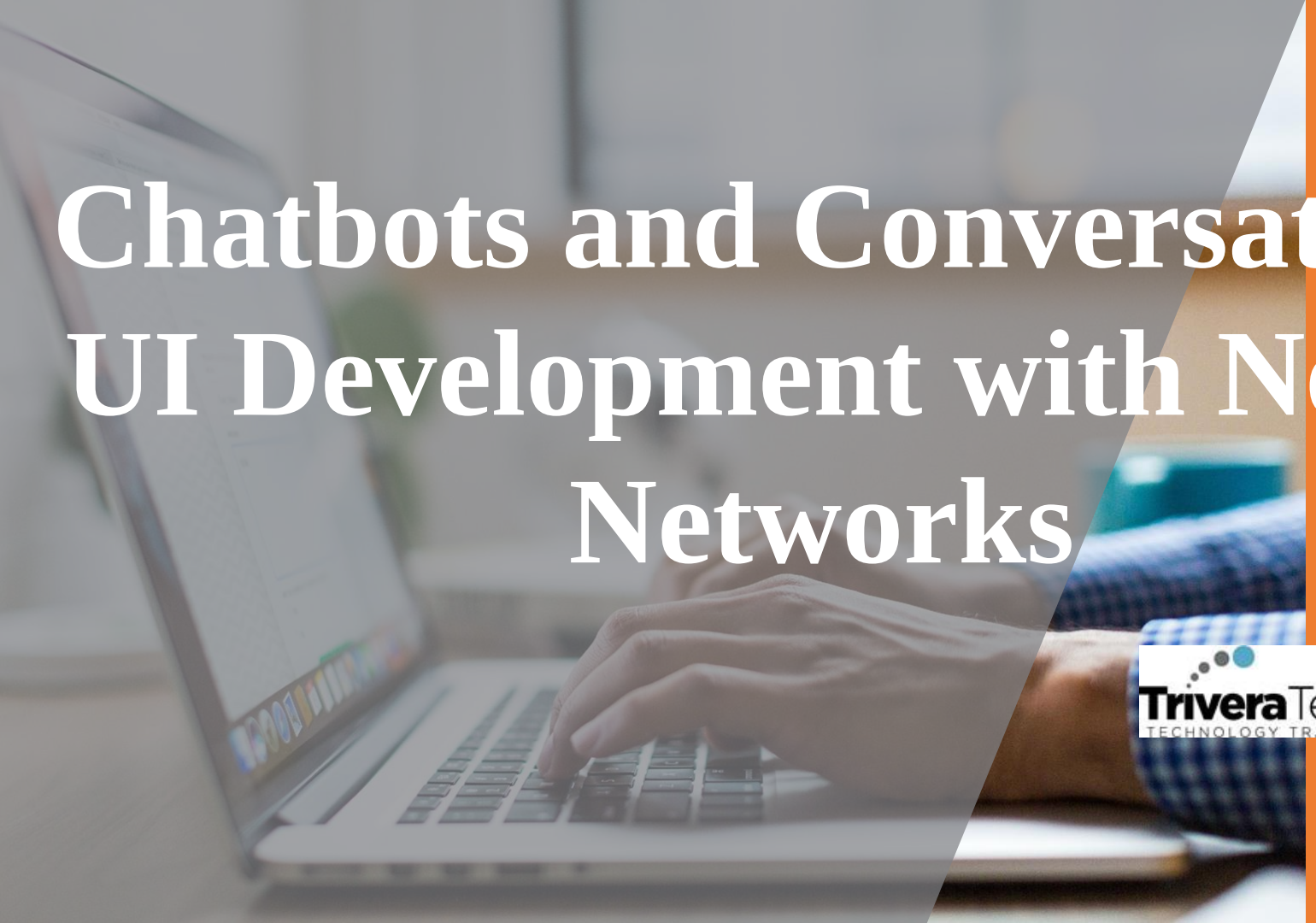
Chatbots and Conversational UI Development with Neural Networks



Trivera Tech
TECHNOLOGY TRAINING

Table of Contents

1. Introduction

TriveraTech
TECHNOLOGY TRAINING

Introduction

- This is a course for programmers beginning to build conversational interfaces.
- Today, basic button-based chatbots can be built without even having to write a single line of code
- We will gradually move toward more complex and flexible architectures, and we will explore channels to use, such as Facebook Messenger, SMS, and Twitter.

Conversational user interfaces

- Conversational user interfaces are as old as modern computers themselves. ENIAC, the first programmable general-purpose computer, was built in the year 1946.

- In 1950, Alan Turing, a British computer scientist, proposed to measure the level of intelligence in machines using a conversational test called the Turing test.

A brief history of chatbots

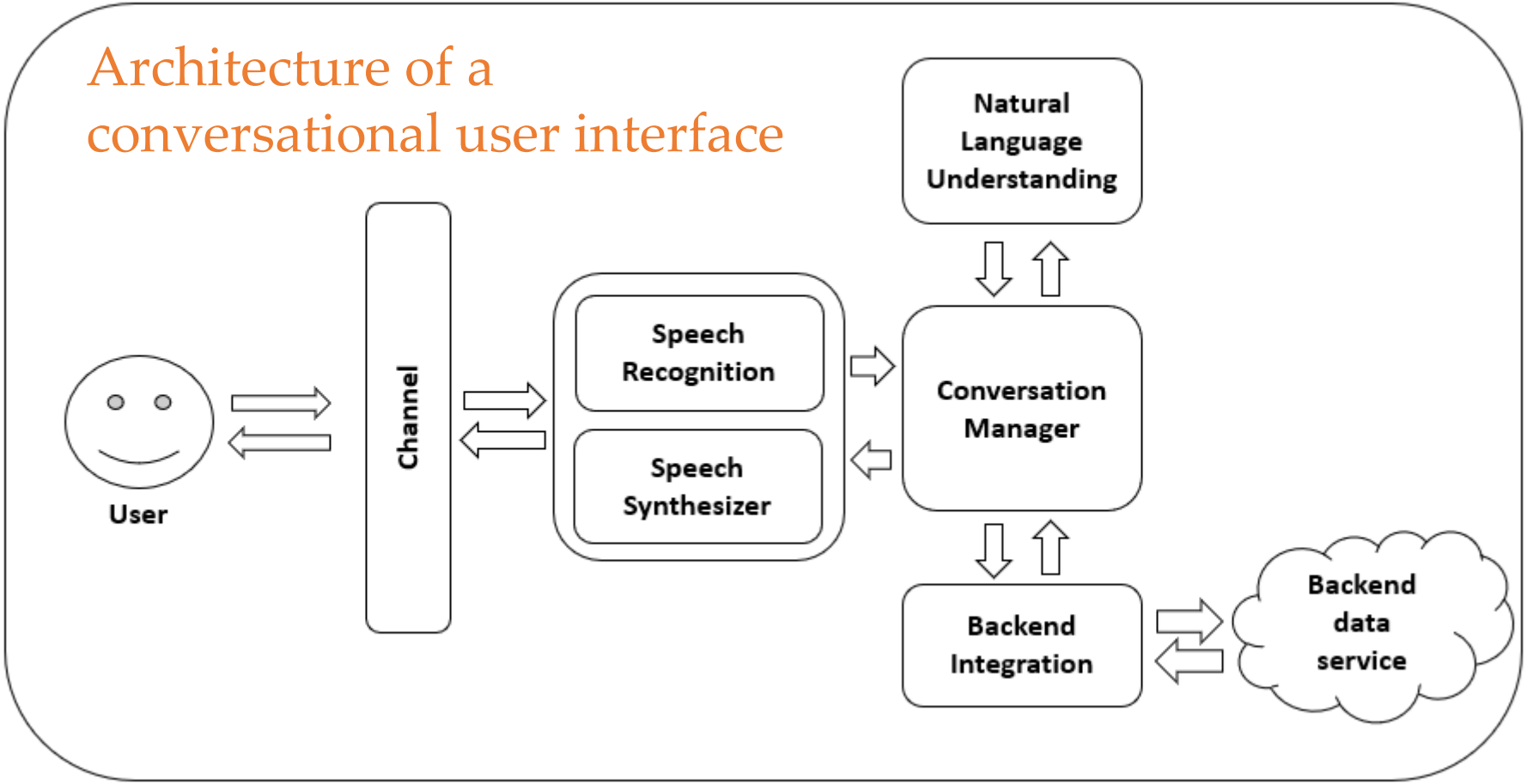
- The origins of modern chatbots can be traced back to 1964 when Joseph Weizenbaum at Massachusetts Institute of Technology (MIT) developed a chatbot called Eliza.

- It used simple rules of conversation and rephrased most of what the users said to simulate a Rogerian therapist.

Recent developments

- In 2011, Apple released an intelligent assistant called Siri as part of their iPhones.
- Siri was modeled to be the user's personal assistant, doing tasks such as making calls, reading messages, and setting alarms and reminders.
- This is one of the most significant events in the recent past that rebooted the story of conversational interfaces.

Architecture of a conversational user interface



Classification

- Conversational user interfaces have found themselves applied in various scenarios.

- Their applications can be classified broadly into two categories: enterprise assistants and personal assistants.

- Enterprise Assistants are chatbots and other conversational user interfaces that are modeled after customer service representatives and store assistants.

Applications

- Although chatbots have been under development for at least a few decades, they did not become mainstream channels for customer engagement until recently.
- Over the past two years, due to serious efforts by industry giants like Apple, Google, Microsoft, Facebook, IBM, and Amazon, and their subsequent investments in developing toolkits, chatbots and conversational interfaces have become a serious contender to other customer contact channels.

Developer's toolkit

Mockup tools

- Mockups can be used to show clients as to how a chatbot would look and behave.
- These are tools that you may want to consider using during conversation design, after coming up with sample conversations between the user and the bot on the back of a napkin.
- Mockup tools allow you to visualize the conversation between the user and the bot and showcase the dynamics of conversational turn-taking.

Channels

- Channels refer to places where users can interact with the chatbot.
- There are several deployment channels over which your bots can be exposed to users.
- These include messaging services such as Facebook Messenger, Skype, Kik, Telegram, WeChat, and Line; office and team chat services such as Slack, Microsoft Teams, and many more.

Chatbot development tools

- There are many tools that you can use to build chatbots without having to code even a single line: Chatfuel, ManyChat, Dialogflow, and so on.

- Chatfuel allows designers to create the conversational flow using visual elements.

- With ManyChat, you can build the flow using a visual map called the FlowBuilder.

Analytics

- Like other digital solutions, chatbots can benefit from collecting and analyzing their usage statistics.
- While you can build a bespoke analytics platform from scratch, you can also use off-the-shelf toolkits that are widely available now.
- Many off-the-shelf analytics toolkits are available that can be plugged into a chatbot, using which incoming and outgoing messages can be logged and examined.

Natural Language understanding

- Chatbots can be built without having to understand utterances from the user.
- However, adding the natural language understanding capability is not very difficult.
- It is one of the hallmark features that sets chatbots apart from their digital counterparts such as websites and apps with visual elements.
- There are many natural language understanding modules that are available as cloud services.

Directory services

- One of the challenges of building the bot is to get users to discover and use it.
- Chatbots are not as popular as websites and mobile apps, so a potential user may not know where to look to find the bot. Once your chatbot is deployed, you need to help users find it.
- There are directories that list bots in various categories.

Monetization

- Chatbots are built for many purposes: to create awareness, to support customers after sales, to provide paid services, and many more.

- In addition to all these, chatbots with interesting content can engage users for a long time and can be used to make some money through targeted personalized advertising.

Benefits

Conversational user interfaces bring in the best of both worlds: human-like natural interaction combined with the benefits of digital technology.

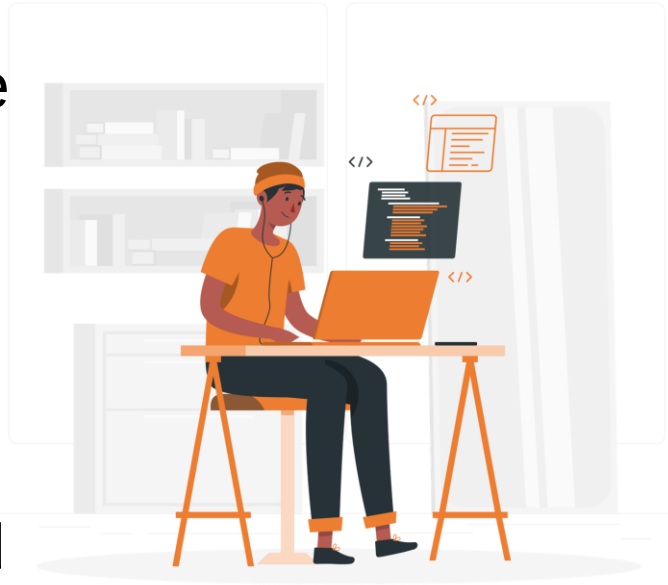
- Availability
- Personalized experience
- Low cost
- Consistency
- Quick response times
- Scale up

Chatbots are here to stay

- The conversational user interface technologies are currently one of the top trending topics in the technology business.
- Most big brands have started formulating their chatbot strategy within their larger AI and automation strategy.
- Innovations such as chatbots, smart speakers, and self-driving cars are driving such major policy decisions.

Lets get started!

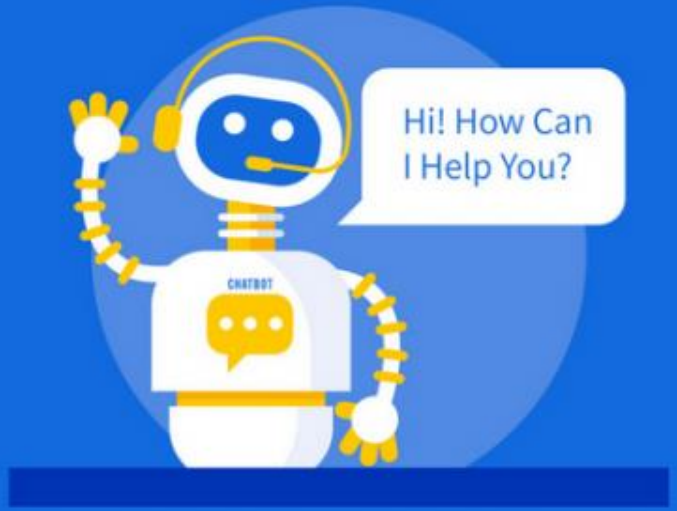
- So are you ready to get started and build some chatbots yet? I hope I have given you a good introduction to the world of chatbots in this lesson.
- We covered historical and recent developments, classification of chatbots, their application in various sectors, their benefits, their future, and their basic architecture.



2. Build a Chatbot using Python/Flask

TriveraTech
TECHNOLOGY TRAINING

Build a Chatbot using Python/Flask



Build a Chatbot using Python/Flask

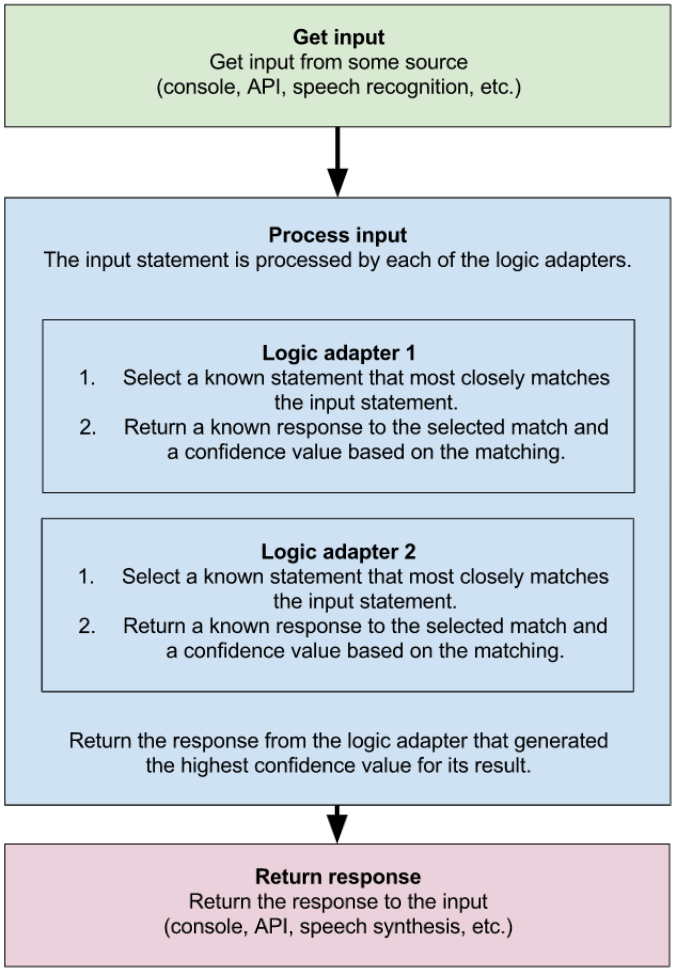
An example of typical input would be something like this:

- user: Good morning! How are you doing?
- bot: I am doing very well, thank you for asking.
- user: You're welcome.
- bot: Do you like hats?

How ChatterBot Works

- ChatterBot is a Python library designed to make it easy to create software that can engage in conversation.
- An untrained instance of ChatterBot starts off with no knowledge of how to communicate.
- Each time a user enters a statement, the library saves the text that they entered and the text that the statement was in response to.

Process flow diagram



Installing dependencies

```
pip install chatterbot
pip install chatterbot_corpus
```

Importing Classes – Getting started!!

```
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer
from chatterbot.trainers import ListTrainer
```

Creating the bot

- We are creating a Flask app, to get started with Flask, you can visit here

```
app = Flask(__name__)
#bot = ChatBot("Pikachu")
```

- We can create and train the bot by creating an instance of ListTrainer and supplying it with the lists of strings:

```
trainer = ListTrainer(bot)
```

Creating the bot

- Getting started with the training part, there are different ways how we can train the bot, by this,

```
trainer.train(['What is your name?', 'My name is Pikachu'])
trainer.train(['How are you?', 'I am good' ])
trainer.train(['Bye?', 'Bye, see you later' ])
```

Creating the bot

```
conversation = [
    "Hello",
    "Hello!!",
    "How are you doing?",
    "I'm doing great.",
    "That is good to hear",
    "Thank you.",
    "You're welcome."
]

trainer.train(conversation)
```

Training the Bot with corpus of data

- You can use your own or an existing corpus of data to train a bot.

- For example, you can use some corpus provided by chatterbot (inbuilt features):

```
corpus_trainer = ChatterBotCorpusTrainer(bot)
corpus_trainer.train('chatterbot.corpus.english')
```

Training the Bot with corpus of data

```
@app.route("/")
def home():
    return render_template("home.html")
@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return str(bot.get_response(userText))
if __name__ == "__main__":
    app.run()
```

Training the Bot with corpus of data

- Go to Anaconda Prompt, and run the below query.

python app.py

- Below message in Python shell is seen, which indicates that our App is now hosted at http://127.0.0.1:5000/ or localhost:5000

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)


Welcome to Chatbot Class!
Chat with us
Hello!!
How are you?
I am good
Good to hear that
Thank you.
Welcome
Unless explicitly silenced.
Bye
Bye, see you later
Message
Send

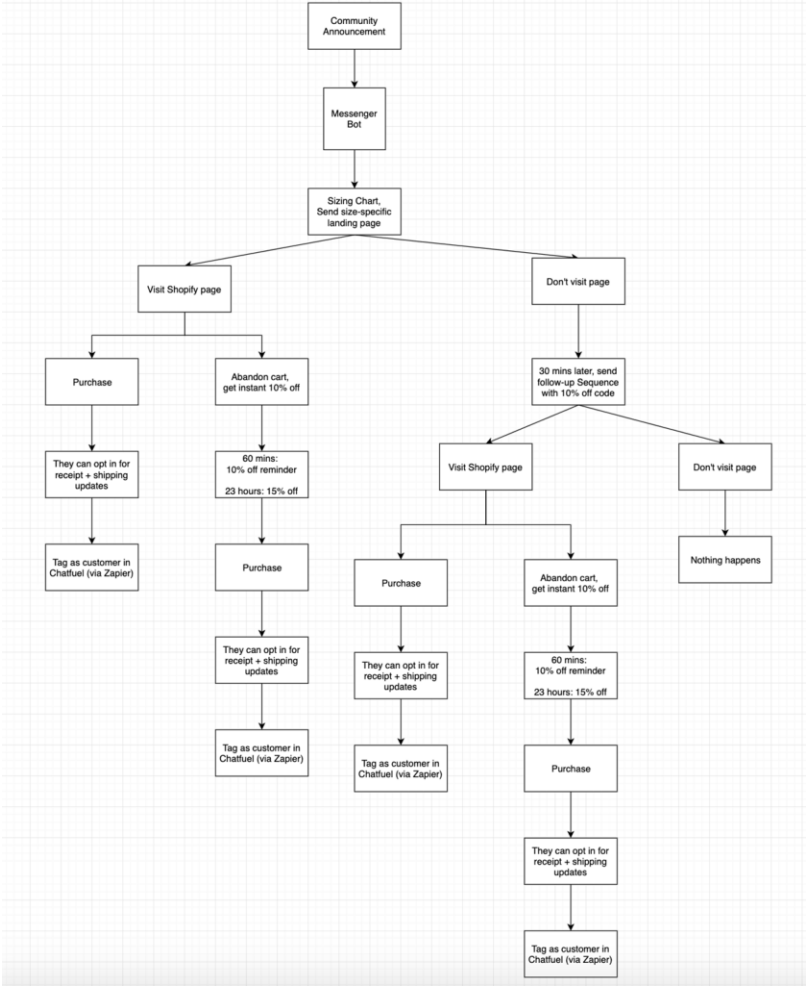
3. Getting Started With Chatfuel

TriveraTech
TECHNOLOGY TRAINING

Laying the Groundwork for Your First Chatbot

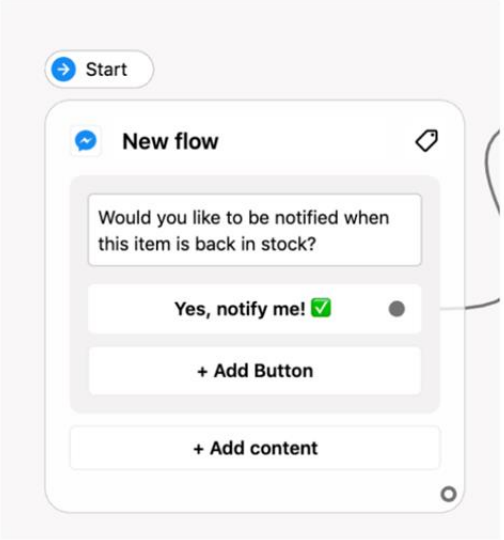
1. Choose a goal and map your flow.

- Decide on the goal for your bot. Is it to gather and qualify leads?
- Answer questions?
- Push people to your online store to make a purchase?
- Choose just one main purpose to start with, and map out a conversation of questions and answer options that will bring users to that desired end point

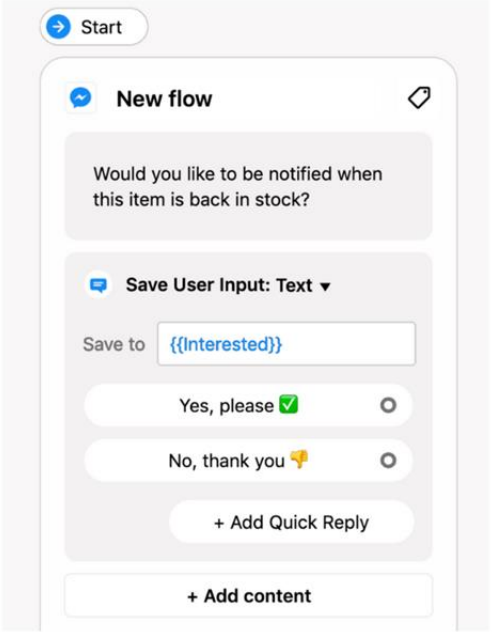

Community Announcement
Messenger Bot
Sizing Chart, Send size-specific landing page
Visit Shopify page
Don't visit page
Purchase
Abandon cart, get instant 10% off
30 mins later, send follow-up Sequence with 10% off code
They can opt in for receipt + shipping updates
60 mins: 10% off reminder
23 hours: 15% off
Visit Shopify page
Don't visit page
Tag as customer in Chatfuel (via Zapier)
Purchase
Purchase
Abandon cart, get instant 10% off
Nothing happens
They can opt in for receipt + shipping updates
They can opt in for receipt + shipping updates
60 mins: 10% off reminder
23 hours: 15% off
Tag as customer in Chatfuel (via Zapier)
Tag as customer in Chatfuel (via Zapier)
Purchase
They can opt in for receipt + shipping updates
Tag as customer in Chatfuel (via Zapier)

2- Plan a bot that's proactive.

Buttons



Quick replies



3- Plan your bot's persona.

- Your bot will act as a virtual ambassador for your brand, so it needs to make the right first impression.

- 👋 Plan out a friendly Welcome Message that lets the user know they’re chatting with a bot, and tells them how it can help.

- You may even want to choose a name for your chatbot.

4- Plan for compliance with Facebook's Messenger policy.

- Facebook has rules for how businesses can communicate with users via Messenger bot.

- These rules exist to protect users from spam, and to make sure they get maximum value from Messenger.

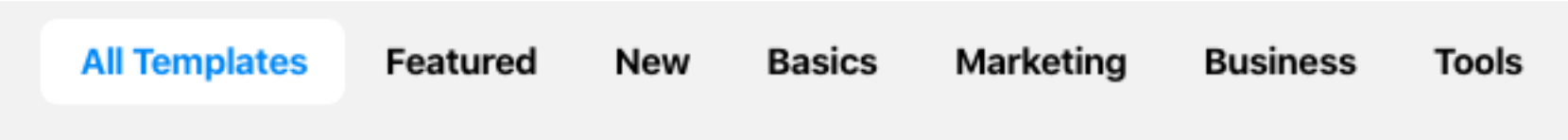
Building Your First Chatbot—Fast

What are Chatfuel's bot templates?

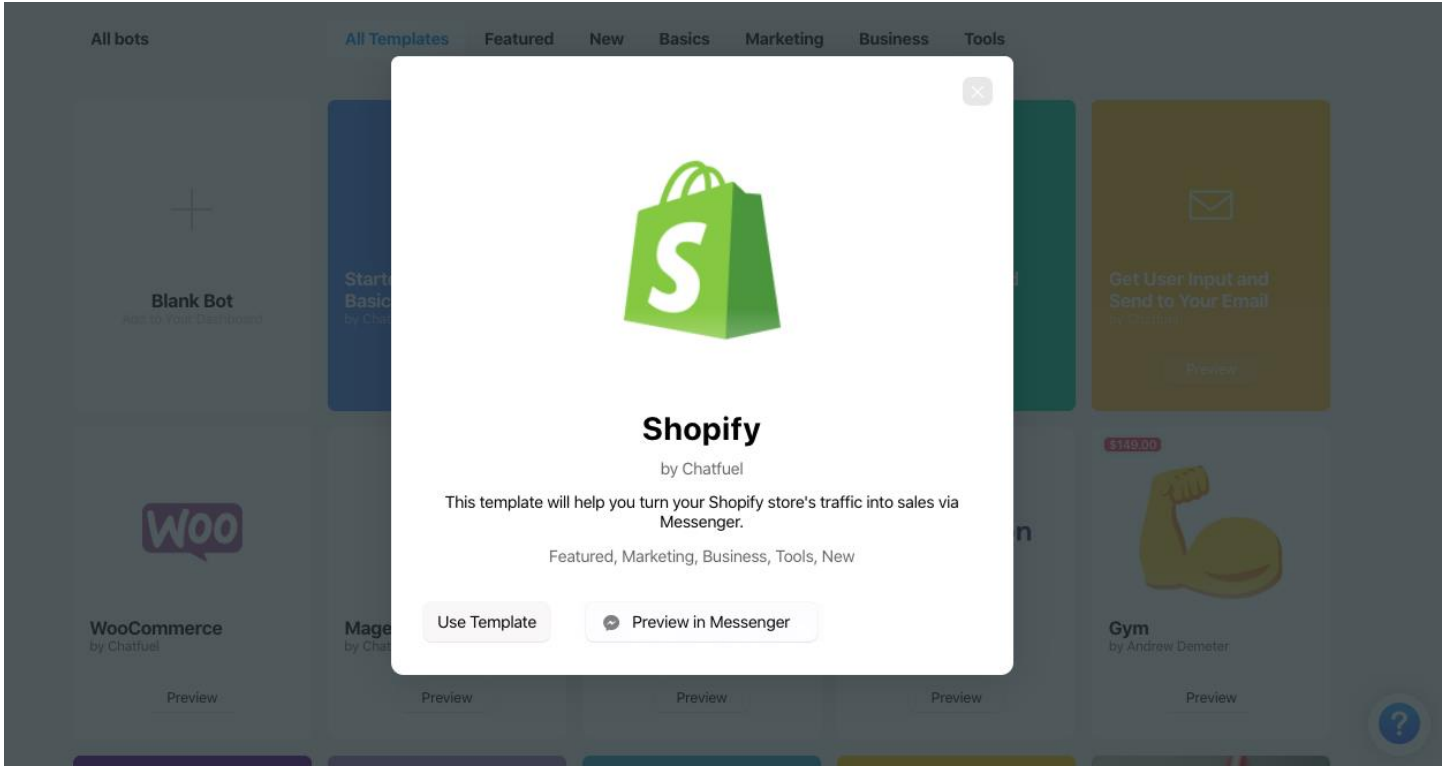
- Our bot templates are fully formed frameworks for an entire bot.
- For chatbots in the Automate tab
- There are almost 100 templates available for bots built in the Automate tab.
- You can find these in Chatfuel's templates library (by clicking Choose from template when you first log in to your Chatfuel dashboard), and most are free.

How to use a bot template

- For chatbots in the Automate tab
- You'll find our bot templates in your Chatfuel dashboard.
- They're organized into categories in the bar at the top of the screen.



Using a bot template is simple:



Shopify
SAVED

Automate
Live Chat
Set Up AI
People
Reengage
Configure
Grow
Analyze
Upgrade

Welcome Message
Every person communicating with the bot sees this block first.

Default Answer
A person will see this block if the bot does not recognize a text

CORE COMPONENTS

Main Menu	About Us	+

CUSTOMER SERVICE

Customer Service Menu	Live Chat	Other Contact Options
Main FAQ	FAQ Q1	FAQ Q2
FAQ Q3	+	

TOP CATEGORIES / PRODUCTS

Category Menu	Cat 1	Cat 2
Cat 3	Cat 1 Prod 1	Cat 1 Prod 2
Cat 1 Prod 3	Cat 2 Prod 1	Cat 2 Prod 2
Cat 2 Prod 3	Cat 3 Prod 1	Cat 3 Prod 2
Cat 3 Prod 3	+	

FACEBOOK AD NEXT STEPS

Welcome Message

⭐*READ THIS FIRST* ⭐

Before you begin using this template, check out our E-commerce Playbook:
👉 *http://tiny.cc/CFecom* 👈

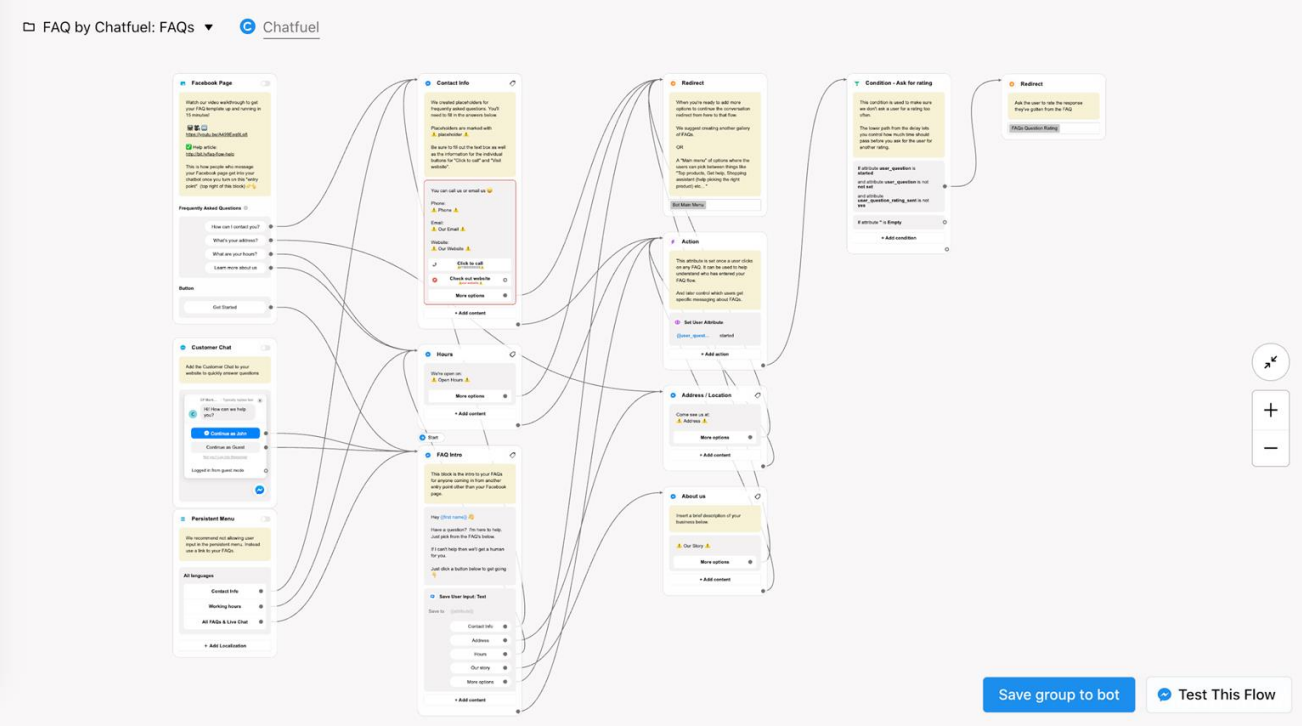
Users seeing the welcome message will likely have come directly from your Facebook page. Typically they'll want to ask a question or want help with an order.

Here are some Welcome Message best practices: https://youtu.be/y4ZFIv8us5E

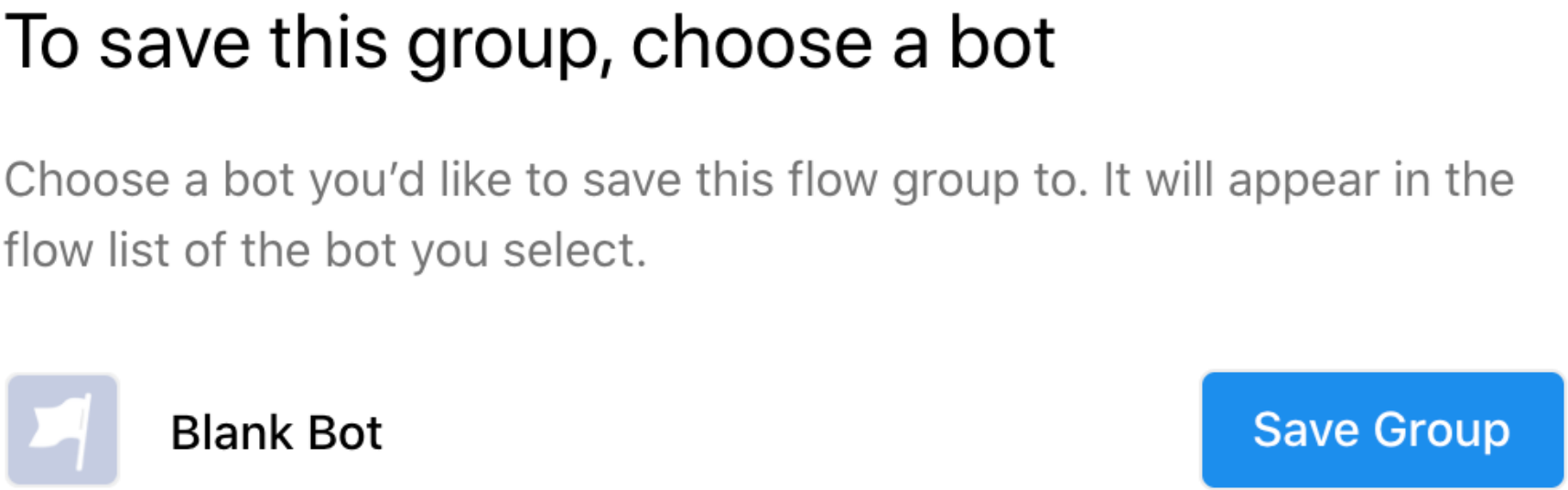
{first name} good to see you!

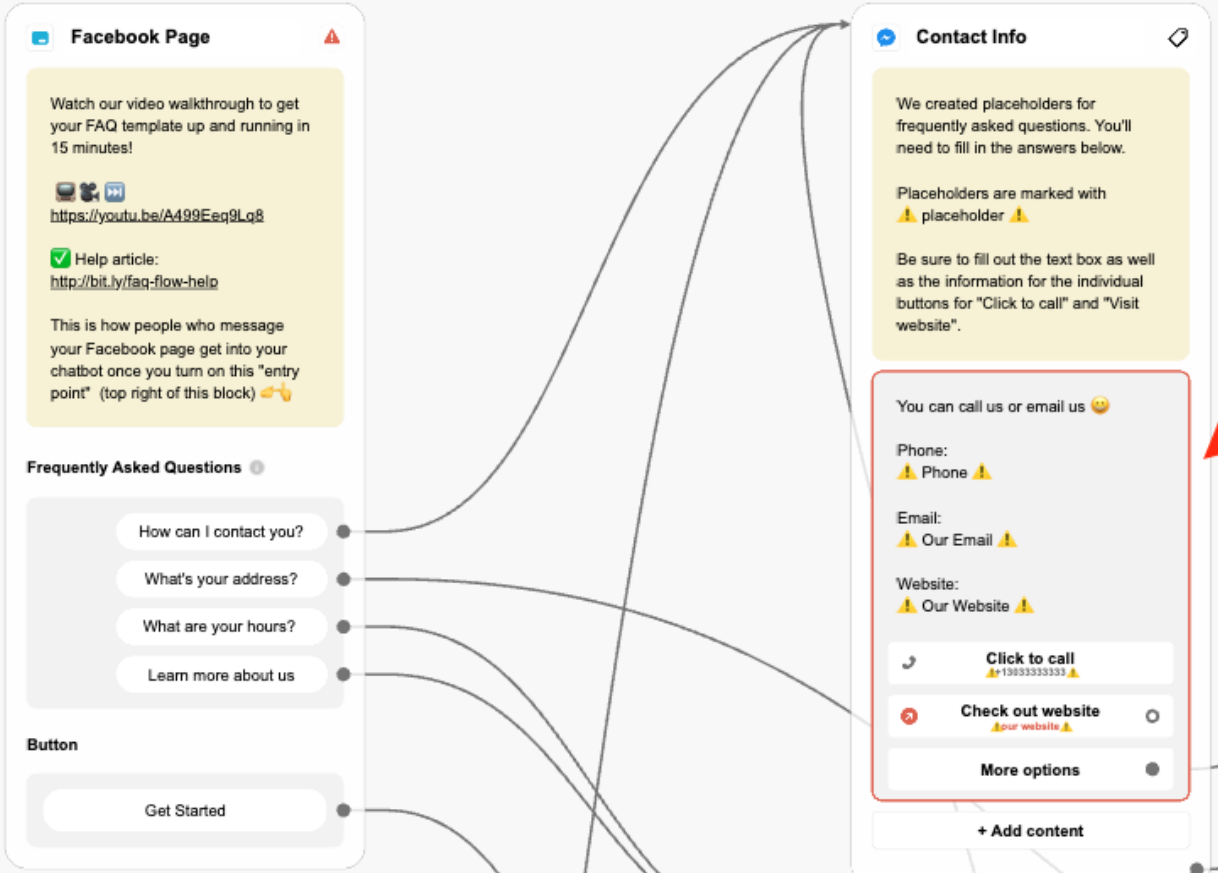
I'm the store's Chatbot. Here's what I can help with:

For chatbots in the Flows tab



For chatbots in the Flows tab




Facebook Page
Watch our video walkthrough to get your FAQ template up and running in 15 minutes!
https://youtu.be/A499Eeq9Lq8
Help article:
http://bit.ly/faq-flow-help
This is how people who message your Facebook page get into your chatbot once you turn on this "entry point" (top right of this block)
Frequently Asked Questions
How can I contact you?
What's your address?
What are your hours?
Learn more about us
Button
Get Started
Contact Info
We created placeholders for frequently asked questions. You'll need to fill in the answers below.
Placeholders are marked with
placeholder
Be sure to fill out the text box as well as the information for the individual buttons for "Click to call" and "Visit website".
You can call us or email us
Phone:
Phone
Email:
Our Email
Website:
Our Website
Click to call
Check out website
More options
+ Add content

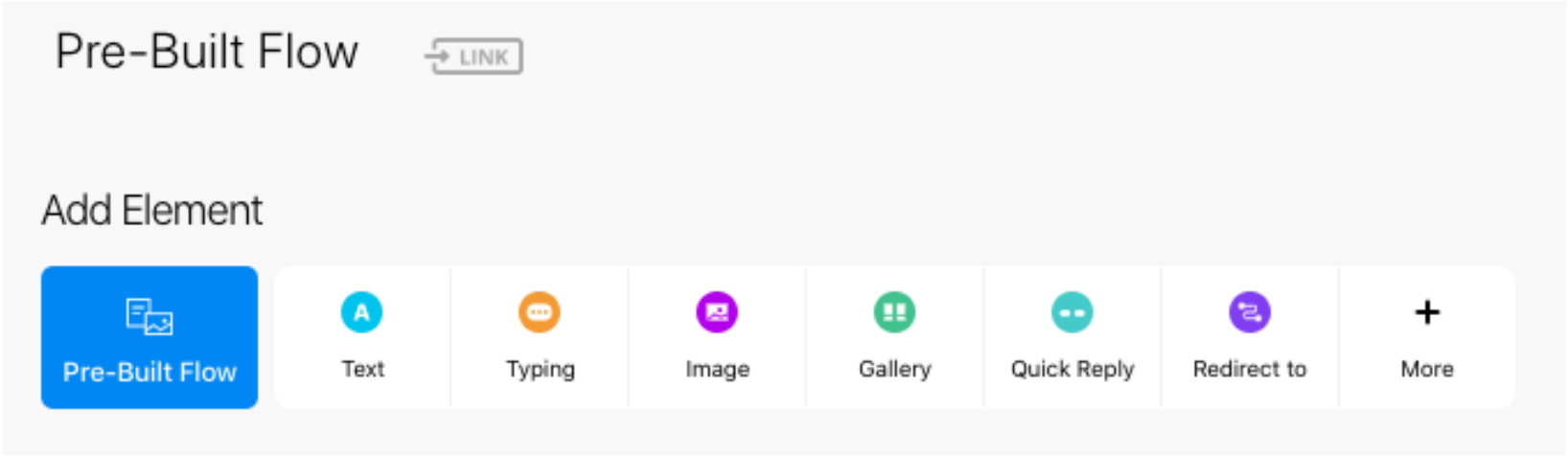
What are Chatfuel’s pre-built flows?

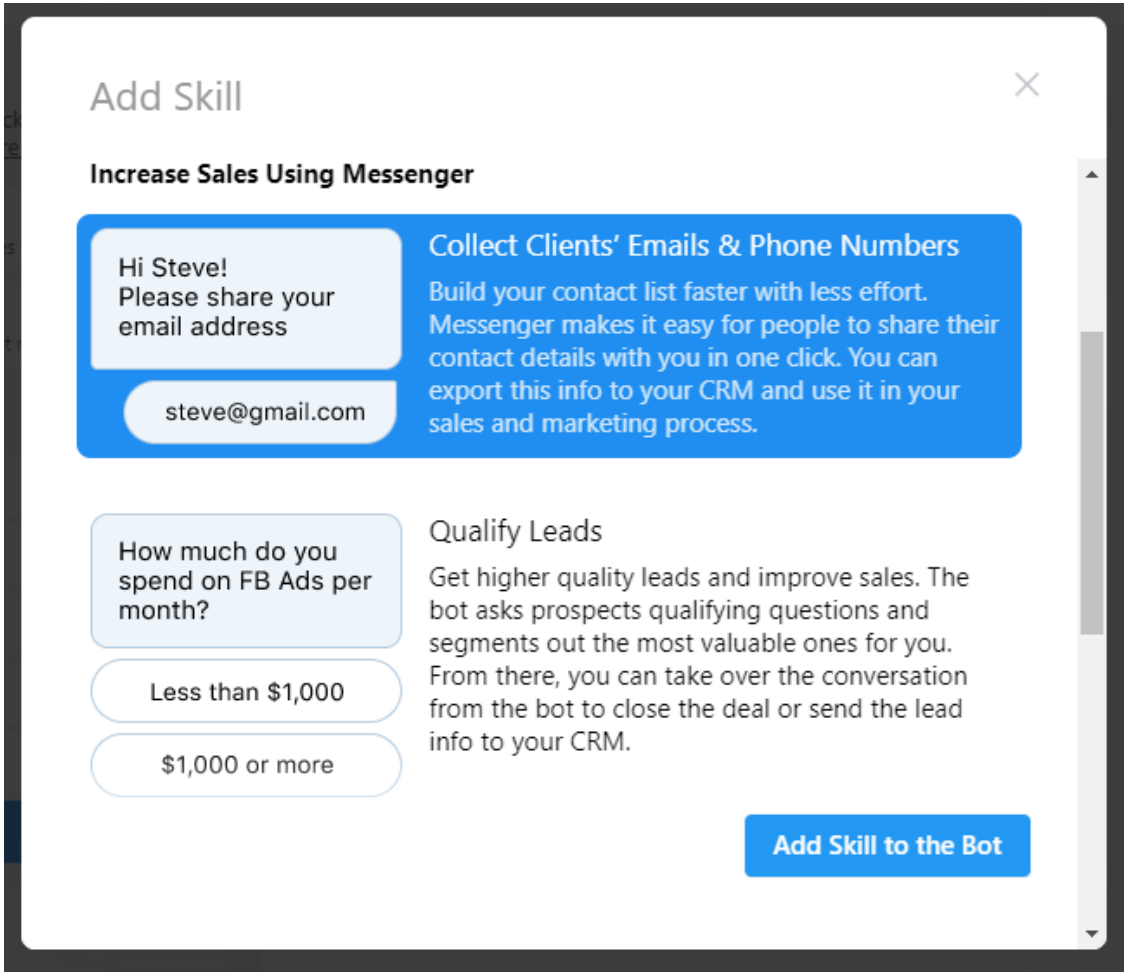
- In Chatfuel, you’ll find pre-set bot flows for four basic functions, or "skills:"

1. Saving time by automating FAQs
2. Increasing sales by collecting emails and phone numbers
3. Qualifying leads
4. Taking over the conversation with live chat

How to use a pre-built flow

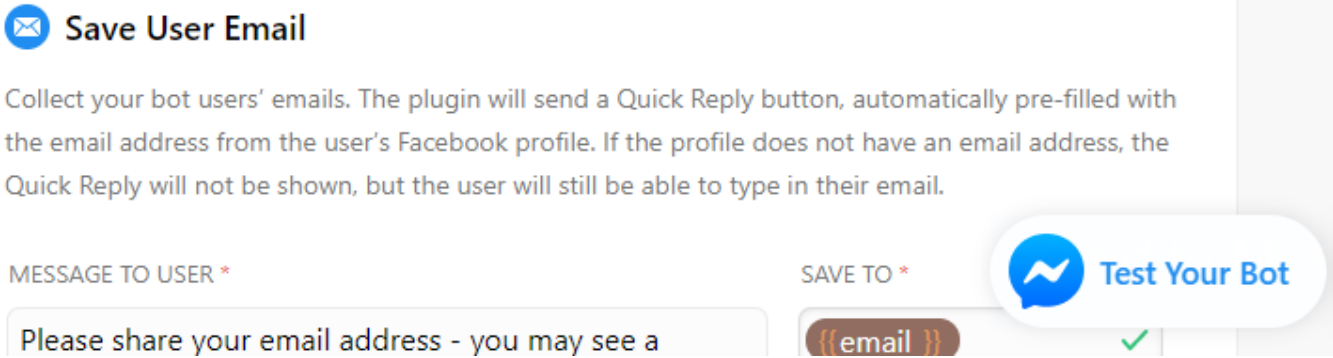
- You'll find all the pre-built flows in the Add Element bar at the bottom of the work area in the Automate tab. ↴



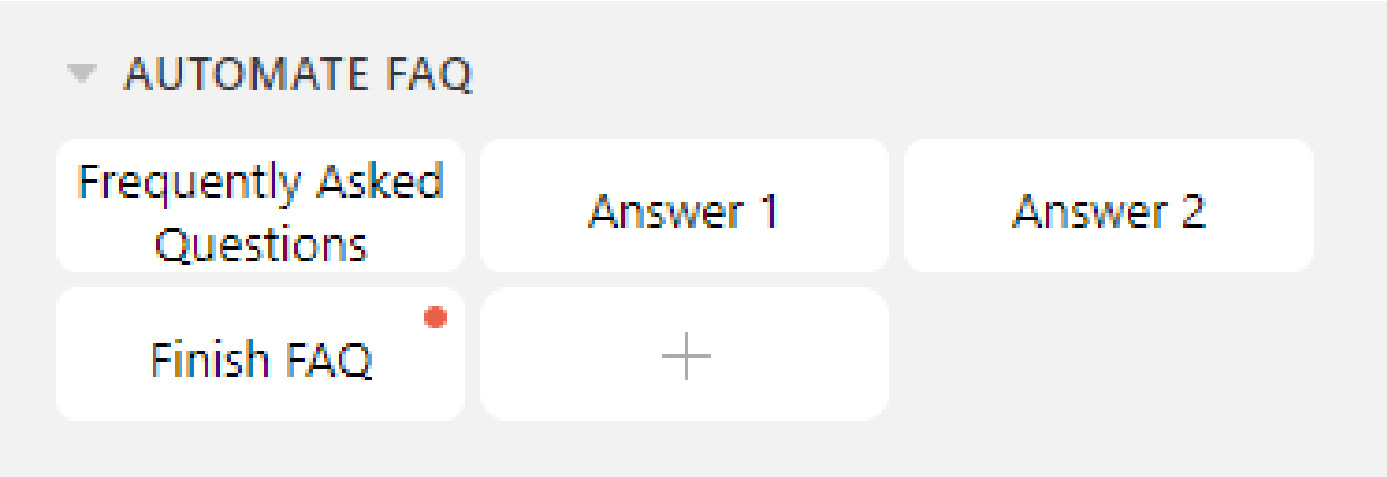

Add Skill
Increase Sales Using Messenger
Hi Steve!
Please share your email address
steve@gmail.com
Collect Clients' Emails & Phone Numbers
Build your contact list faster with less effort. Messenger makes it easy for people to share their contact details with you in one click. You can export this info to your CRM and use it in your sales and marketing process.
How much do you spend on FB Ads per month?
Less than $1,000
$1,000 or more
Qualify Leads
Get higher quality leads and improve sales. The bot asks prospects qualifying questions and segments out the most valuable ones for you. From there, you can take over the conversation from the bot to close the deal or send the lead info to your CRM.
Add Skill to the Bot

TO-DO

- [ ] (optional) Edit the message asking a person to share their contact details
- [ ] Edit the message at the bottom that appears when a person provides the information
- [ ] If you don't need both email and phone, delete one of the plugins
- [ ] (optional) Teach your bot another skill at the bottom of the block to continue interacting with the person



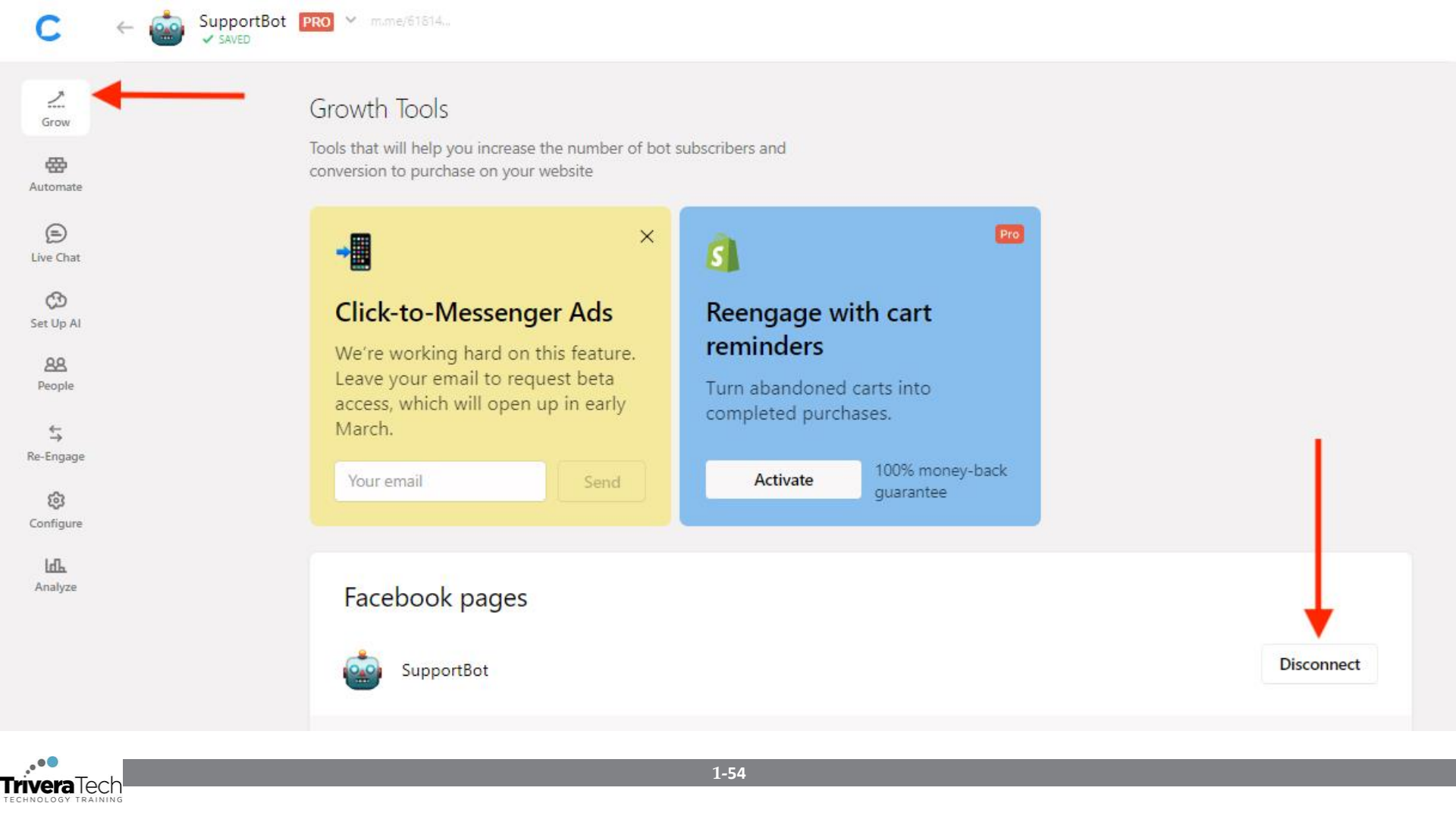
How to use a pre-built flow

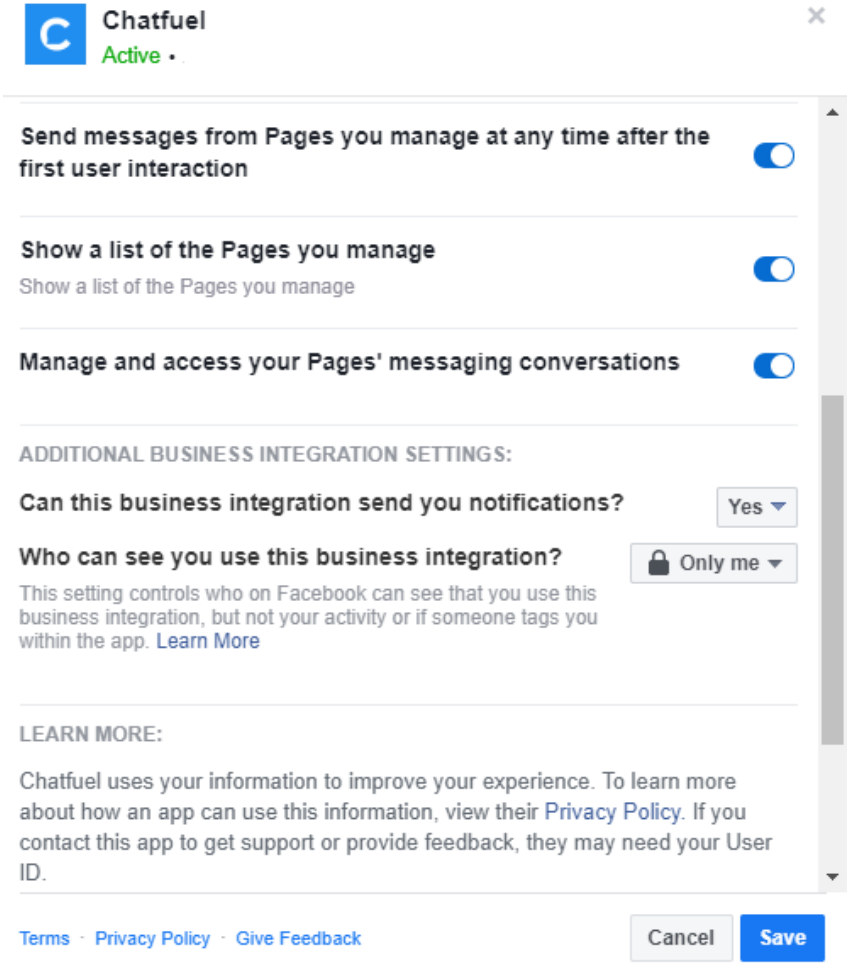


Connecting Your Bot to a Facebook Page

Once you've built your Messenger chatbot, you can set it live (so it can communicate with users) by connecting it to a Facebook page. Here's how:

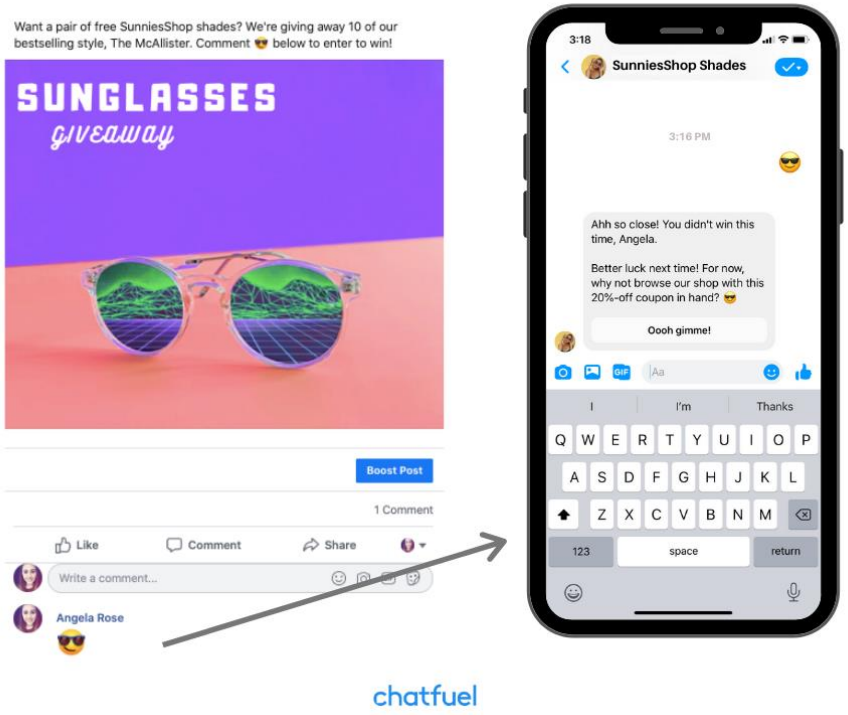
1. Go to the Grow tab in Chatfuel.
2. Under the Facebook pages section, you'll see a list of pages associated with your account.
3. Choose the Facebook page you want to connect your bot to, and click the Connect button next to that page.


SupportBot
PRO
SAVED
Grow
Automate
Live Chat
Set Up AI
People
Re-Engage
Configure
Analyze
Growth Tools
Tools that will help you increase the number of bot subscribers and conversion to purchase on your website
Click-to-Messenger Ads
We're working hard on this feature. Leave your email to request beta access, which will open up in early March.
Your email
Send
Pro
Reengage with cart reminders
Turn abandoned carts into completed purchases.
Activate
100% money-back guarantee
Facebook pages
SupportBot
Disconnect

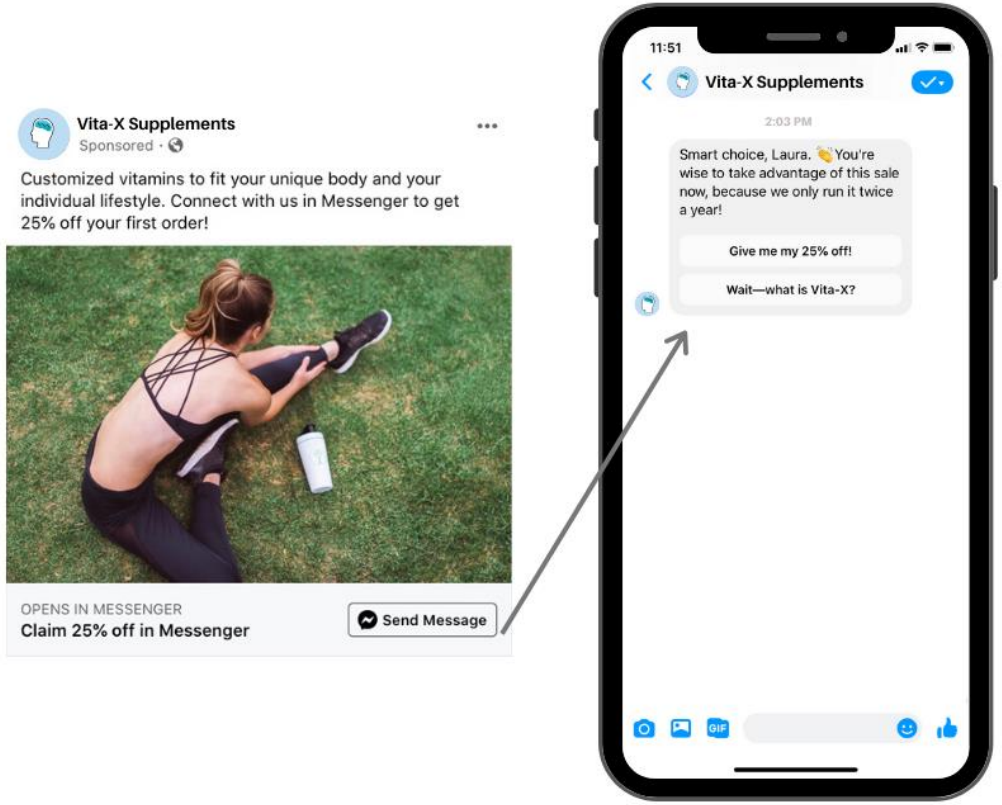

Chatfuel
Active
Send messages from Pages you manage at any time after the first user interaction
Show a list of the Pages you manage
Show a list of the Pages you manage
Manage and access your Pages' messaging conversations
ADDITIONAL BUSINESS INTEGRATION SETTINGS:
Can this business integration send you notifications?
Yes
Who can see you use this business integration?
Only me
This setting controls who on Facebook can see that you use this business integration, but not your activity or if someone tags you within the app. Learn More
LEARN MORE:
Chatfuel uses your information to improve your experience. To learn more about how an app can use this information, view their Privacy Policy. If you contact this app to get support or provide feedback, they may need your User ID.
Terms · Privacy Policy · Give Feedback
Cancel
Save

Bringing New Users to Your Bot

Promote your bot on Facebook.

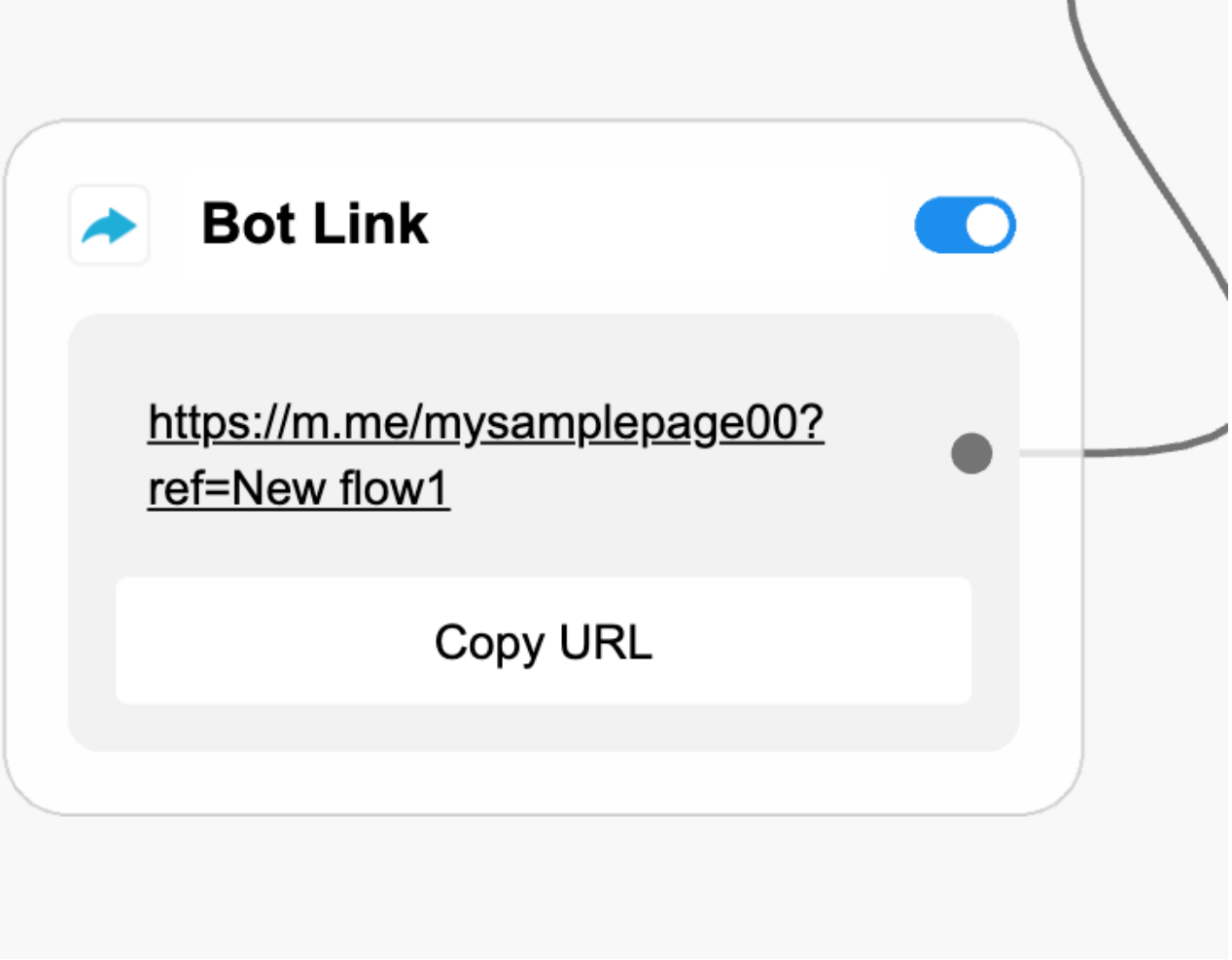



Petfuel Pet Supplies
February 21 at 6:38 PM ·
Need a recommendation on puppy food? Want help finding the perfect toy for your pup? Reach out to us; we'd love to help! 👇
Petfuel Pet Supplies
Pet Supplies
Send Message


Vita-X Supplements
Sponsored
Customized vitamins to fit your unique body and your individual lifestyle. Connect with us in Messenger to get 25% off your first order!
OPENS IN MESSENGER
Claim 25% off in Messenger
Send Message
11:51
Vita-X Supplements
2:03 PM
Smart choice, Laura. You're wise to take advantage of this sale now, because we only run it twice a year!
Give me my 25% off!
Wait—what is Vita-X?
GIF

chatfuel

Share the link to your bot.


Bot Link
https://m.me/mysamplepage00?ref=New flow1
Copy URL

Of the People

Enough about the candidates. What about the voters? Read our special photo essay on what the people have to say on the issues that matter most.

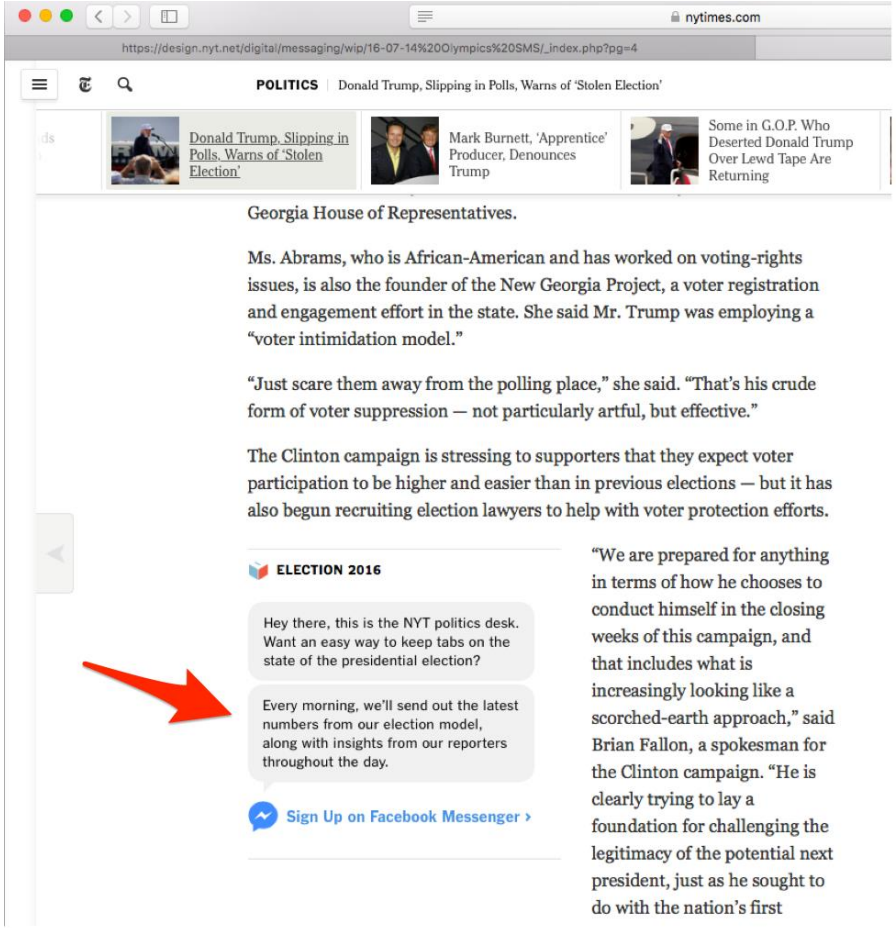
READ NOW

Want an easy way to keep up?

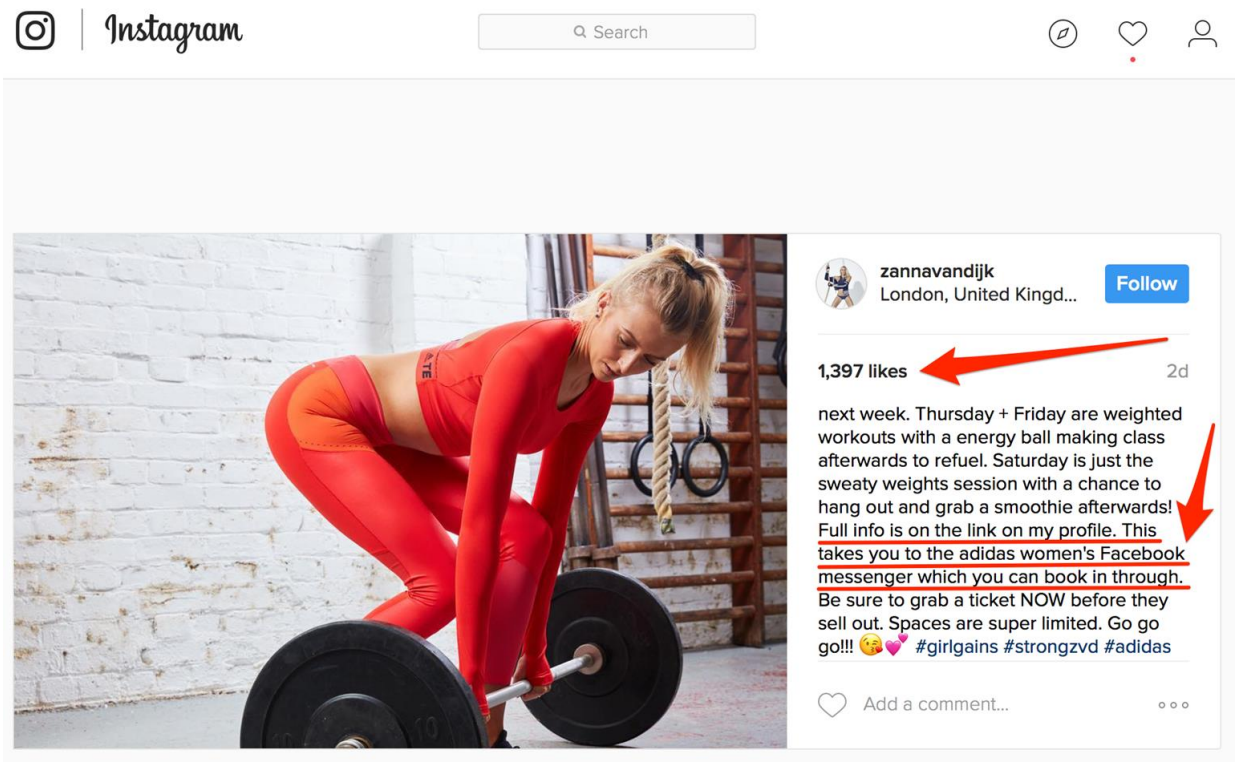
Every morning, we will send out the latest numbers from our election forecast, along with insights from our reporters throughout the day.

SIGN UP

Promote your bot on other channels.



Share your bot on Instagram




Sites
ABC
Search
NEWS
SET LOCATION
for local news & weather
Just In
US Election
Australia
World
Business
Sport
Arts
Analysis & Opinion
Programs
More
ABC News on Facebook Messenger
Welcome to ABC News on Messenger. Stay across the news people will be talking about with daily digests and breaking news alerts all within Facebook Messenger.
Thanks, I'm done
You can get back in touch at any time by messaging "latest news" or "coming up", or by tapping on the menu icon in the bottom left of the screen.
Type a message...

Golden State Warriors

@warriors nba.com

2.6K 3

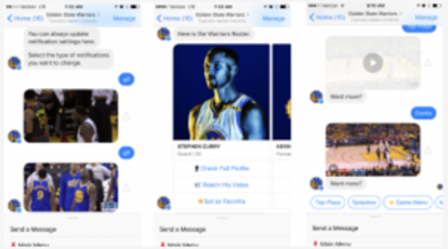
Have a question about the Dubs in the postseason? The Warriors' new bot for Facebook Messenger can find your answer!

GET THIS BOT ON

MESSENGER

AMAZON ECHO

MEDIA



4. Let's Talk Weather

TriveraTech
TECHNOLOGY TRAINING

Let's Talk Weather

By the end of this lesson, you will be able to:

- Design conversational tasks to talk about the weather
- Create backend integrations using the OpenWeatherMap API,
- Build a chatbot in Java 8

Conversational tasks

Let's first figure out the tasks that we want the chatbot to perform. I would think that a chatbot capable of talking about the weather should be able to do it along the following lines:

- Weather now
- Weather today
- Weather this week
- Weather this weekend
- Weather in the future

Conversational design

- Example 1:

User: Hi
Bot: Hi there! I am WeatherMan, your weather bot. What would you like to know? Current weather or forecast?
User: current weather
Bot: Ok. Which city?
User: London, GB
Bot: Ok. Weather now in London, GB. Temperature is 10 degrees Celsius. Clear Skies.
User: Thanks.
Bot: No problem! :)

Conversational design

- Example 2:

User: Hi there
Bot: Hi there! I am WeatherMan, your weather bot. What would you like to know? Current weather or forecast?
User: Forecast
Bot: Ok. When? Tomorrow or this weekend?
User: Tomorrow
Bot: Ok. Which city?
User: Edinburgh
Bot: Edinburgh, US or Edinburgh, GB?
User: Edinburgh, GB
Bot: Ok. Weather tomorrow in Edinburgh, GB. Temperature is 12 degrees Celsius. Cloudy.

Backend tasks

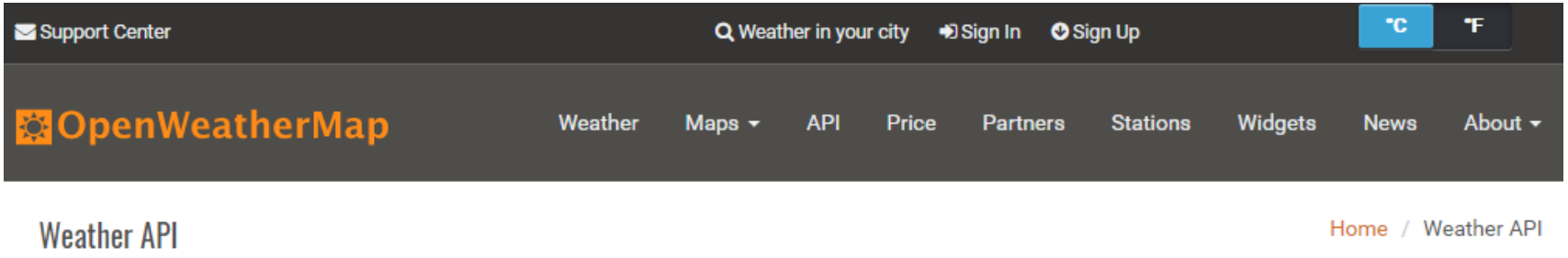
- Before we move on to implementing the chatbot, let us take a look at the source of information for weather data, OpenWeatherMap (https://openweathermap.org).

- OpenWeatherMap is a cloud service serving weather info about 200,000+ cities across the globe.

Getting the API key

- To get started with the data, we will have to subscribe to the appropriate API service.

- To do this, navigate through the website by clicking the API option on the main menu on the home page:



Subscribe to current weather, forecasts, and historical data collections and enjoy our fast simple API!

Current weather and forecasts collection

	Free	Startup	Developer	Professional	Enterprise
Price Price is fixed, no other hidden costs.	**Free**	**40 USD / month**	**180 USD / month**	**470 USD / month**	**2,000 USD / month**
Subscribe	Get API key and Start	Subscribe	Subscribe	Subscribe	Subscribe
Calls per minute (no more than)	60	600	3,000	30,000	200,000
Current weather API	✓	✓	✓	✓	✓
5 days/3 hour forecast API	✓	✓	✓	✓	✓
16 days/daily forecast API	-	-	✓	✓	✓
Weather maps API	✓	✓	✓	✓	✓

API keys

Activation of an API key for **Free** and **Startup accounts** takes **10 minutes**. For **other accounts** it takes from **10** to **60 minutes**.
You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

There is no api keys. Generate new one.

Create key

* Name

Generate

Trying your key

- We have to construct the URL to get the data we need. Here is a basic one:

http://api.openweathermap.org/data/2.5/forecast?id=<CITY_CODE>&APPID=<YOUR_API_KEY>

https://openweathermap.org/city/2643743

Support Center | Weather in your city | Sign In | Sign Up | °C | °F

OpenWeatherMap

Weather | Maps | API | Price | Partners | Stations | Widgets | News | About

Weather forecast

Home / Weather forecast

Your city name | Search

Current weather and forecasts in your city

Main | Daily | Hourly | Chart | Map

Weather in London, GB

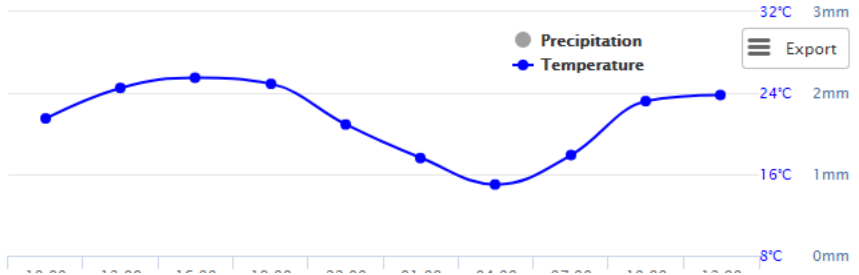
14 °C

Fog

07:11 Jun 14 Wrong data?

Wind	Calm, 1.5 m/s, NorthEast (40)
Cloudiness	Sky is clear
Pressure	1019 hpa

Weather and forecasts in London, GB



```
{
    "cod": "200",
    "message": 0.003,
    "cnt": 37,
    "list": [ … ], // 37 items
    "city": {
        "id": 2643743,
        "name": "London",
        "coord": {
            "lat": 51.5085,
            "lon": -0.1258
        },
        "country": "GB"
    }
}
```


Trying your key

- The list key with 37 items can be expanded, in that you will find current and forecast information for every three hours starting from the current time:

Building the backend interface

1. Open Eclipse.
2. Create a new Maven project.
3. Choose Create a simple project.
4. Provide the location of the project and hit Next.
5. On the next page, type Group Id, Artifact Id, Name, and Description. Click Finish.

Building the backend interface

- We now have a blank Maven project ready.
- Before we move on to developing the backend code, let's add a few dependency packages to our project.
- Find the POM file (pom.xml) and add the following Maven dependencies:

Refer to the file 3_1.txt

Building the backend interface

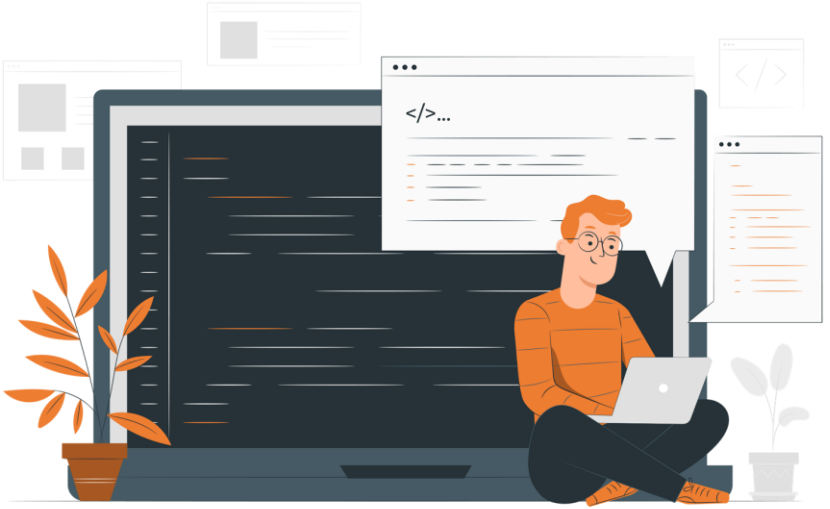
- We may need other dependencies later. But for the backend code, these packages will suffice.
- Let's now create a Java class, Weather.java, to access weather data.
- The following code shows the basic structure of the class:

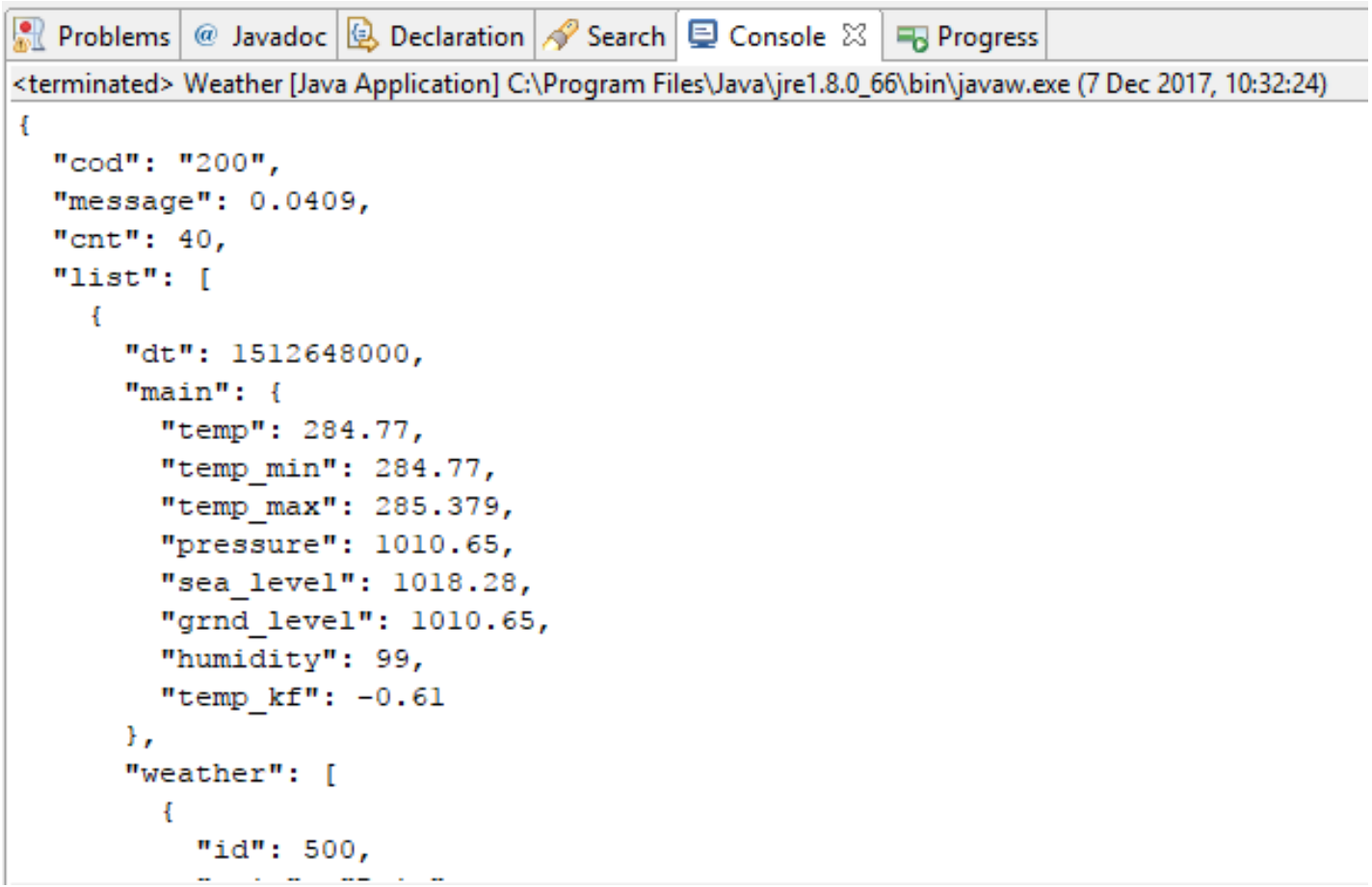
Refer to the file 3_2.txt

Building the backend interface

- We will now add the necessary code to get the actual data from OpenWeatherMaps service:

Refer to the file 3_3.txt




Problems
Javadoc
Declaration
Search
Console
Progress
<terminated> Weather [Java Application] C:\Program Files\Java\jre1.8.0_66\bin\javaw.exe (7 Dec 2017, 10:32:24)
{
"cod": "200",
"message": 0.0409,
"cnt": 40,
"list": [
{
"dt": 1512648000,
"main": {
"temp": 284.77,
"temp_min": 284.77,
"temp_max": 285.379,
"pressure": 1010.65,
"sea_level": 1018.28,
"grnd_level": 1010.65,
"humidity": 99,
"temp_kf": -0.61
},
"weather": [
{
"id": 500,

Building the backend interface

- Let's now implement two methods, getWeatherAtTime() and getWeatherReport(), to generate a short weather report:

Refer to the file 3_4.txt

Building the backend interface

- Calling the getWeatherReport() method from main with the name of the city and time as parameters will result in a short textual weather report like the following one:

Temperature is 297.8 degrees . clear sky.

Implementing the chatbot

Let's start by implementing the Chatbot.java class. We will begin by working out an algorithm to process and respond to users' utterances:

- Process user input.
- Update context.
- Identify bot intent.
- Generate bot utterance and output structure.
- Respond.

Implementing the chatbot

- Let us start by implementing the basic structure based on the mentioned algorithm:

Refer to the file 3_5.txt

Implementing the chatbot

- We will now modify the main() method to simulate a chat window where the user can type in their requests and responses and have a chat with the bot:

Refer to the file 3_6.txt

Implementing the chatbot

Let us first figure out an initial list of user intents for the tasks we have based on the example conversations that we have created:

- greet
- request_current_weather
- inform_city
- thank

Implementing the chatbot

- The request_current_weather intent is used to represent the utterances where the user is requesting current weather info and the inform_city intent is where they mention the name of the city:

Refer to the file 3_7.txt

Implementing the chatbot

- Intent and its associated parameters are boxed up as a JSON object userAction.

- Let us move on to updating the context:

Refer to the file 3_8.txt

Implementing the chatbot

- informWeather is used when the bot has successfully retrieved a report from the backend service that we have built in the previous section:

Refer to the file 3_9.txt

Implementing the chatbot

- When both the time and place are known, it fetches the report and updates the context.

- Next, the bot's intent needs to be translated into an utterance:

Refer to the file 3_10.txt

Implementing the chatbot

- On the console, you will be prompted to start the conversation.

- Have a look at the following example conversation:

Refer to the file 3_11.txt

5. Build Chatbot using NLTK & Keras

TriveraTech
TECHNOLOGY TRAINING

What is Chatbot?

- A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human.
- Chatbots are used a lot in customer interaction, marketing on social network sites and instantly messaging the client.
- There are two basic types of chatbot models based on how they are built; Retrieval based and Generative based models.

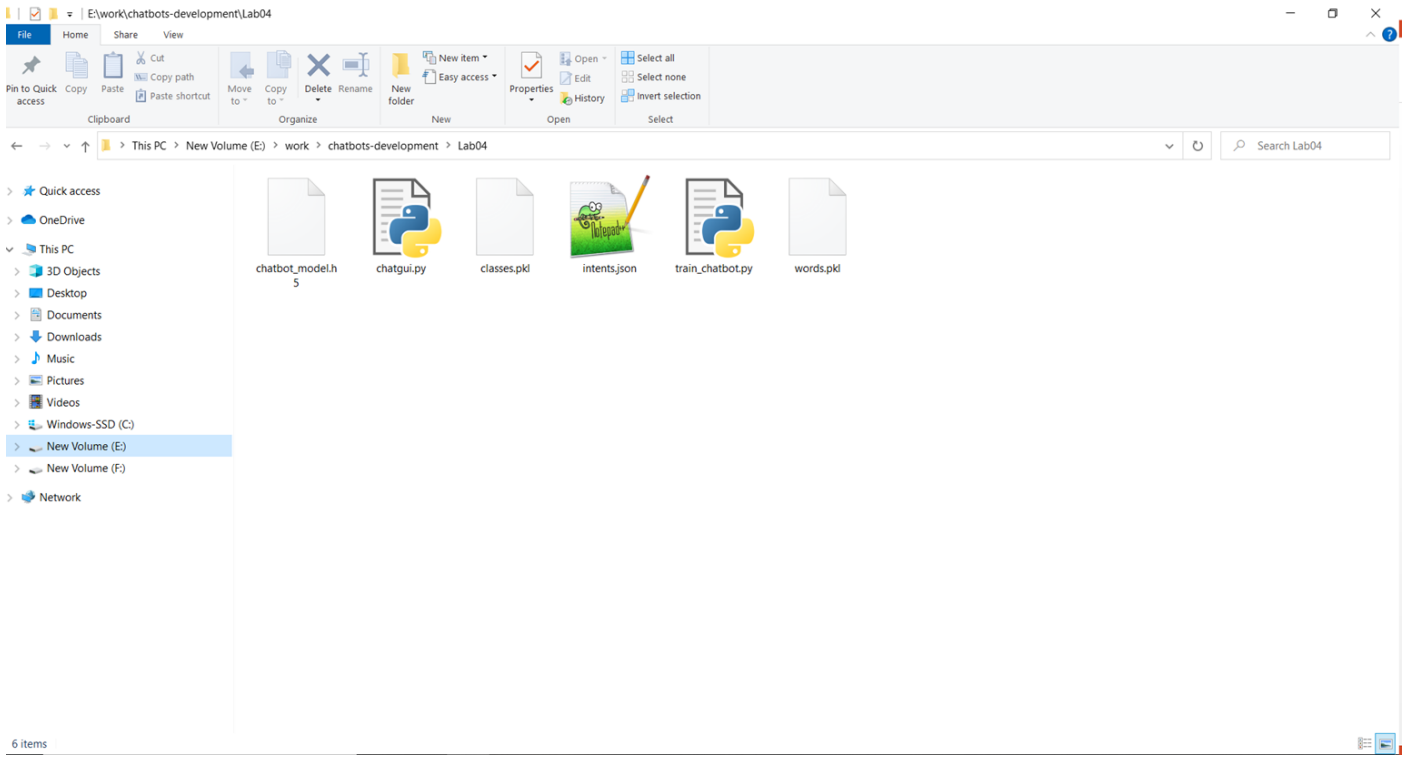
Prerequisites

- The project requires you to have good knowledge of Python, Keras, and Natural language processing (NLTK).

- Along with them, we will use some helping modules which you can download using the python-pip command.

pip install tensorflow, keras, pickle, nltk

How to Make Chatbot in Python?



Import and load the data file

```
1.  import nltk
2.  from nltk.stem import WordNetLemmatizer
3.  lemmatizer = WordNetLemmatizer()
4.  import json
5.  import pickle
6.
7.  import numpy as np
8.  from keras.models import Sequential
9.  from keras.layers import Dense, Activation, Dropout
10. from keras.optimizers import SGD
11. import random
12.
13. words=[]
14. classes = []
15. documents = []
16. ignore_words = ['?', '!']
17. data_file = open('intents.json').read()
18. intents = json.loads(data_file)
```

intents.json - D:\dataflair projects\final chatbot\intents.json (3.6.0)

File Edit Format Run Options Window Help

```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
         "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"
         "context": [""]
        },
        {"tag": "goodbye",
         "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"]
         "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
         "context": [""]
        },
        {"tag": "thanks",
         "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping m
         "responses": ["Happy to help!", "Any time!", "My pleasure"],
         "context": [""]
        },
        {"tag": "noanswer",
         "patterns": [],
         "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understan
         "context": [""]
        },
        {"tag": "options",
         "patterns": ["How you could help me?", "What you can do?", "What help you provide?", "How you
         "responses": ["I can guide you through Adverse drug reaction list, Blood pressure tracking, Ho
         "context": [""]
        },
        {"tag": "adverse_drug",
         "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a li
         "responses": ["Navigating to Adverse drug reaction module"],
```

Ln: 1 Col: 0

Preprocess data

```
for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

Preprocess data

```
1.  # lemmatize, lower each word and remove duplicates
2.  words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
3.  words = sorted(list(set(words)))
4.  # sort classes
5.  classes = sorted(list(set(classes)))
6.  # documents = combination between patterns and intents
7.  print (len(documents), "documents")
8.  # classes = intents
9.  print (len(classes), "classes", classes)
10. # words = all words, vocabulary
11. print (len(words), "unique lemmatized words", words)
12.
13. pickle.dump(words,open('words.pkl','wb'))
14. pickle.dump(classes,open('classes.pkl','wb'))
```

Create training and testing data

```
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
```

Build the model

```
1.  # Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer
    contains number of neurons
2.  # equal to number of intents to predict output intent with softmax
3.  model = Sequential()
4.  model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
5.  model.add(Dropout(0.5))
6.  model.add(Dense(64, activation='relu'))
7.  model.add(Dropout(0.5))
8.  model.add(Dense(len(train_y[0]), activation='softmax'))
9.
10. # Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good
    results for this model
11. sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
12. model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
13.
14. #fitting and saving the model
15. hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
16. model.save('chatbot_model.h5', hist)
17.
18. print("model created")
```

Predict the response (Graphical User Interface)

```
1.  import nltk
2.  from nltk.stem import WordNetLemmatizer
3.  lemmatizer = WordNetLemmatizer()
4.  import pickle
5.  import numpy as np
6.
7.  from keras.models import load_model
8.  model = load_model('chatbot_model.h5')
9.  import json
10. import random
11. intents = json.loads(open('intents.json').read())
12. words = pickle.load(open('words.pkl','rb'))
13. classes = pickle.load(open('classes.pkl','rb'))
```

```
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list
```

Predict the response (Graphical User Interface)

```
1.  def getResponse(ints, intents_json):
2.      tag = ints[0]['intent']
3.      list_of_intents = intents_json['intents']
4.      for i in list_of_intents:
5.          if(i['tag']== tag):
6.              result = random.choice(i['responses'])
7.              break
8.      return result
9.
10. def chatbot_response(text):
11.     ints = predict_class(text, model)
12.     res = getResponse(ints, intents)
13.     return res
```

```
#Creating GUI with tkinter
import tkinter
from tkinter import *


def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)

ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set


#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
                    bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
                    command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")
#EntryBox.bind("<Return>", send)


#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)

base.mainloop()
```

Run the chatbot

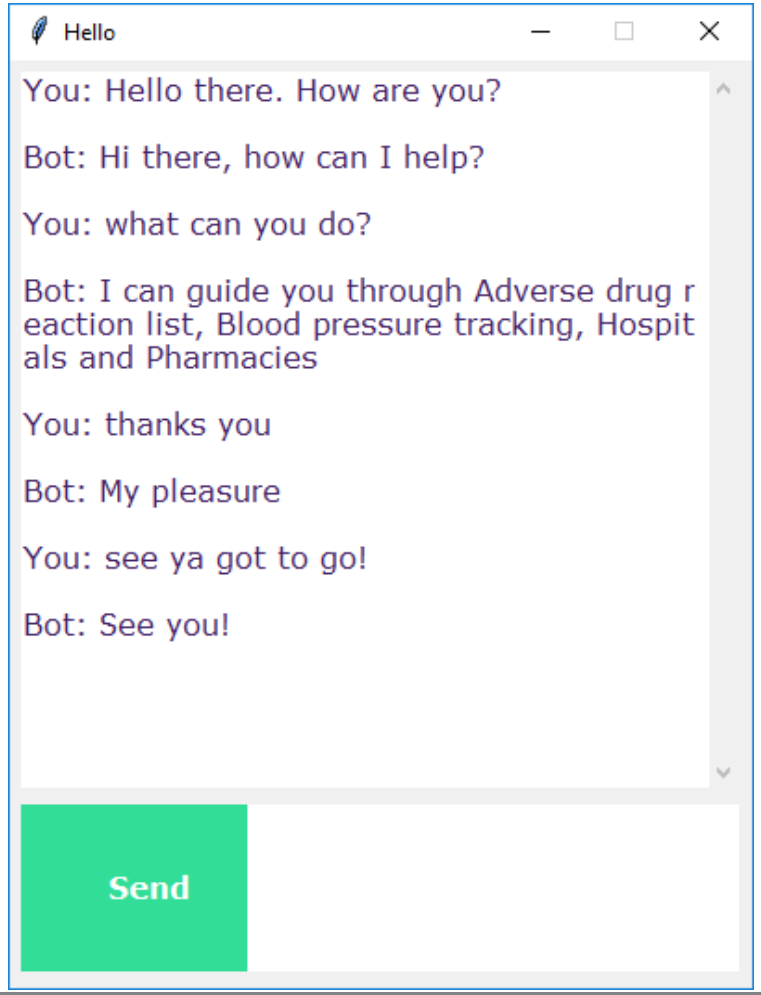
- To run the chatbot, we have two main files; train_chatbot.py and chatapp.py.
- First, we train the model using the command in the terminal:

```
python train_chatbot.py
```

- If we don’t see any error during training, we have successfully created the model.
- Then to run the app, we run the second file.

```
python chatgui.py
```

```
Using TensorFlow backend.
47 documents
9 classes ['adverse_drug', 'blood_pressure', 'blood_pressure_search', 'goodbye', 'greeting',
 'hospital_search', 'options', 'pharmacy_search', 'thanks']
88 unique lemmatized words ["'s", ',', 'a', 'adverse', 'all', 'anyone', 'are', 'awesome', 'b
e', 'behavior', 'blood', 'by', 'bye', 'can', 'causing', 'chatting', 'check', 'could', 'data'
, 'day', 'detail', 'do', 'dont', 'drug', 'entry', 'find', 'for', 'give', 'good', 'goodbye',
'have', 'hello', 'help', 'helpful', 'helping', 'hey', 'hi', 'history', 'hola', 'hospital', '
how', 'i', 'id', 'is', 'later', 'list', 'load', 'locate', 'log', 'looking', 'lookup', 'manag
ement', 'me', 'module', 'nearby', 'next', 'nice', 'of', 'offered', 'open', 'patient', 'pharm
acy', 'pressure', 'provide', 'reaction', 'related', 'result', 'search', 'searching', 'see',
'show', 'suitable', 'support', 'task', 'thank', 'thanks', 'that', 'there', 'till', 'time', '
to', 'transfer', 'up', 'want', 'what', 'which', 'with', 'you']
Training data created
2019-11-28 14:10:10.207987: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU su
pports instructions that this TensorFlow binary was not compiled to use: AVX2
Epoch 1/200
47/47 [==============================] - 0s 2ms/step - loss: 2.2080 - accuracy: 0.1489
Epoch 2/200
47/47 [==============================] - 0s 211us/step - loss: 2.1478 - accuracy: 0.1277
Epoch 3/200
47/47 [==============================] - 0s 228us/step - loss: 2.1427 - accuracy: 0.1277
```


Hello
You: Hello there. How are you?
Bot: Hi there, how can I help?
You: what can you do?
Bot: I can guide you through Adverse drug r eaction list, Blood pressure tracking, Hospit als and Pharmacies
You: thanks you
Bot: My pleasure
You: see ya got to go!
Bot: See you!
Send

Summary

- In this Python data science project, we understood about chatbots and implemented a deep learning version of a chatbot in Python which is accurate.
- You can customize the data according to business requirements and train the chatbot with great accuracy.
- Chatbots are used everywhere and all businesses are looking forward to implementing bot in their workflow.

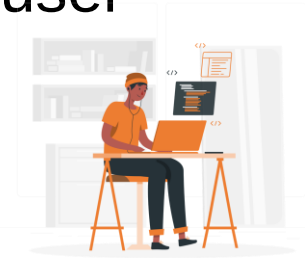
6. Let's Catch a Train

TriveraTech
TECHNOLOGY TRAINING

Let's Catch a Train

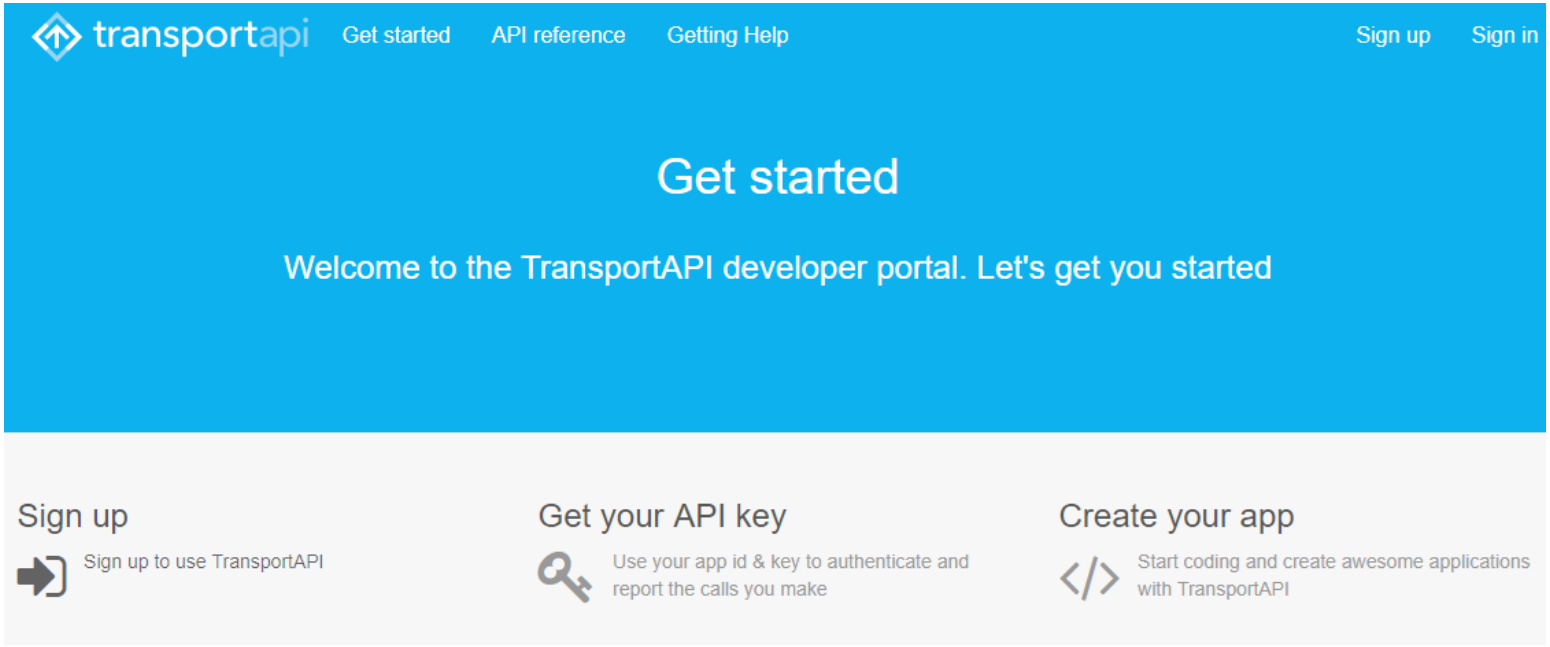
By the end of this lesson, you will be able to:

- Design conversational tasks based on data
- Create backend task modules using Transport API
- Build SMS bots using Twilio
- Integrate a Dialogflow agent to understand user utterances



Exploring Transport API

Creating a developer account



Creating a developer account

- Execute the following GET request from a web browser.
- At this request, we are trying to retrieve information concerning Euston train station, in London:

http://transportapi.com/v3/uk/places.json?query=euston&type=train_station&app_id=YOUR_APP_ID&app_key=YOUR_APP_KEY

Raw | Parsed


```
{
  "request_time": "2017-08-05T07:06:08+01:00",
  "source": "Network Rail",
  "acknowledgements": "Contains information of Network Rail Infrastructure Limited. License
  http://www.networkrail.co.uk/data-feeds/terms-and-conditions/",
  "member": [
    {
      "type": "train_station",
      "name": "Edinburgh Gateway",
      "latitude": 55.940938,
      "longitude": -3.320251,
      "accuracy": 100,
      "station_code": "EGY",
      "tiploc_code": "EDINGWY"
    },
    {
      "type": "train_station",
      "name": "Edinburgh",
      "latitude": 55.952391,
      "longitude": -3.188228,
      "accuracy": 100,
      "station_code": "EDB",
      "tiploc_code": "EDINBUR"
    },
    {
      "type": "train_station",
      "name": "Edinburgh Park",
      "latitude": 55.927549,
      "longitude": -3.307664,
      "accuracy": 100,
      "station_code": "EDP",
      "tiploc_code": "EDINPRK"
    }
  ]
}
```


Exploring the dataset

Train stations near you

- This endpoint provides a list of train stations near a given location.
- The search location should be provided as a latLon coordinate, as follows:

http://transportapi.com/v3/uk/train/stations/near.json?lat=55.9485&lon=-3.2021&app_id=YOUR_APP_ID&app_key=YOUR_APP_KEY

Exploring the dataset


```
{
    "minlon": -3.4521,
    "minlat": 55.6985,
    "maxlon": -2.9521,
    "maxlat": 56.1985,
    "searchlon": -3.2021,
    "searchlat": 55.9485,
    "page": 1,
    "rpp": 25,
    "total": 54,
    "request_time": "2017-08-05T06:21:24+01:00",
    "stations": [ … ] // 25 items
}
```

```
"stations": [
    {
        "station_code": "EDB",
        "atcocode": null,
        "tiploc_code": "EDINBUR",
        "name": "Edinburgh Waverley",
        "mode": "train",
        "longitude": -3.188228,
        "latitude": 55.952391,
        "distance": 966
    },
    { … }, // 8 items
    { … }, // 8 items
    { … }, // 8 items
```

Exploring the dataset

Trains in the area

- Let's try this using the bounding box coordinates for Edinburgh:

http://transportapi.com/v3/uk/train/stations/bbox.json?minlon=-3.4521&minlat=55.6985&maxlon=-2.9521&maxlat=56.1985&app_id=YOUR_APP_ID&app_key=YOUR_APP_KEY

Response


```
{
    "minlon": -3.4521,
    "minlat": 55.6985,
    "maxlon": -2.9521,
    "maxlat": 56.1985,
    "searchlon": -3.2021,
    "searchlat": 55.9485,
    "page": 1,
    "rpp": 25,
    "total": 54,
    "request_time": "2017-08-05T06:58:20+01:00",
    "stations": [
        {
            "station_code": "EDB",
            "atcocode": null,
            "tiploc_code": "EDINBUR",
            "name": "Edinburgh Waverley",
            "mode": "train",
            "longitude": -3.188228,
            "latitude": 55.952391,
            "distance": 966
        },
        {
            "station_code": "HYM",
            "atcocode": null,
            "tiploc_code": "HAYMRKT",
            "name": "Haymarket",
            "mode": "train",
            "longitude": -3.218449,
            "latitude": 55.945806,
            "distance": 1061
        },
```


Live departures

- The live status of trains arriving and departing a given station can be obtained using the following endpoint.

- Let's try it out for Edinburgh Waverley station, whose station code is EDB:

http://transportapi.com/v3/uk/train/station/EDB/live.json?&app_id=YOUR_APP_ID&app_key=YOUR_APP_KEY

Response

Raw | Parsed


```
{
  "date": "2017-08-05",
  "time_of_day": "06:00",
  "request_time": "2017-08-05T06:49:44+01:00",
  "station_name": "Edinburgh Waverley",
  "station_code": "EDB",
  "departures": {
    "all": [
      {
        "mode": "train",
        "service": "13560015",
        "train_uid": "G83644",
        "platform": "10",
        "operator": "SR",
        "operator_name": "Scotrail",
        "aimed_departure_time": "06:07",
        "aimed_arrival_time": null,
        "aimed_pass_time": null,
        "origin_name": "Edinburgh Waverley",
        "source": "ATOC",
        "destination_name": "Milngavie",
        "category": "OO",
        "service_timetable": {
          "id": "http://transportapi.com/v3/uk/train/service/train_uid:G83644/2017-08-
          05/timetable.json?app_id=60ce46ea&app_key=251dc61414961e8ebfe110329ffa367d"
        }
      },
      {
        "mode": "train",
        "service": "22180008",
        "train_uid": "P31962",
        "platform": "7",
        "operator": "XC",
        "operator_name": "CrossCountry",
```

Station timetables

- Timetables of trains arriving and departing from a given station on a given date and time can be obtained using the following endpoint.
- Let's get all the trains departing Edinburgh Waverley (EDB) station on 2017-08-05 at 06:00:

http://transportapi.com/v3/uk/train/station/EDB/2017-08-05/06:00/timetable.json?app_id=YOUR_APP_ID&app_key=YOUR_APP_KEY

Response

Raw | Parsed


```
{
  "date": "2017-08-05",
  "time_of_day": "06:00",
  "request_time": "2017-08-05T06:49:44+01:00",
  "station_name": "Edinburgh Waverley",
  "station_code": "EDB",
  "departures": {
    "all": [
      {
        "mode": "train",
        "service": "13560015",
        "train_uid": "G83644",
        "platform": "10",
        "operator": "SR",
        "operator_name": "Scotrail",
        "aimed_departure_time": "06:07",
        "aimed_arrival_time": null,
        "aimed_pass_time": null,
        "origin_name": "Edinburgh Waverley",
        "source": "ATOC",
        "destination_name": "Milngavie",
        "category": "OO",
        "service_timetable": {
          "id": "http://transportapi.com/v3/uk/train/service/train_uid:G83644/2017-08-
          05/timetable.json?app_id=60ce46ea&app_key=251dc61414961e8ebfe110329ffa367d"
        }
      },
      {
        "mode": "train",
        "service": "22180008",
        "train_uid": "P31962",
        "platform": "7",
        "operator": "XC",
        "operator_name": "CrossCountry",
```

Service timetables

- Let's try an example out with service number, 23587103, on 2017-08-05 at 06:00:

http://transportapi.com/v3/uk/train/service/23587103/2017-08-05/06:00/timetable.json?app_id=YOUR_APP_ID&app_key=YOUR_APP_KEY

Response


```
{
  "service": "23587103",
  "train_uid": "G82394",
  "headcode": "",
  "toc": {
      "atoc_code": "SR"
  },
  "train_status": "P",
  "origin_name": "Markinch",
  "destination_name": "Edinburgh Waverley",
  "stop_of_interest": null,
  "date": "2017-08-05",
  "time_of_day": "06:00",
  "mode": "train",
  "request_time": "2017-08-05T06:46:22+01:00",
  "category": "OO",
  "operator": "SR",
  "operator_name": "Scotrail",
  "stops": [
    {
        "station_code": "MNC",
        "tiploc_code": "MKIN",
        "station_name": "Markinch",
        "stop_type": "LO",
        "platform": "1",
        "aimed_departure_date": "2017-08-05",
        "aimed_departure_time": "06:19",
        "aimed_arrival_date": null,
        "aimed_arrival_time": null,
        "aimed_pass_date": null,
        "aimed_pass_time": null
    },
    {
        "station_code": "GLT",
        "tiploc_code": "GLNRTHS",
        "station_name": "Glenrothes With Thornton",
```

Building a simple SMS bot

- Let's now build the SMS platform interface for the chatbot.
- To bear with the complexity, let us do this in two steps.
- First, let us build a bot to simply send SMS text messages to a mobile number.
- This could be a message concerning the status of a train arriving at a station or the next train to a certain destination from a given station.

Setting up the dashboard

Programmable SMS

- Dashboard
- Messaging Services
- Short Codes
- Tools
- Logs
- Insights
- Add-ons *Beta*
- Usage
- Settings
- Getting Started

Home / SMS /

Show API Credentials >

Programmable SMS Dashboard

Programmatically send and receive SMS worldwide. Route text messages globally to and from your application over local, mobile, toll-free, and short code numbers.

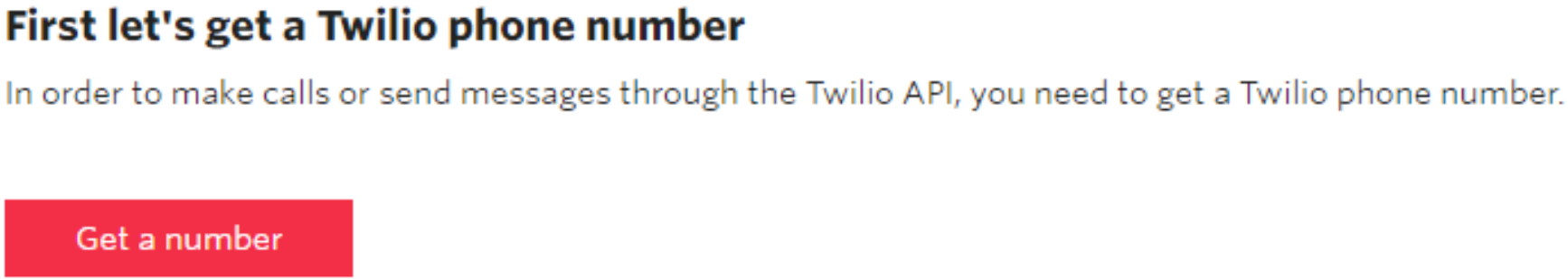
Get Started | Read the Tutorial Docs ↗ | Features & Pricing

You have a Trial Account

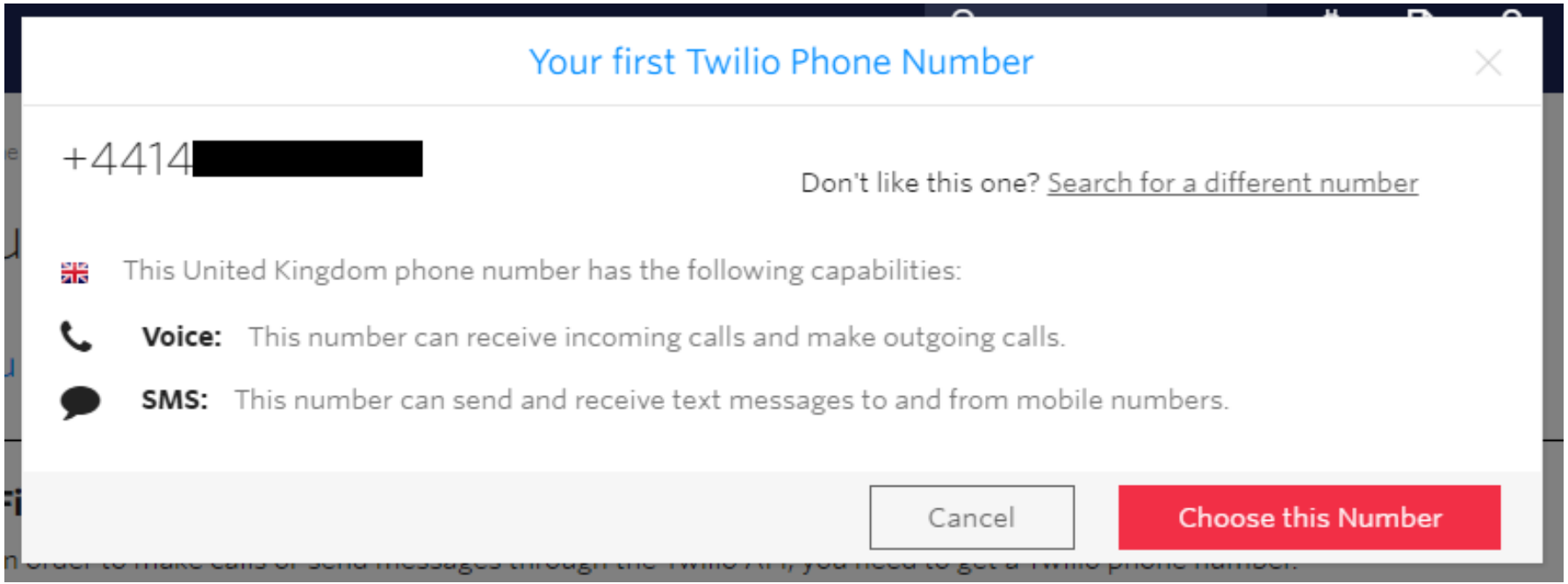
- Your trial account has $15.50 remaining
- Trial accounts can only send messages to verified numbers in these countries
- Messages sent in trial will begin with "Sent from a Twilio Trial Account"
- While you have a trial account, you're limited to one Twilio number

Setting up the dashboard

- In order to send SMS messages, you need a phone number.
- Click Get a number to get one:

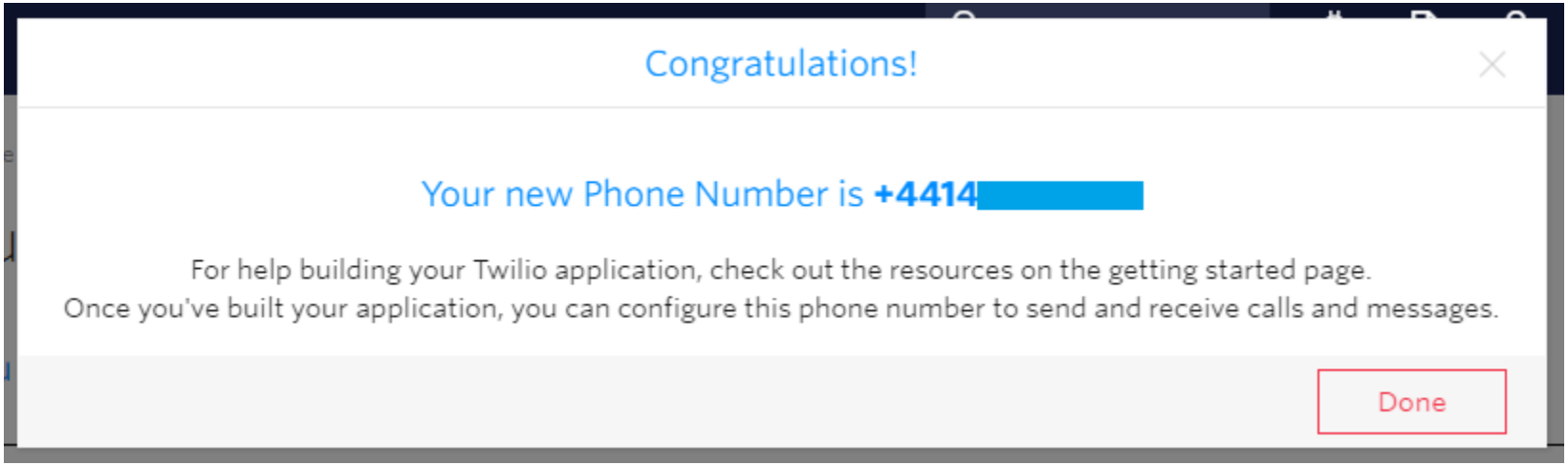


Setting up the dashboard

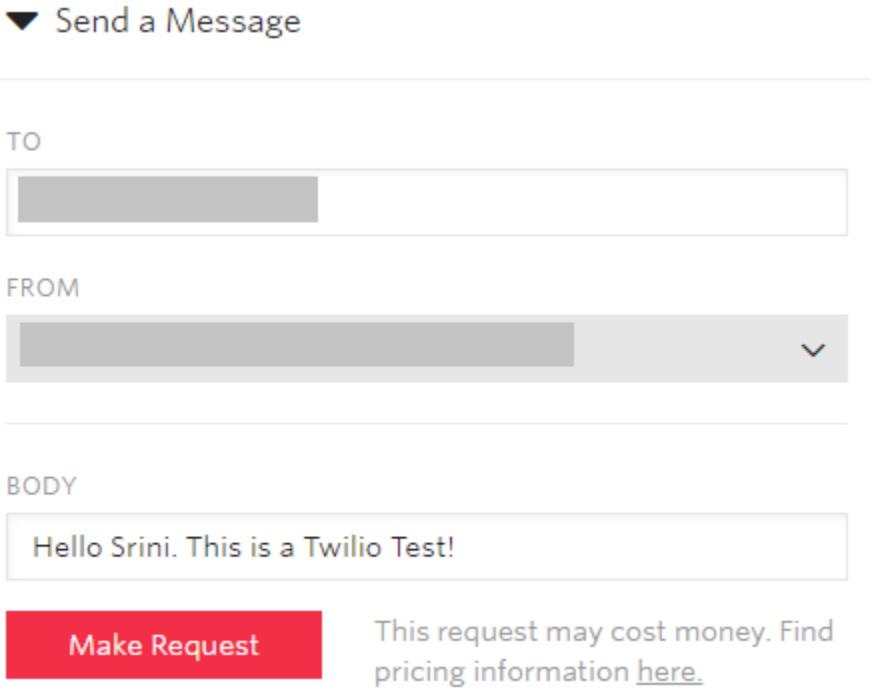


Setting up the dashboard

- You will receive an acknowledgment that you have been allocated the number:

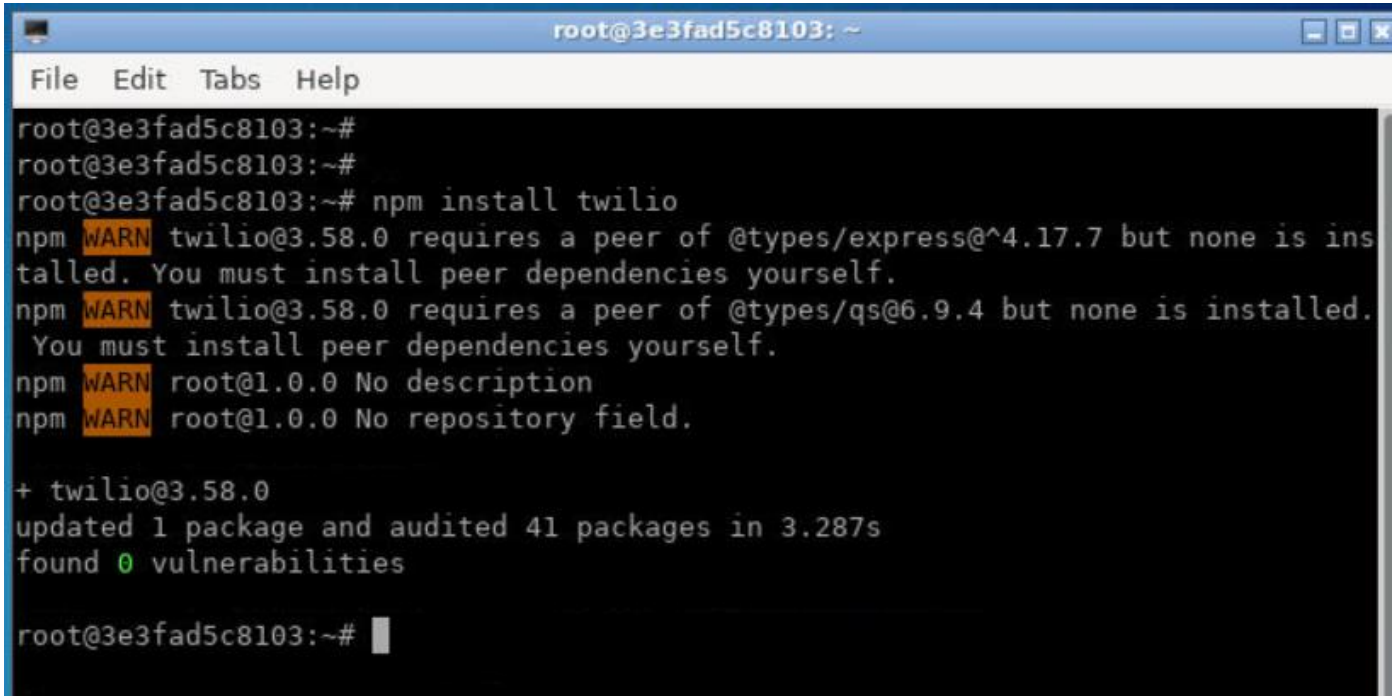


Setting up the dashboard



```
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (twitterbot)
version: (1.0.0)
description: my news bot
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to  package.json:

{
  "name": "twitterbot",
  "version": "1.0.0",
  "description": "my news bot",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Simple Message Sender



Simple Message Sender

- Let's create a new JS file called index.js. Add the following code to the file:

Refer to the file 5_1.txt

My train notifier

- Install the request library using npm install request --save.
- Create a function to send SMS notifications:

Refer to the file 5_2.txt

My train notifier

- Create a function to get all trains departing from a given station:

Refer to the file 5_3.txt

- Create a function to retrieve all trains going to a certain destination station:

Refer to the file 5_4.txt

My train notifier

- Create a function to call the preceding functions to send the user a notification of all trains heading to a certain destination station from the user's preferred station:

Refer to the file 5_5.txt

- And set the variables and call the main module:

Refer to the file 5_6.txt

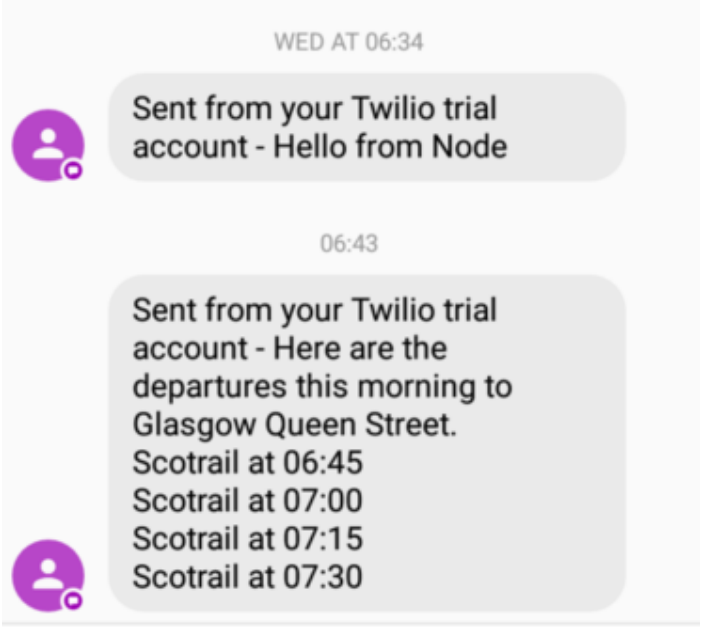
My train notifier

"node index.js" Output:

```
Summary: Here are the departures this morning to Glasgow Queen Street.
Scotrail at 06:45
Scotrail at 07:00
Scotrail at 07:15
Scotrail at 07:30

SM79cb0e9ed7e5499f99ec35b3c2f77681

C:\Users\Srini\Documents\workspace\SMSBot>
```

My train notifier

- Check your cellphone to see whether the message has arrived:



Summary

- Brilliant! I hope you had a great time exploring and building a transport chatbot providing useful information on trains to users.

- Besides building the chatbot, we also explored the use of Twilio communication APIs to expose the chatbot over the SMS platform.

7. Restaurant Search

TriveraTech
TECHNOLOGY TRAINING

Restaurant Search

By the end of this lesson, you will be able to:

- Understand the basics of MS Bot Framework
- Build a chatbot with the Botbuilder Node.js library
- Register the bot with Bot Framework
- Host the bot in the cloud
- Understand message types and card types
- Manage context and conversational flow
- Integrate with the Zomoto data API
- Integrate the bot with Skype

MS Bot Framework

- MS Bot Framework is a Microsoft product for chatbot development.
- It houses three products: Bot Builder SDK, Bot Framework Portal, and channels.
- Bot Builder SDK is the toolkit for building chatbots.
- It has libraries of classes and code that represent various elements of a conversation.

Channel emulator

- Before we begin, we need to install software called a channel emulator.
- We will be using this to emulate the channel to connect to the bot locally for development and testing purposes.
- You can chat with your bot as well as inspect the messages sent and received to identify any bugs.

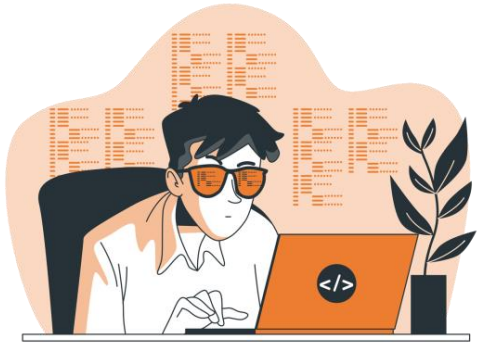
Building a bot

- Create a Node.js project called foodie-bot:

```
> npm init
```

- Install the two libraries that we need to use:

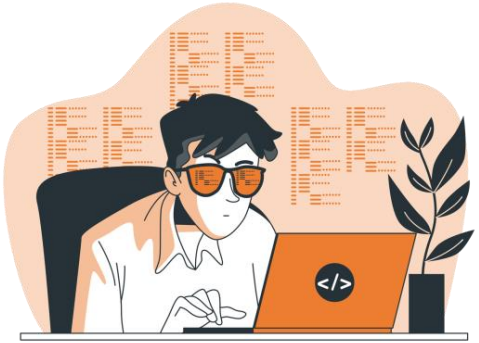
```
> npm install botbuilder --save
> npm install restify --save
```



Building a bot

- Create a file named app.js.
- In app.js, paste the following code (from the Bot Framework tutorials):

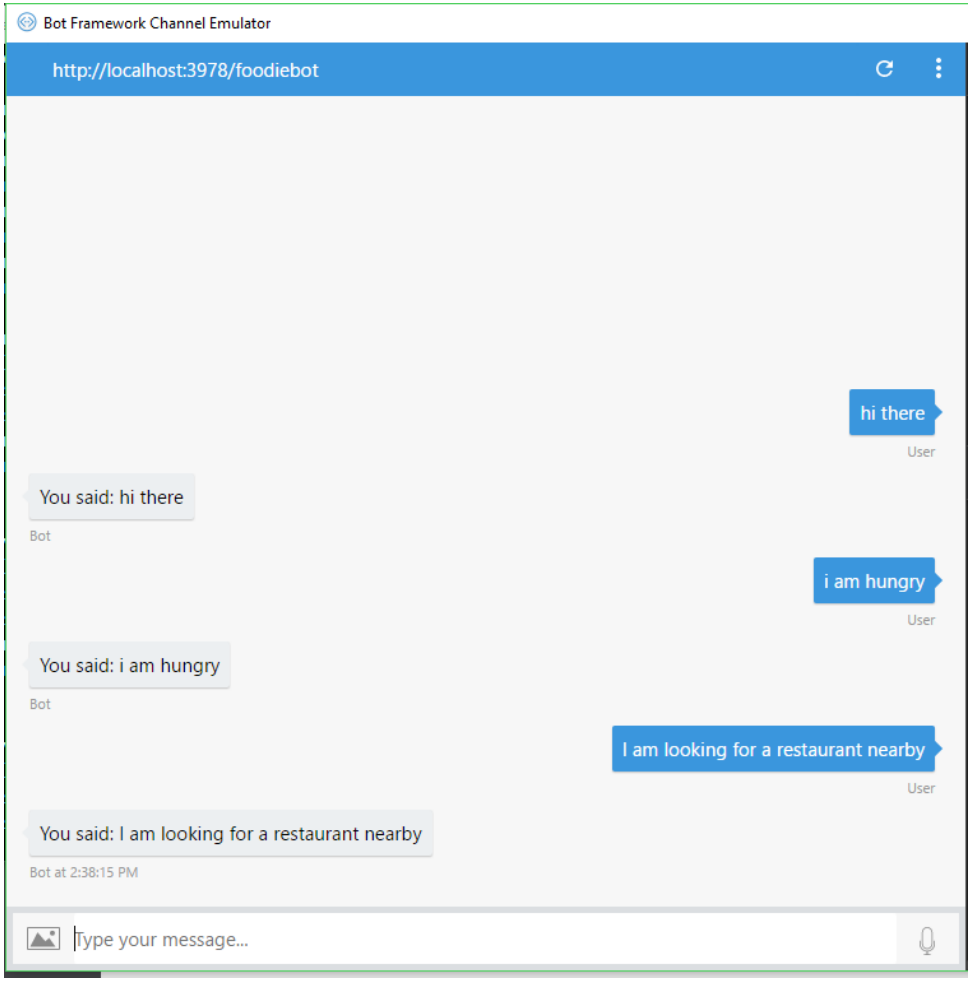
Refer to the file 6_1.txt

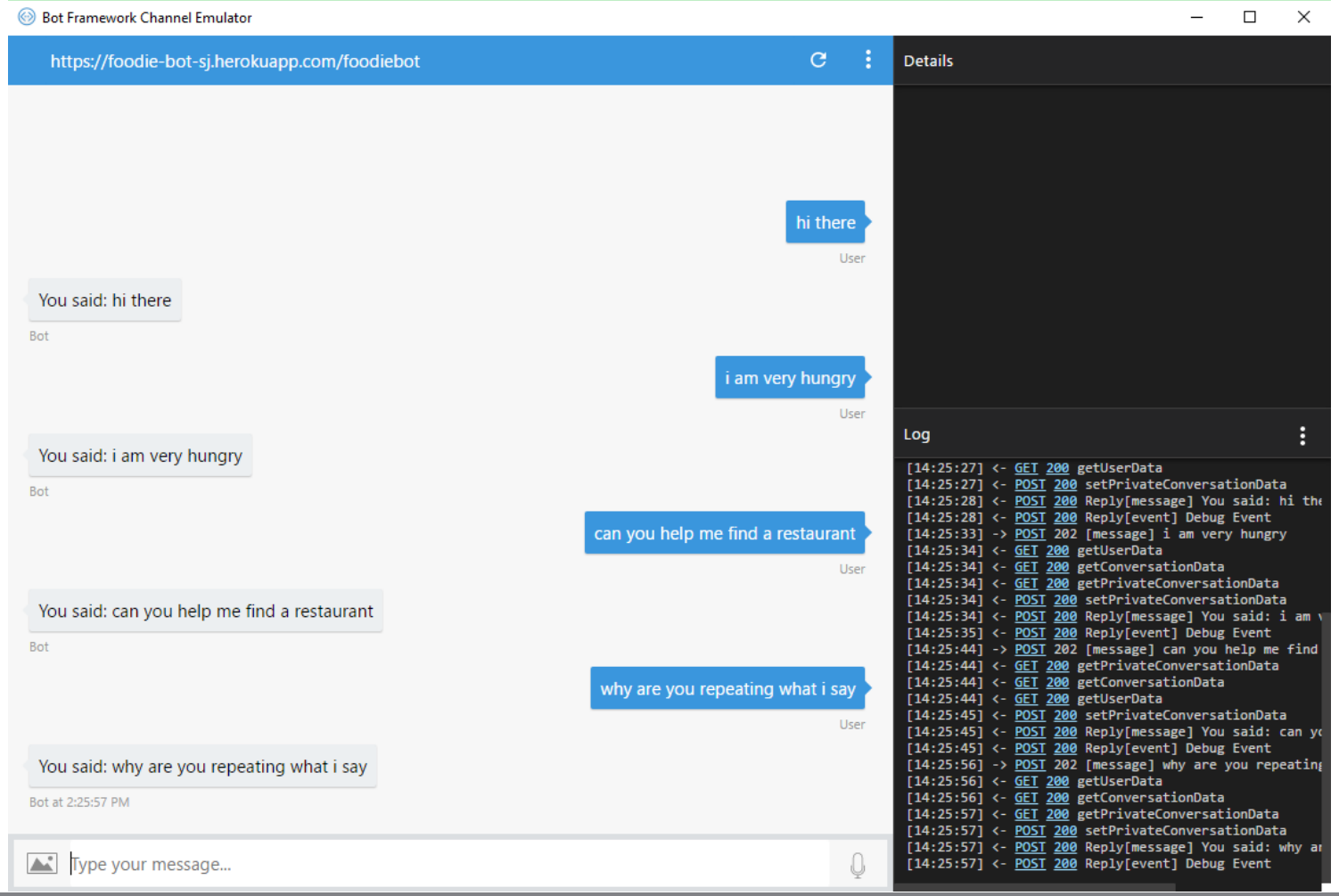


Building a bot

- The emulator will connect to the bot (running in app.js).
- The app will start logging messages on the console, as shown here: “node app.js”

```
restify listening to http://[::]:3978
WARN: ChatConnector: receive - emulator running without security enabled.
ChatConnector: message received.
WARN: ChatConnector: receive - emulator running without security enabled.
ChatConnector: message received.
WARN: ChatConnector: receive - emulator running without security enabled.
```


Bot Framework Channel Emulator
http://localhost:3978/foodiebot
hi there
User
You said: hi there
Bot
i am hungry
User
You said: i am hungry
Bot
I am looking for a restaurant nearby
User
You said: I am looking for a restaurant nearby
Bot at 2:38:15 PM
Type your message...

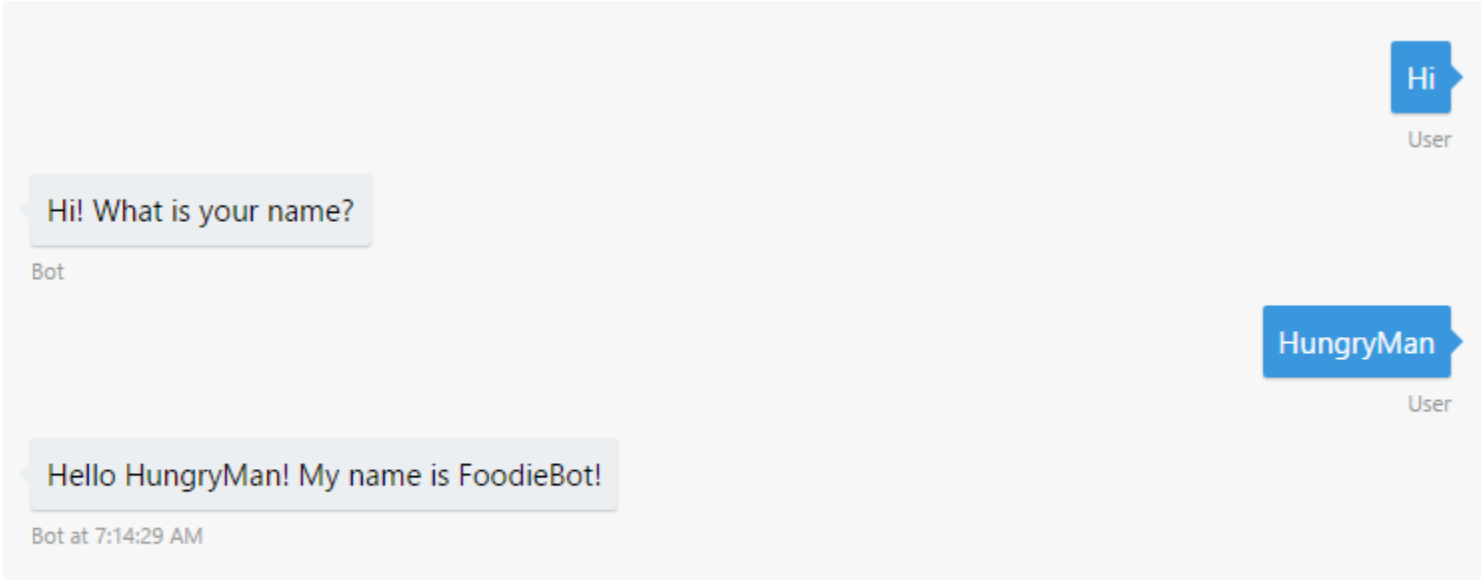

Bot Framework Channel Emulator
https://foodie-bot-sj.herokuapp.com/foodiebot
hi there
User
You said: hi there
Bot
i am very hungry
User
You said: i am very hungry
Bot
can you help me find a restaurant
User
You said: can you help me find a restaurant
Bot
why are you repeating what i say
User
You said: why are you repeating what i say
Bot at 2:25:57 PM
Type your message...
Details
Log
[14:25:27] <- GET 200 getUserData
[14:25:27] <- POST 200 setPrivateConversationData
[14:25:28] <- POST 200 Reply[message] You said: hi the
[14:25:28] <- POST 200 Reply[event] Debug Event
[14:25:33] -> POST 202 [message] i am very hungry
[14:25:34] <- GET 200 getUserData
[14:25:34] <- GET 200 getConversationData
[14:25:34] <- GET 200 getPrivateConversationData
[14:25:34] <- POST 200 setPrivateConversationData
[14:25:34] <- POST 200 Reply[message] You said: i am
[14:25:35] <- POST 200 Reply[event] Debug Event
[14:25:44] -> POST 202 [message] can you help me find
[14:25:44] <- GET 200 getPrivateConversationData
[14:25:44] <- GET 200 getConversationData
[14:25:44] <- GET 200 getUserData
[14:25:45] <- POST 200 setPrivateConversationData
[14:25:45] <- POST 200 Reply[message] You said: can yo
[14:25:45] <- POST 200 Reply[event] Debug Event
[14:25:56] -> POST 202 [message] why are you repeating
[14:25:56] <- GET 200 getUserData
[14:25:56] <- GET 200 getConversationData
[14:25:57] <- GET 200 getPrivateConversationData
[14:25:57] <- POST 200 setPrivateConversationData
[14:25:57] <- POST 200 Reply[message] You said: why ar
[14:25:57] <- POST 200 Reply[event] Debug Event

Sending more than one message per turn

- First, we can send more than one message at a time.
- So when the chatbot gets its turn, it can send multiple messages using the session.send() method:

```
var bot = new builder.UniversalBot(connector, [
    function (session) {
        session.send('Hello there!');
        session.send('Welcome to New India restaurant!');
        });
    }
]);
```

MS Bot

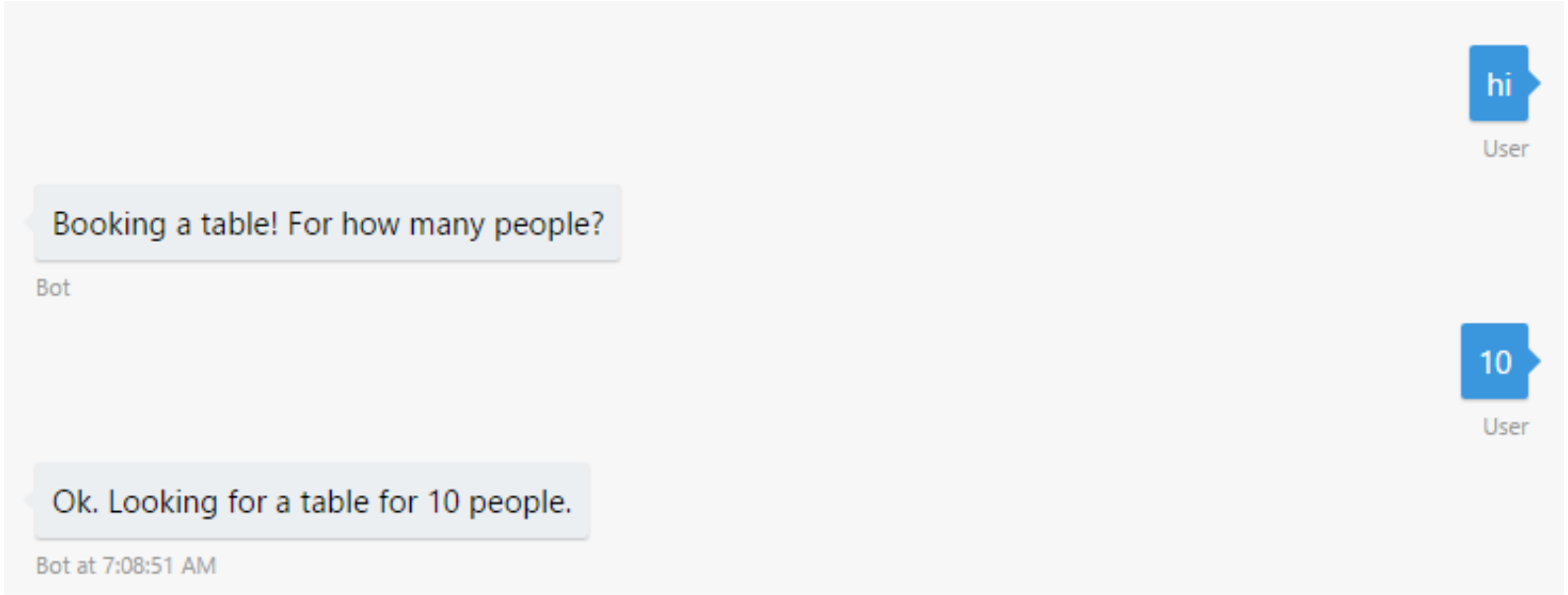

Hi
User
Hi! What is your name?
Bot
HungryMan
User
Hello HungryMan! My name is FoodieBot!
Bot at 7:14:29 AM

MS Bot Numeric Prompt

- You can get numeric data using builder.Prompts.number():

```
var bot = new builder.UniversalBot(connector, [
    function (session) {
        builder.Prompts.number(session, 'courseing a table!
                                For how many people?');
    },
    function (session, results) {
        session.endDialog('Ok. Looking for a table for ' +
                          results.response + ' people.');
    }
]);
```

MS Bot


hi
User
Booking a table! For how many people?
Bot
10
User
Ok. Looking for a table for 10 people.
Bot at 7:08:51 AM

MS Bot

```
var bot = new builder.UniversalBot(connector, [
    function (session) {
        builder.Prompts.choice(session, 'courseing a table!
        Any specific cuisine?', ['Indian', 'Chinese', 'Italian']);
    },
    function (session, results) {
        session.endDialog('Ok. Looking for a ' +
        results.response.entity + ' restaurant.');
    }
]);
```

hello

User

Booking a table! Any specific cuisine? (1. Indian, 2. Chinese, or 3. Italian)

Bot

Indian

User

Ok. Looking for a Indian restaurant.

Bot

hello

User

Booking a table! Any specific cuisine? (1. Indian, 2. Chinese, or 3. Italian)

Bot

2

User

Ok. Looking for a Chinese restaurant.

Bot at 7:05:50 AM

MS Bot Choice

- You can also provide choices in the following format, instead of an array, as shown here:

```
builder.Prompts.choice(session, 'courseing a table! Any specific cuisine?', 'Indian|Chinese|Italian');
```

MS Bot DateTime

- You can also prompt for date and time and parse varied inputs such as tomorrow at 2pm, Saturday at 8, or next Friday using the EntityRecognizer class, as follows:

```
builder.Prompts.time(session, "So when is the party?");
....
session.dialogData.partyDate =
builder.EntityRecognizer.resolveTime([results.response]);
```

Rich messages

- Now that we know how to serve messages and prompts, let's dig a little deeper to learn how to make it look more visually appealing by adding images and cards.
- To do this, we will use Hero card & Hero card is a template for presenting information in a rich format using images, URLs, and so on. Here is an example:

Refer to the file 6_2.txt

Show me Indian restaurants

User

Chennai Kitchen

Authentic South Indian Restaurant

Great tasting dosas. 5 star reviews.

Book a table

Mumbai Tandoor

Best Indian Restaurant in town

Amazing reviews!

Book a table

Bot at 7:49:18 AM

Rich messages

- However, you can also program it to open a web page using the openUrl() method, as follows:

```
builder.CardAction.openUrl(session,
'https://mumbaitandoor.com/courseTable','course a
table');
```

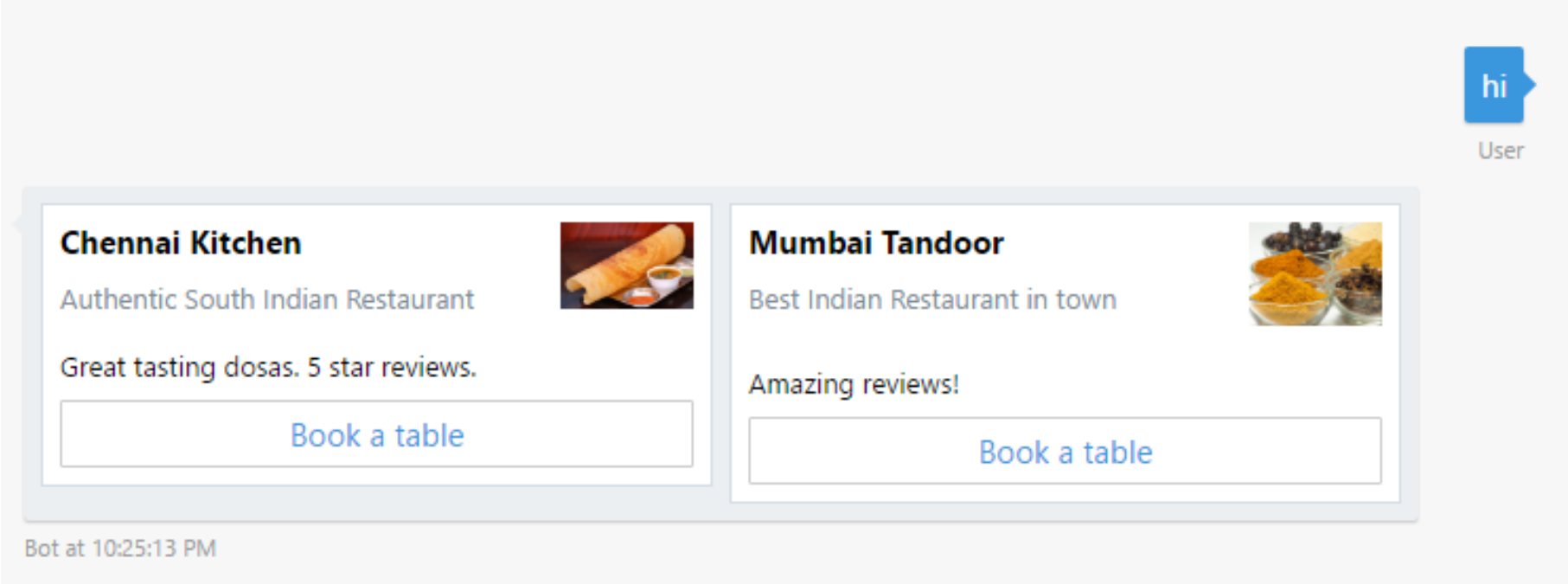
Rich messages

- Thumbnail cards are similar to Hero cards but smaller.

- You can create Thumbnail cards using the ThumbnailCard class, as shown here:

Refer to the file 6_3.txt

Rich messages



Rich messages

```
newbuilder.AnimationCard(session)
        .title('Microsoft Bot Framework')
        .subtitle('Animation Card')
        .image(builder.CardImage.create(session,
        'https://makeYourOwnCurry.com/curryAnimation.jpeg'))
        .media([
            { url:'http://i.giphy.com/Ki55RUbOV5njy.gif' }
        ])
```

Rich messages

- Audio and Video cards can be used to present audio and video information:

Refer to the file 6_4.txt

Refer to the file 6_5.txt

How is chicken tikka made

User

8:55

Chicken Tikka

from Sanjeev Kapoor Khazana

Authentic Chicken Tikka recipe by Chef Harpal Singh Sokhi

Order Chicken Tikka

Bot at 10:54:27 PM

Rich messages

- In addition to these cards, there is a special card called the Receipt card which will present information in a receipt format.

- It can be used to present an itemized bill with payment information, as follows:

Refer to the file 6_6.txt

Can i have my receipt please

User

James White

Order Number	12345
Payment Method	VISA 2392-****
Chicken Tikka	£ 6.50
Garlic Naan	£ 5.00
Tax	£ 1.15
Total	**£ 12.65**

Send by email

Bot at 11:13:10 PM

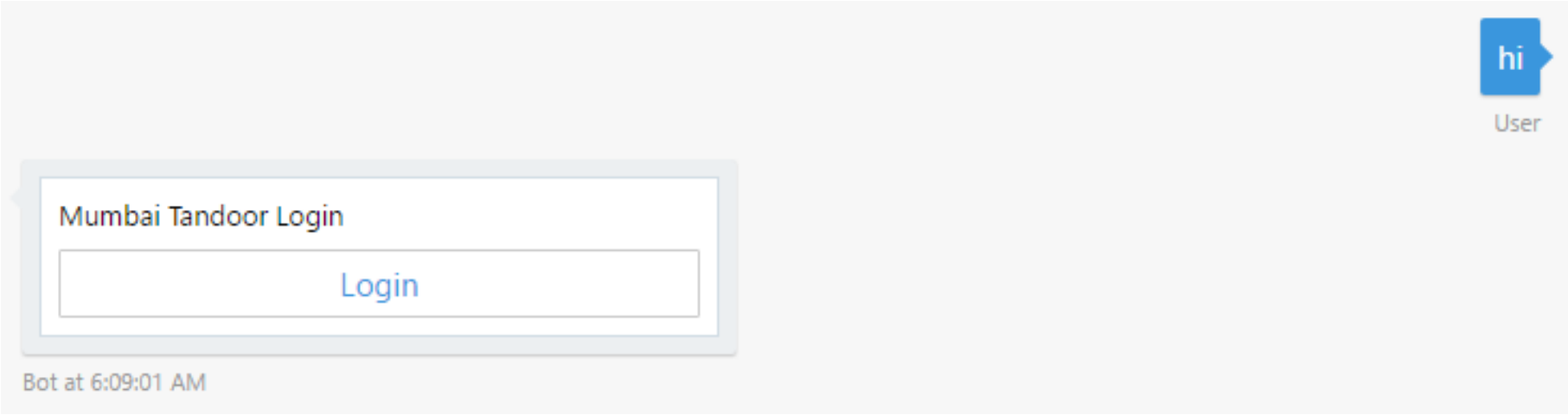
Rich messages

- Finally, there is a card that can be used to authenticate the user by asking them to sign in.
- This flow can be initiated using the SignIn card:

```
new builder.SigninCard(session)
        .text('Mumbai Tandoor Login')
        .button('Login', 'https://mumbaitandoor.com/login')
        ]);
```

Rich messages

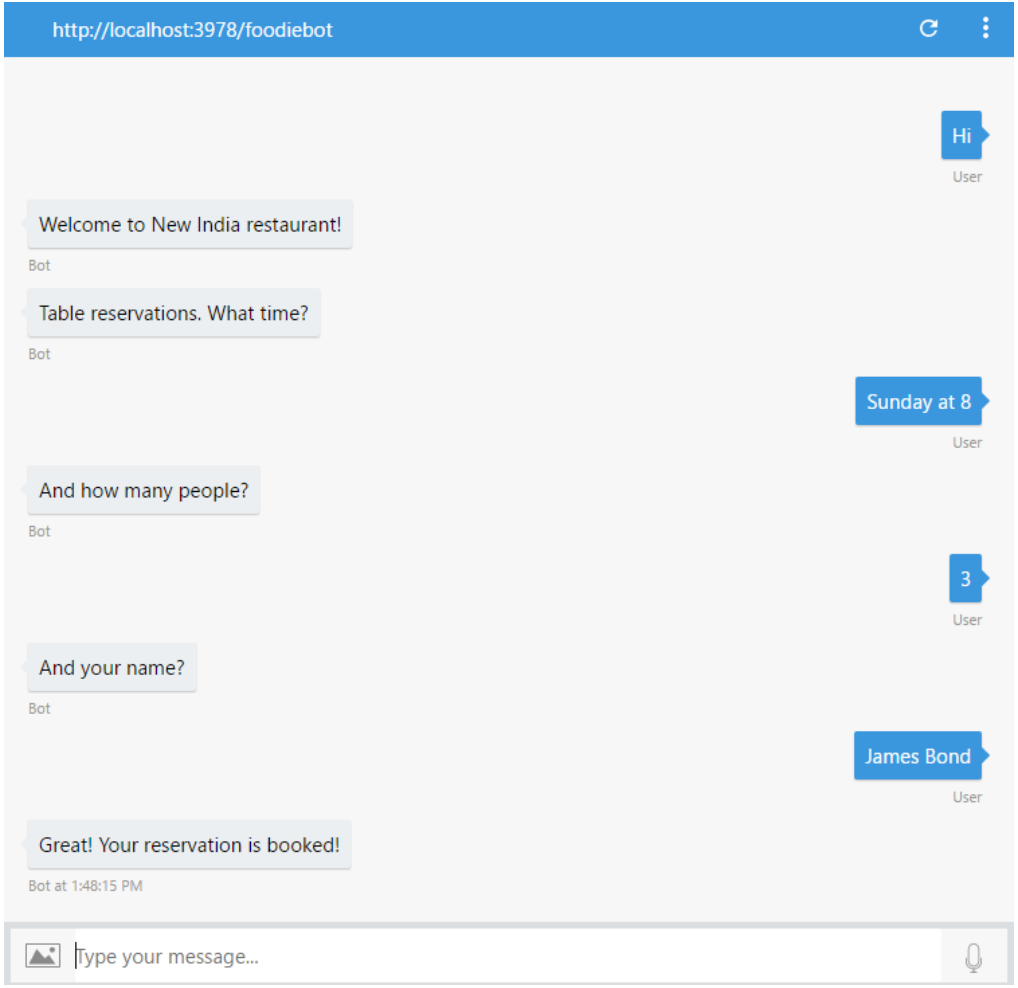
- Clicking the SignIn card takes the user to the web page where the user can be authenticated:



Rich messages

- Get the time of reservation, the number of people at the table, and the name of the user:

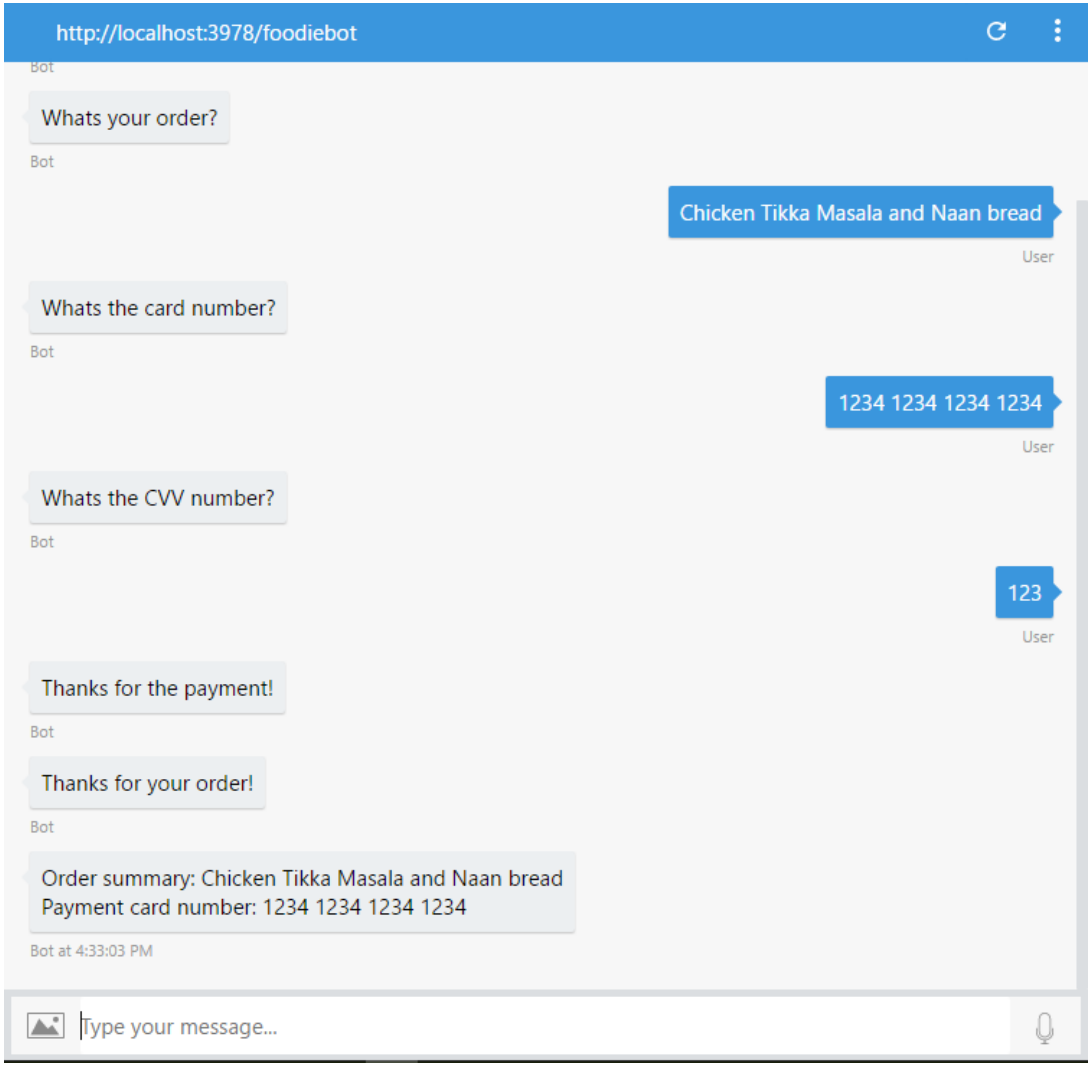
Refer to the file 6_7.txt


http://localhost:3978/foodiebot
Hi
User
Welcome to New India restaurant!
Bot
Table reservations. What time?
Bot
Sunday at 8
User
And how many people?
Bot
3
User
And your name?
Bot
James Bond
User
Great! Your reservation is booked!
Bot at 1:48:15 PM
Type your message...

Rich messages

- Let's now build a root dialogue and embed within it two sub-dialogues asking for the order and asking for payment:

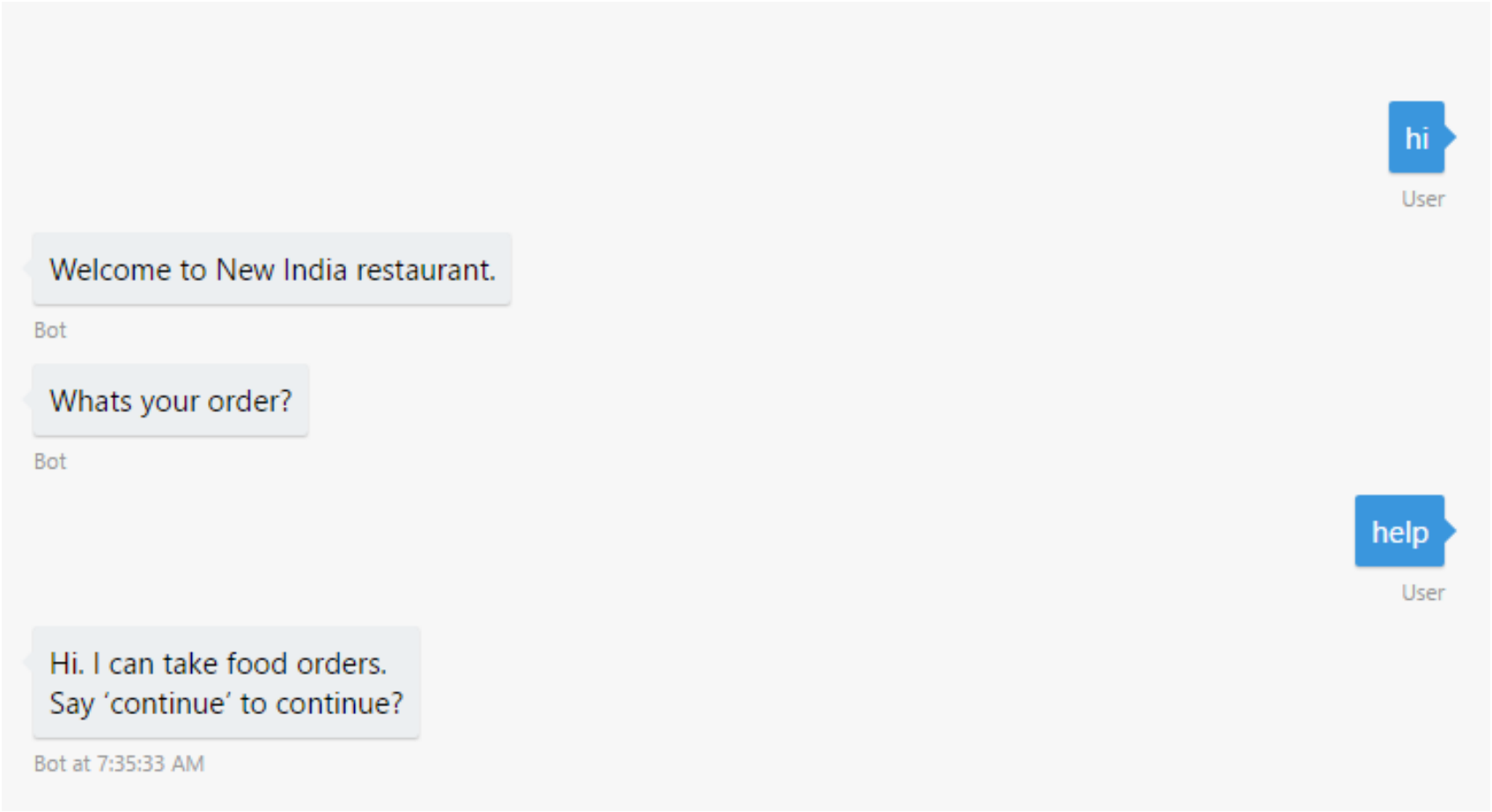
Refer to the file 6_8.txt


http://localhost:3978/foodiebot
Bot
Whats your order?
Bot
Chicken Tikka Masala and Naan bread
User
Whats the card number?
Bot
1234 1234 1234 1234
User
Whats the CVV number?
Bot
123
User
Thanks for the payment!
Bot
Thanks for your order!
Bot
Order summary: Chicken Tikka Masala and Naan bread
Payment card number: 1234 1234 1234 1234
Bot at 4:33:03 PM
Type your message...

Responding to user utterances

- Let us now explore how to respond when the user says help in the middle of the conversation:

```
bot.dialog('help', function (session, args, next) {
    session.endDialog("Hi. I can take food orders.<br/>
                    Say 'continue' to continue?");
})
.triggerAction({
    matches: /^help$/i,
});
```


hi
User
Welcome to New India restaurant.
Bot
Whats your order?
Bot
help
User
Hi. I can take food orders.
Say 'continue' to continue?
Bot at 7:35:33 AM

Responding to user utterances

- There are two other ways of interpreting user utterances: a custom recognizer and using NLU services such as LUIS.
- Let us try the custom recognizer first.
- To your bot, attach the following recognizer:

Refer to the file 6_9.txt

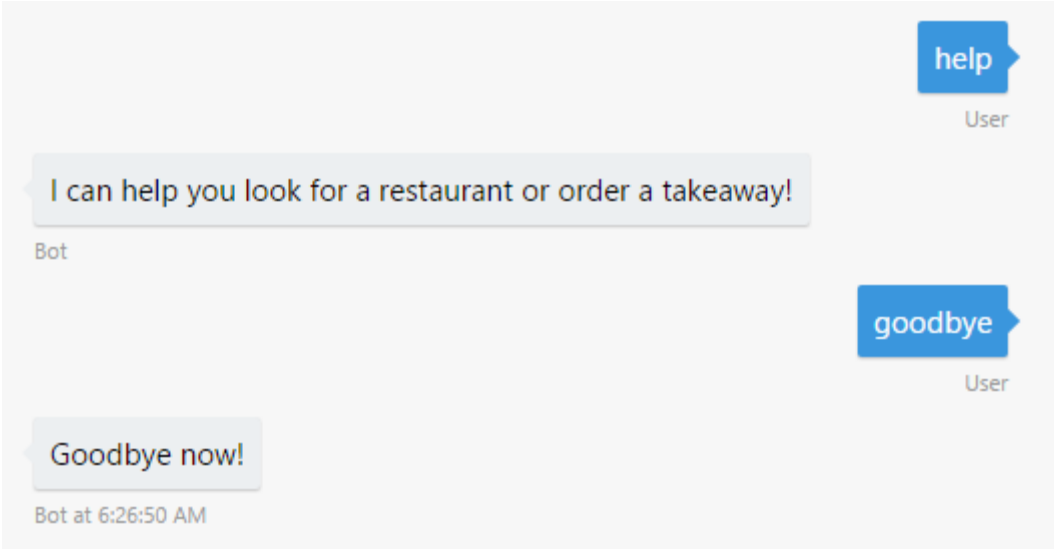
Responding to user utterances

- And create appropriate sub-dialogues for the intents:

Refer to the file 6_10.txt

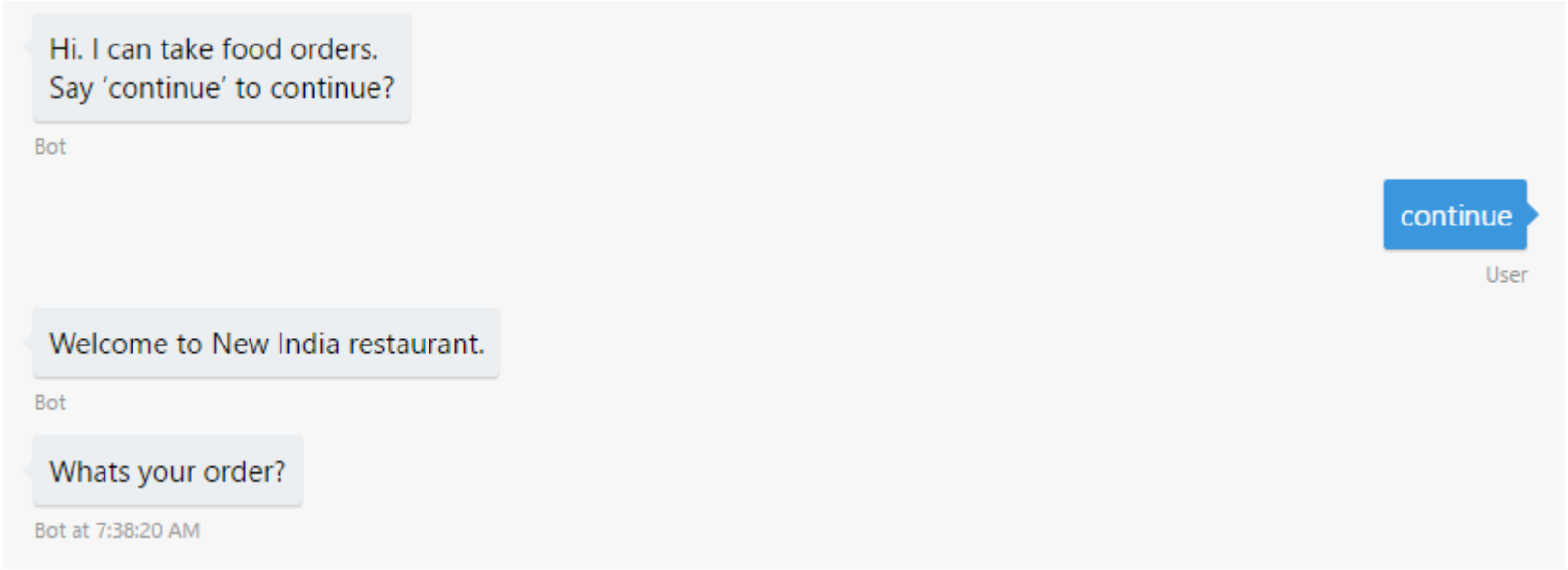
Responding to user utterances

- Once the intents are identified, they can be used to trigger appropriate sub-dialogues, as shown here:



Keeping context intact

- Processing user utterances using sub-dialogues can take the conversation out of context:



Keeping context intact

- There is a way to keep context intact even when users interrupt with questions and remarks.
- This can be done by adding the onSelectAction option to the sub-dialogue that gets invoked.
- This will keep the dialog stack intact and not clear it:

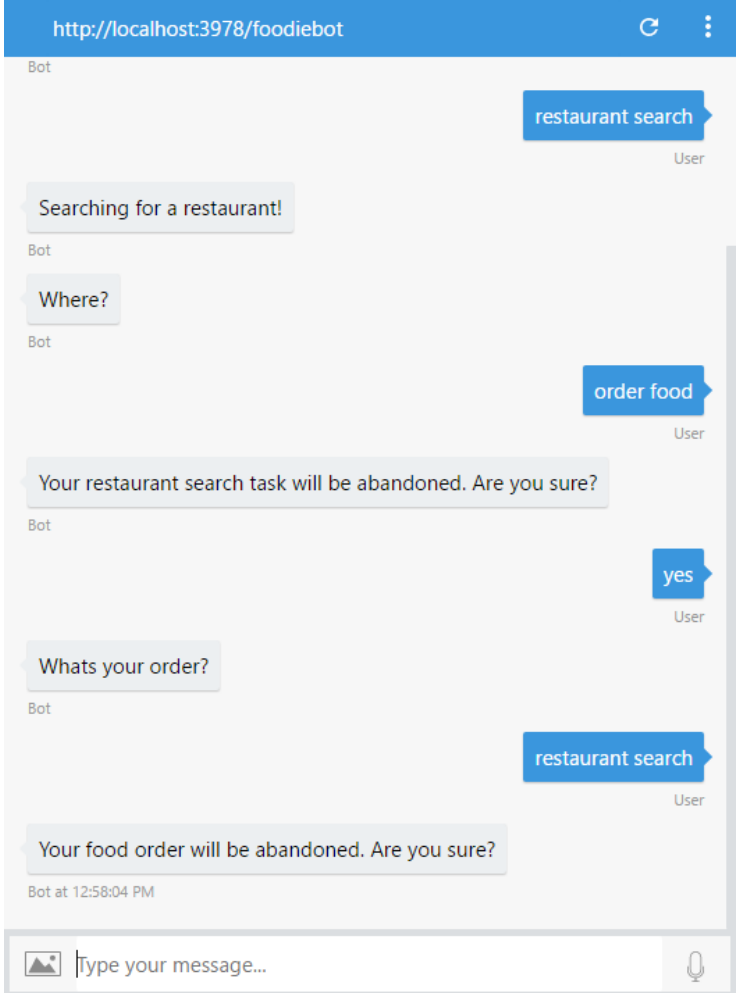
Refer to the file 6_11.txt

Context switching

- It is also a good idea to let the user know that the bot is going to abandon the current task to take up the next task.

- This can be done using the confirmPrompt option in the triggerAction() method:

Refer to the file 6_12.txt


http://localhost:3978/foodiebot
Bot
restaurant search
User
Searching for a restaurant!
Bot
Where?
Bot
order food
User
Your restaurant search task will be abandoned. Are you sure?
Bot
yes
User
Whats your order?
Bot
restaurant search
User
Your food order will be abandoned. Are you sure?
Bot at 12:58:04 PM
Type your message...

Contextual NLU

- Let us see how utterances can be processed contextually.

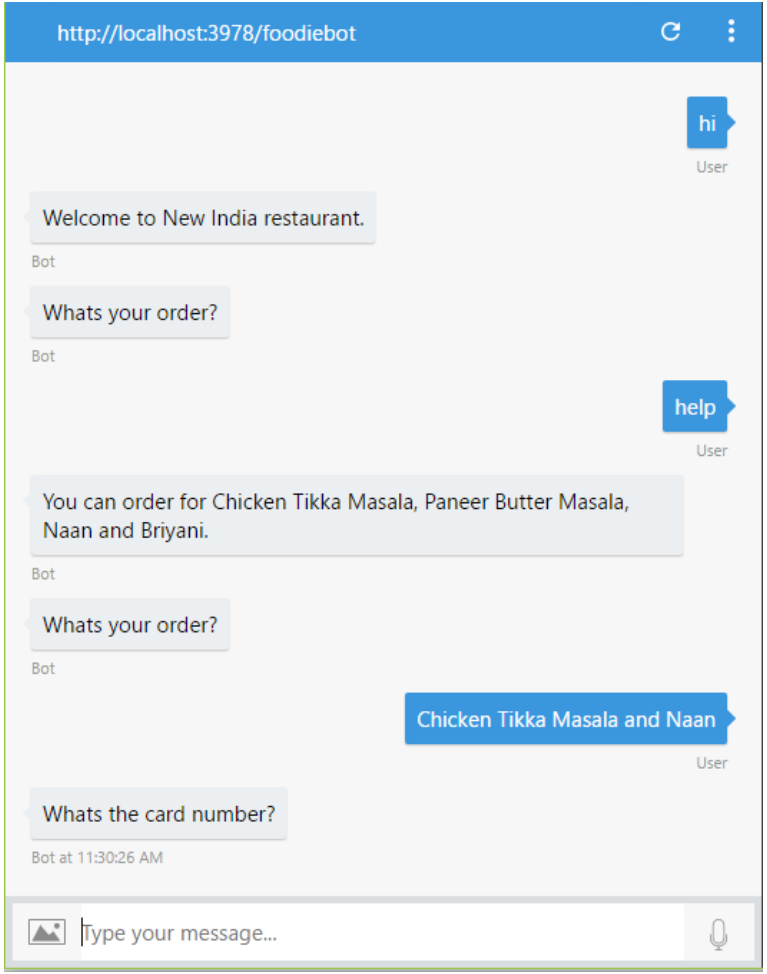
- To do this, we need to create a help sub-dialogue and append it to an existing dialogue so that it can trigger when what the user says matches the template provided:

Refer to the file 6_13.txt

Contextual NLU

- Notice how we use the beginDialogAction() method to link the orderHelp sub-dialogue.

- orderHelp gets triggered when the user says help during the food ordering step:



Ending the conversation

- Use the session.endConversation() method.
- So, let us rewrite our root dialogue with the session.endConversation() method:

Refer to the file 6_14.txt

Ending the conversation

```
bot.dialog('endConversation', [
    session.endConversation("Goodbye!")
])
.endConversationAction(
    "endTasks", "Ok. Goodbye.",
    {
        matches: /^goodbye$/i,
        confirmPrompt: "Cancelling current task. Are you sure?"
    }
);
```

Summary

- In the previous lessons, we explored and learned about how to build a chatbot using a variety of tools.
- These include development environments such as Chatfuel, natural language processing tools such as API.AI, and channel-specific SDKs such as Messenger SDK.
- However, when it comes to coding the conversational flow to manage the conversation, we either used form-based tools or built it from scratch.

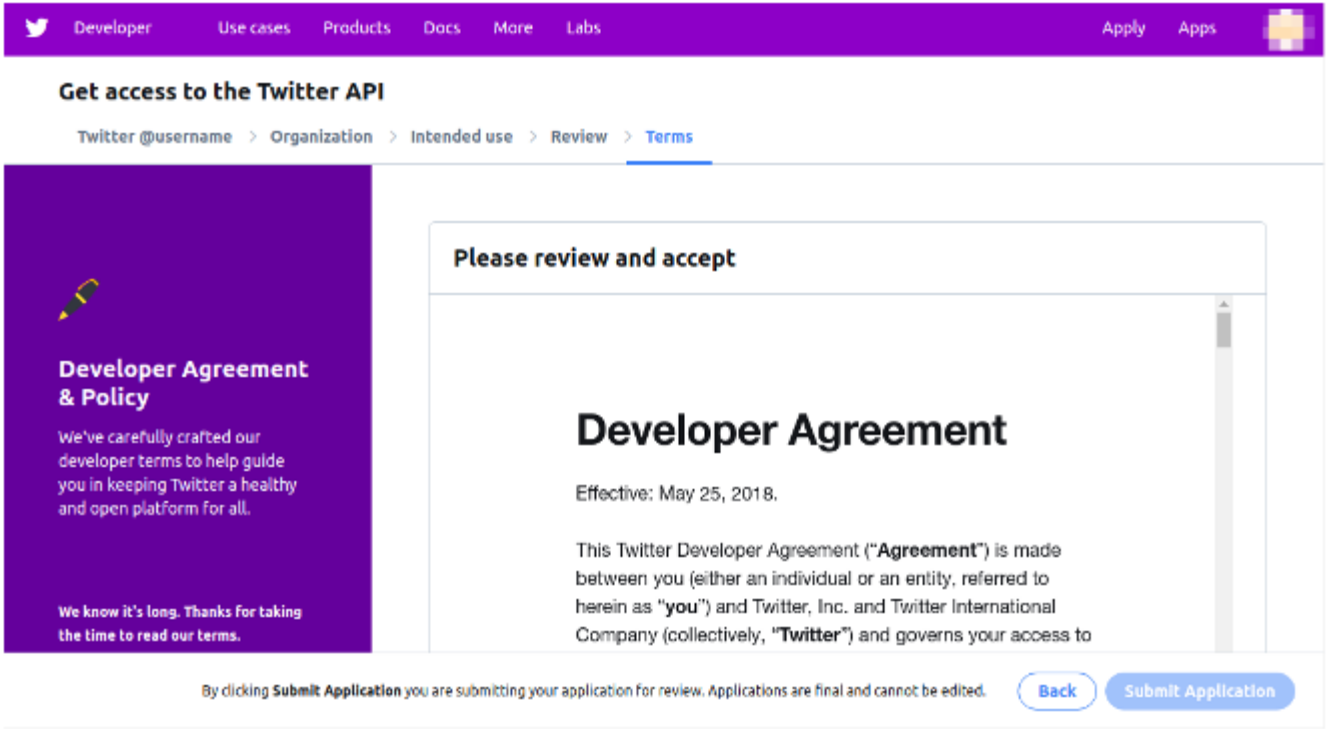
8. The News Bot

TriveraTech
TECHNOLOGY TRAINING

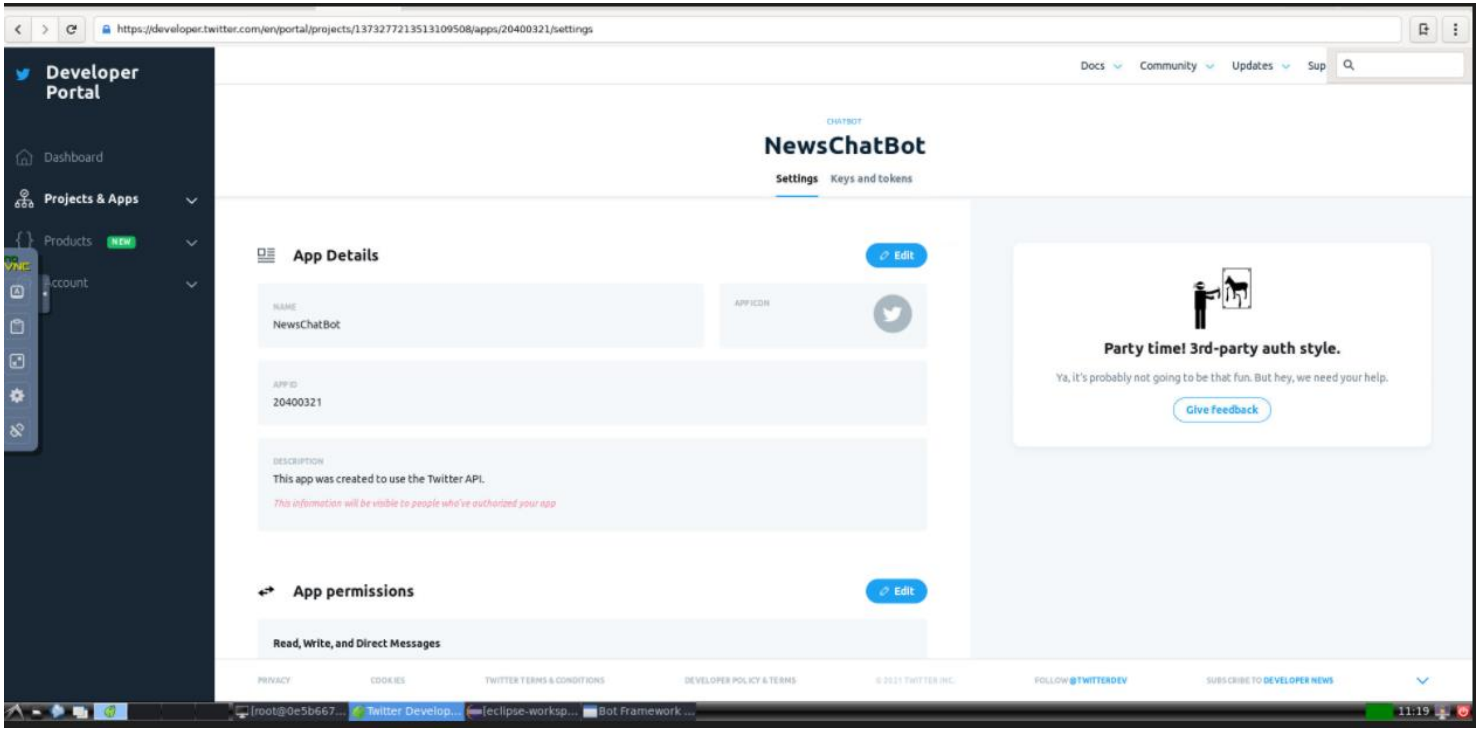
The News Bot

- Understand the basics of the Twitter API
- Create a bot that listens to hashtags
- Build a Twitter bot that tweets and retweets
- Integrate NewsAPI and tweet top stories

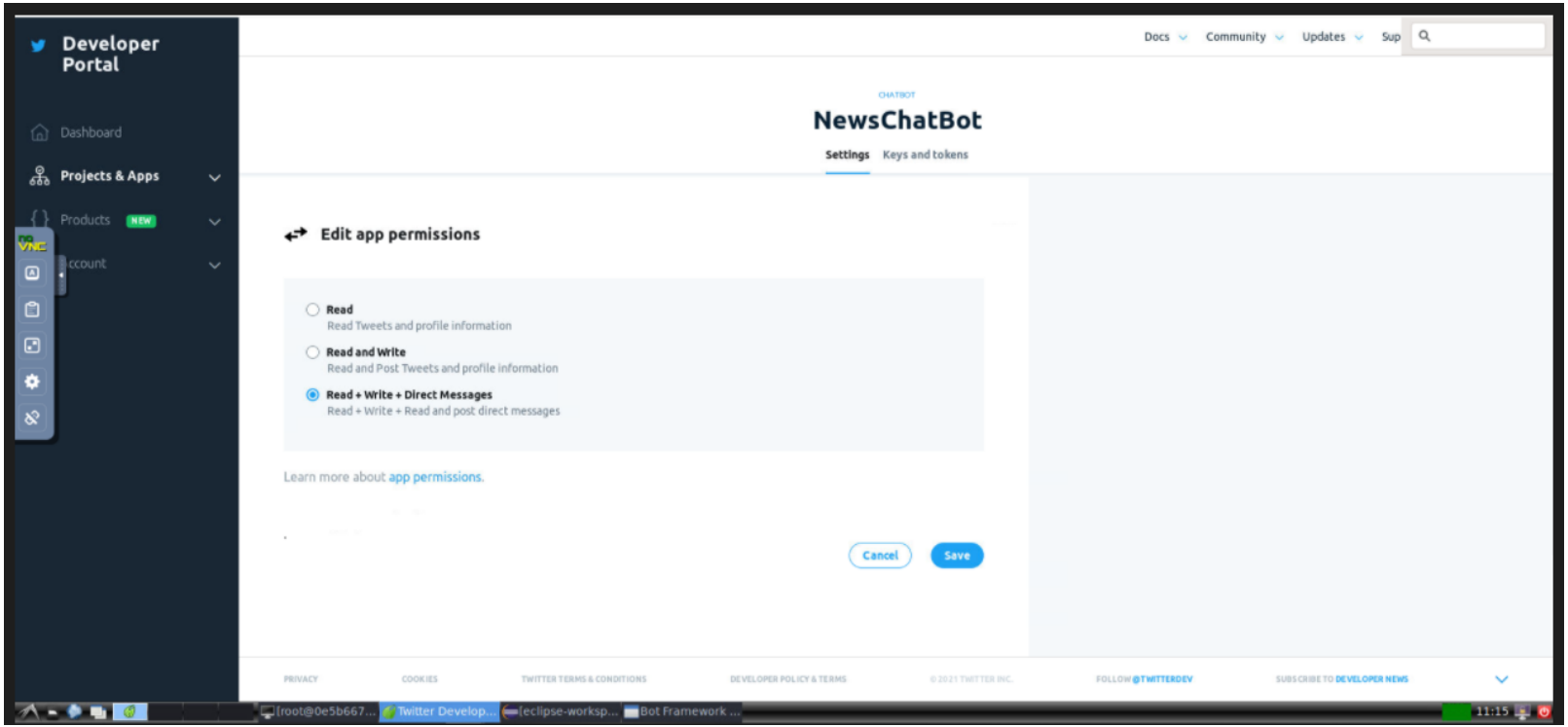
Getting started with the Twitter app



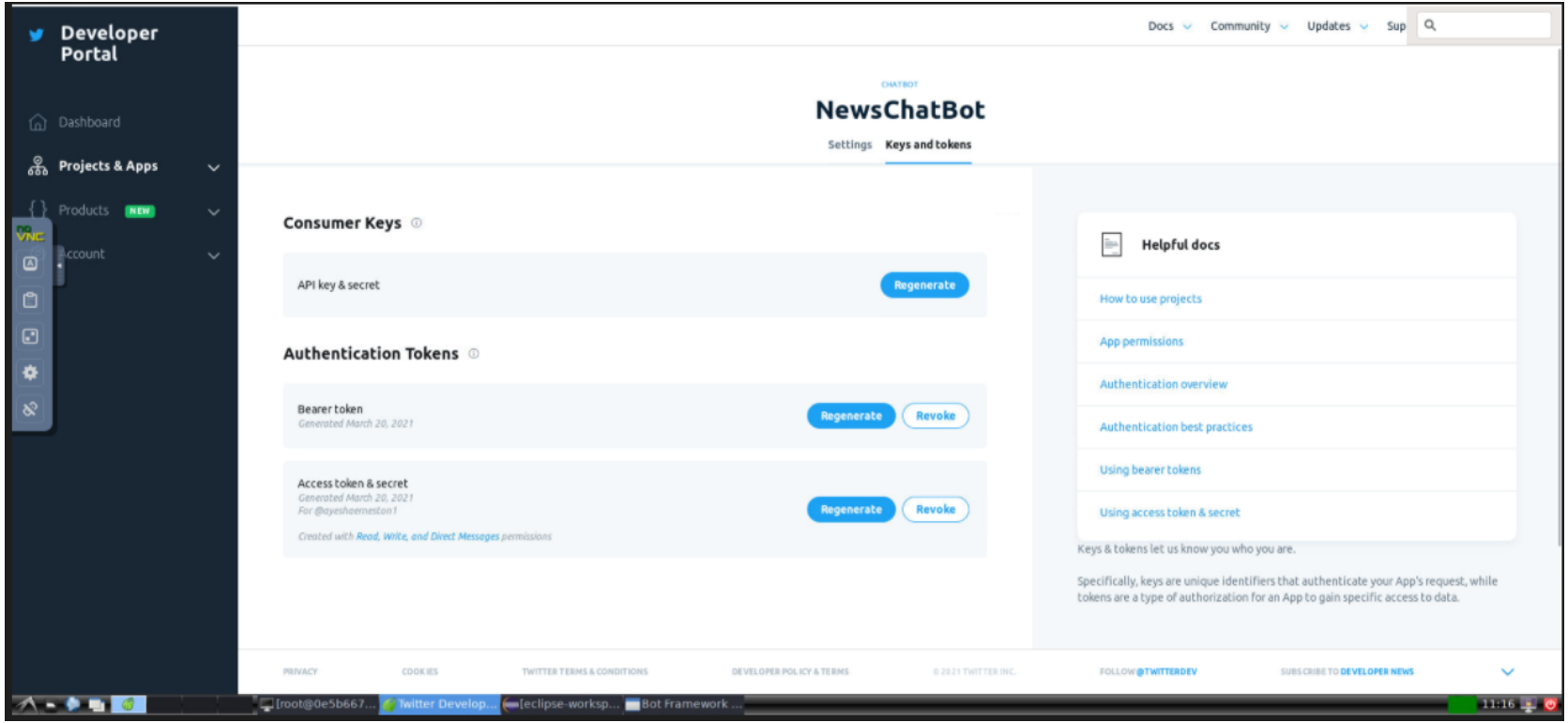
Getting started with the Twitter app



Change App Permissions



API Key, Access Token & Secret



npm init

```
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (twitterbot)
version: (1.0.0)
description: my news bot
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to  package.json:

{
  "name": "twitterbot",
  "version": "1.0.0",
  "description": "my news bot",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Building your first Twitter bot

- Execute the npm install twitter --save command to install the Twitter Node.js library:

```
root@3e3fad5c8103:~# npm install twitter --save
npm WARN deprecated request@2.88.2: request has been deprecated, see https://git
hub.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN twilio@3.58.0 requires a peer of @types/express@^4.17.7 but none is ins
talled. You must install peer dependencies yourself.
npm WARN twilio@3.58.0 requires a peer of @types/qs@6.9.4 but none is installed.
 You must install peer dependencies yourself.
npm WARN root@1.0.0 No description
npm WARN root@1.0.0 No repository field.

+ twitter@1.7.1
added 48 packages from 63 contributors and audited 89 packages in 3.859s
```

- Explore your package.json file in the root directory:

```
{
  "name": "twitterbot",
  "version": "1.0.0",
  "description": "my news bot",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "request": "^2.81.0",
    "twitter": "^1.7.1"
  }
}
```

- Create an index.js file with the following code:

```
//index.js

var TwitterPackage = require('twitter');
var request = require('request');

console.log("Hello World! I am a twitter bot!");

var secret = {
 consumer_key: 'YOUR_CONSUMER_KEY',
 consumer_secret: 'YOUR_CONSUMER_SECRET',
 access_token_key: 'YOUR_ACCESS_TOKEN_KEY',
 access_token_secret: 'YOUR_ACCESS_TOKEN_SECRET'
}

var Twitter = new TwitterPackage(secret);
```

- Now let's create a hashtag listener to listen to the tweets on a specific hashtag:

```
//Twitter stream

var hashtag = '#brexit'; //put any hashtag to listen e.g. #brexit
console.log('Listening to:' + hashtag);

Twitter.stream('statuses/filter', {track: hashtag}, function(stream) {
  stream.on('data', function(tweet) {
    console.log('Tweet:@' + tweet.user.screen_name +
                '\t' + tweet.text);
    console.log('------')
  });

  stream.on('error', function(error) {
    console.log(error);
  });
});
```

```
Hello World! I am a twitter bot!
Listening to:#brexit
Tweet:@belanisiya       RT @weloveeconomics: The UK economy grew faster after joining the EU but popu
lism (no facts!) made people believe the opposite.…
------
Tweet:@BazzieSmith      RT @jlivingstone100: Brexit support fading it seems -'Marketplace did not enc
ounter any farmers at the show who admitted voting for Brex…
------
Tweet:@frankietaggart   RT @brexitcountdow1: Brexit is 13850 hours away. #brexit
------
Tweet:@pm_kristin       RT @laute_europaeer: Ist nach Abschluss der #Brexit -Verhandlungen ein 2. #Re
ferendum über das Abkommen zwischen #UK und #EU notwendig?
htt…
------
Tweet:@GillianRAdams    RT @nickreeves9876: Shocking how seldom the question of Freedom of Movement i
s couched in terms of the loss of our Right to live &amp;amp; work in…
------
Tweet:@hpw_llp  RT @TheLawSociety: What is the European Court of Justice and why does it matter? A #B
rexit Q&amp;amp;A https://t.co/C0Vi6J0fm4 https://t.co/8ankRoV…
------
Tweet:@h1llbillies      RT @LeedsEurope: Leeds for Europe Stop Brexit Day of Action! 16 Sep - Rally,
Question Time &amp;amp; Social with @RCorbettMEP @emmyzen &amp;amp; mor…
------
Tweet:@KarenMc10        RT @trevdick: Britain 2017 with Brexit looming is peddling backwards to 1971.
..
Memories are not made of this...
#Brexit is Barmy https://t…
```

Exploring the Twitter SDK

Updating your status

- You can also update your status on your Twitter timeline by using the following status update module code:

Refer to the file 7_1.txt

Building your first Twitter bot

Retweet to your followers

- You can retweet a tweet to your followers using the following retweet status code:

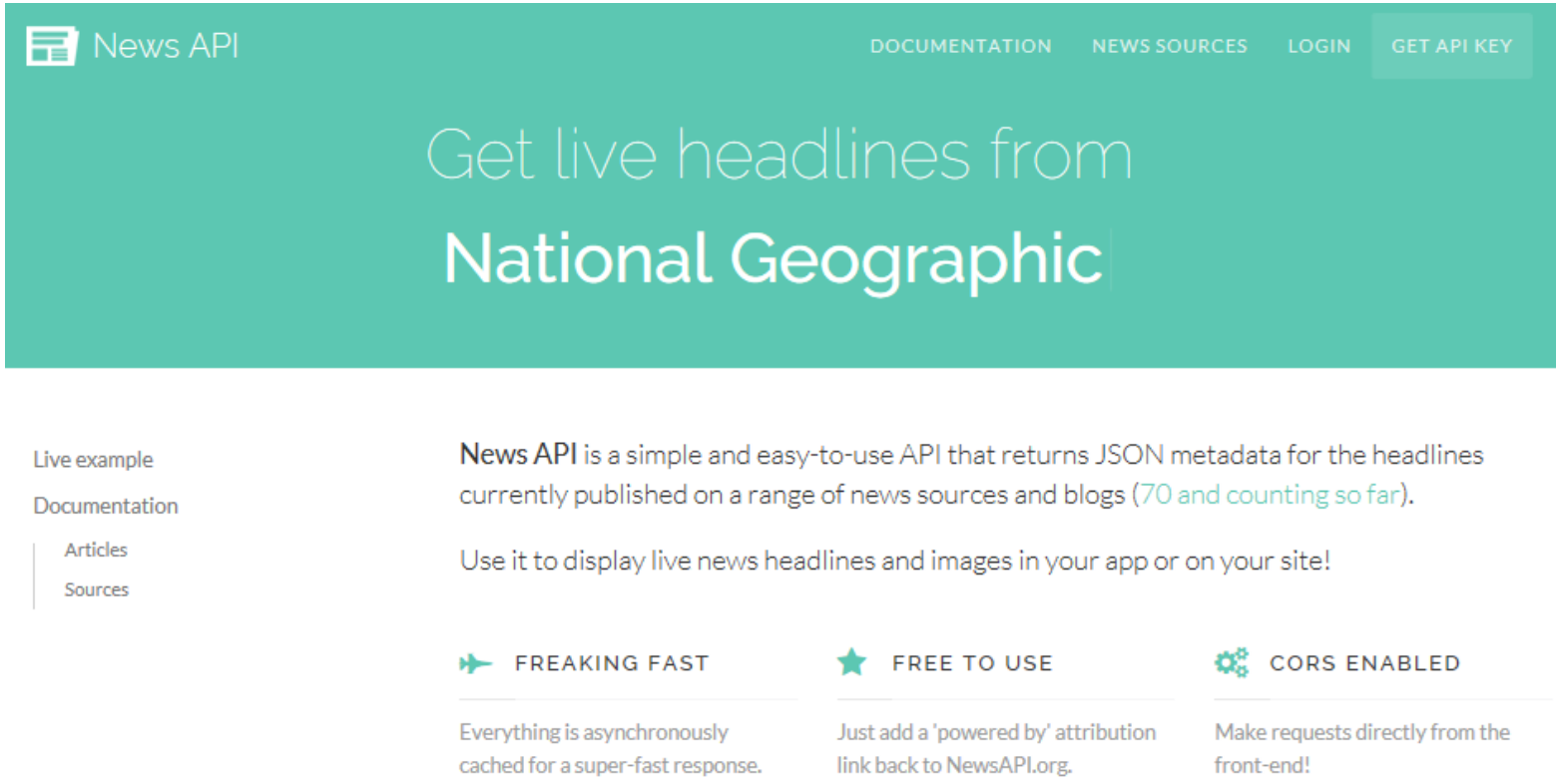
Refer to the file 7_2.txt

Searching for tweets

- You can also search for recent or popular tweets with hashtags using the following search hashtags code:

Refer to the file 7_3.txt

Exploring a news data service



Exploring a news data service

- You might also notice that each source also has language (en, de, fr) and country (au, de, gb, in, it, us) tags.

- The following is the information on the BBC-News source:

Refer to the file 7_4.txt

Exploring a news data service

- Get sources for a specific category, language, or country using:
- https://newsapi.org/v1/sources?category=business&apiKey=YOUR_API_KEY
- The following is the part of the response to the preceding query asking for all sources under the business category:

Refer to the file 7_5.txt

Exploring a news data service

Explore the articles:

- https://newsapi.org/v1/articles?source=bbc-news&apiKey=YOUR_API_KEY

- The following is the sample response:

Refer to the file 7_6.txt

Building a Twitter news bot

- Let's build a news tweeter module that tweets the top news article given the source.

- The following code uses the tweet() function we built earlier:

Refer to the file 7_7.txt

```
Hello World! I am a twitter bot!
{ status: 'ok',
  source: 'cnn',
  sortBy: 'top',
  articles:
   [ { author: 'Nicole Chavez, CNN',
       title: 'Hurricane Harvey strengthens to Category 2',
       description: 'As heavy rain and gusty winds move in over Texas, coastal residents are deciding whethe
r to flee their homes or to stay put and brace for a potentially life-threatening hurricane.',
       url: 'http://www.cnn.com/2017/08/25/us/hurricane-harvey/index.html',
       urlToImage: 'http://i2.cdn.cnn.com/cnnnext/dam/assets/170825094917-01-hurricane-harvey-nws-super-teas
e.jpg',
       publishedAt: '2017-08-25T10:25:21Z' },
     { author: null,
       title: 'What Hurricane Harvey looks like from space - CNN Video',
       description: 'The International Space Station captured video of Hurricane Harvey from about 220 miles
 above the Earth.',
       url: 'http://www.cnn.com/videos/us/2017/08/25/hurricane-harvey-space-station-sje-lon-orig.cnn',
       urlToImage: 'http://i2.cdn.cnn.com/cnnnext/dam/assets/170825100553-hurricane-harvey-space-station-3-s
uper-tease.jpg',
       publishedAt: null },
```

Building a Twitter news bot

Building a Twitter news bot

- Now, let us build a module that tweets news stories from a randomly-chosen source in a list of sources:

```
function tweetFromRandomSource(sources, screen_name, status_id){
    var max = sources.length;
    var randomSource = sources[Math.floor(Math.random() *
                            (max + 1))];
    //topNewsTweeter(randomSource, screen_name, status_id);
}
```

Building a Twitter news bot

- Let's call the tweeting module after we acquire the list of sources:

Refer to the file 7_8.txt

Building a Twitter news bot

- Let's create a new JS file called tweeter.js. In the tweeter.js file, call getSourcesAndTweet() to get the process started:

Refer to the file 7_9.txt

THANK YOU ☺