COMP 10222 - Emerging Web Technologies

Lab #2 - Slackbot with NodeJS

Due date: Friday October 9th at 11:59pm

Worth: 14% of total grade

Marks: 100 marks total

Learning objectives

- Create a Slackbot with NodeJS
- Use a third-party API to obtain desired data (YELP API)

Requirements

We will create a Slackbot using NodeJS and the YELP API. Our Slackbot will deliver information about local restaurants according to commands delivered in the Slack chat window.

YELP API

Use the YELP Business APIs to build the Slackbot. You will need to create a Slack account and an App within your account to obtain the required ID and secret information to obtain access to the API (see in-class notes). Use the yelp-api package to access the Yelp API.

Slackbot

Create a new Slack team with a URL of your choosing (whatever.slack.com). Your Slackbot should have the name **yelphelp**. Create the Slackbot as discussed in the Slackbot notes in-class (i.e. you will need to obtain an API token, and you must use the Slackbot NodeJS package).

Your Slackbot should respond to messages from any team member in the **general channel** only. You can manually add your Slackbot to the general channel. Your Slackbot should use the YELP API to query for the relevant information given the command, and present that information in general channel.

Your Slackbot should respond to the following commands. The syntax of the command is given in bold, and the expected behaviour is described. Your Slackbot does not need to respond at all if a message in the general channel does not conform to one of these commands.

Commands

- **Nearby Address** - the Slackbot should list the name and address of **any** 5 nearby restaurants in a neatly formatted way. Nearby restaurants will be considered those within a 10,000 meter radius of the provided Address. If no nearby restaurants

can be found, the Slackbot should output "No nearby restaurants can be found". *Example usage:* Nearby 135 Fennel Avenue West, Hamilton, Ontario

- **Events Longitude Latitude** - the Slackbot should list the name, address and description of **any** 5 events in a neatly formatted way. Search for events within a 10,000 meter radius of the provided longitude and latitude position. If no close by events can be found, the Slackbot should output "No close by events can be found". *Example usage:* Events 79.8861W 43.2383N

- **Top Xnumber Address** - the Slackbot should list the name and address of **the top Xnumber** nearby restaurants in a neatly formatted way, where the "top" restaurants are considered to be those with the highest YELP rating. Again nearby restaurants will be considered those within a 10,000 meter radius of the provided Address. If no nearby restaurants can be found, the Slackbot should output "No nearby restaurants can be found". If less than Xnumber restaurants can be found, the restaurants that have been found should still be presented. *Example usage:* Top 10 135 Fennel Avenue West, Hamilton, Ontario

- **Closest Xnumber Address** - the Slackbot should list the name and address of **the closest Xnumber** restaurants in a neatly formatted way. If no nearby restaurants can be found, the Slackbot should output "No nearby restaurants can be found". If less than Xnumber of closest restaurants can be found, the restaurants that have been found should still be presented.

Example usage: Closest 7 135 Fennel Avenue West, Hamilton, Ontario

- **FindMe Category Address** - the Slackbot should return the name, address and rating of any restaurant matching the category given that is within 20,000 meters of the given address. The category given should correspond to the **alias** field of the business search responses, for example "coffee", "sushi", or "seafood". If no nearby restaurant with the provided category can be found, the Slackbot should output "No category restaurant can be found" where category is the provided category (e.g. "No sushi restaurant can be found"). *Example usage:* FindMe sushi 135 Fennel Avenue West, Hamilton, Ontario

- **Reviews RestaurantName Address** - the Slackbot should return the review excerpt text, reviewer username, rating, and link to the full review, for three reviews of the restaurant with the name RestaurantName closest to the supplied Address. If no nearby restaurant with the provided name can be found, the Slackbot should output "RestaurantName cannot be found" where RestaurantName is the provided restaurant name (e.g. "Spring Sushi cannot be found"). *Example usage:* Reviews Spring Sushi 135 Fennel Avenue West, Hamilton, Ontario

- **SearchByPhone PhoneNumber** - the Slackbot should return the name and address of any restaurants found with the supplied phone number PhoneNumber. If no restaurant with the provided phone number can be found, the Slackbot should output "No restaurant with phone number PhoneNumber can be found"

where PhoneNumber is the provided phone number. *Example usage:* SearchByPhone 14165552222

- **StatusUpdate Status Message** - this should insert a record into an SQLite database with the exact same definition as Lab #1 (including a timestamp for when the message was received).

Command input formatting

Your Slackbot will need to parse the commands. You can make the following assumptions about the command-line input to make parsing the commands clearer and easier.

Assume that **Address** has the format **Street, City, Province/State, Country**, for example: **135 Fennel Avenue West, Hamilton, Ontario, Canada**.

Assume that **Longitude** and **Latitude** are formatted as a number followed by W, N, E or S, for example: **79.8861W 43.2383N**.

Assume that **Xnumber** will be no more than 20.

Assume that **PhoneNumber** is formatted as country code, followed by area code, followed by the rest of the phone number, for example **14165552222** (1 is the country code, 416 is the area code, and the rest of the phone number is 5552222).

Assume that **Category** will only be one word, for example "sushi".

Assume that **RestaurantName** may include spaces ("Burger King") but will not include any digits ("August 8"), i.e. you can use the first 0-9 digit after the word "Reviews" to know that the Address information for the command has begun.

You can assume that **Status** will only be one word in the StatusUpdate command.

Test session

Test your Slackbot by logging into your Slack channel and interacting with the bot. Run one test for each of the above commands which will produce the desired result (i.e. don't test for the "cannot be found" cases). Save a log of your tests in a file named **testlog.txt**.

brownek [12:11 PM]
Reviews Spring Sushi 135 Fennel Avenue West, Hamilton, Ontario

yelphelp BOT [12:13 PM]
Went for dinner with friends. Some colleagues happened to be there already, so that was encouraging. Out of the six of us, five got the mac and cheese with...
...
...
...
...

Hints

Read over the NodeJS Slackbots library.

Check out the YELP API documentation:

- Business search

- Phone search
- Business reviews

In the case of the **Reviews** command, you will likely need to use both the Business search API to find the business id based on the business name, and then use the Business reviews API to find the reviews.

Submission instructions

When you have completed the lab, put your source files in a folder called **lab2.zip** and submit it along with your **testlog.txt** file in the Dropbox. In your Dropbox submission comment box, specify the URL of your Slack team (whatever.slack.com). Invite kevin.browne3@mohawkcollege.ca to join your Slack team.

I will run your Slackbot on my machine, and test it in your channel.

Failure to follow submission instructions may result in a mark deduction up to receiving a mark of zero on the lab.

Statement of Authorship

Copying others people work and claiming it as your own is a serious breach of ethics. If copied work is found, all parties involved will receive a failing grade as per departmental policy. Group work is encouraged but it is expected that you do your own work. If you do, you'll learn the material and feel better for it. Since all work submitted is assumed to be your own original work, you must

include the following "Statement of Authorship" in every program file you submit for grading:

"StAuth10065: I John Doe, 123456 certify that this material is my original work. No other person's work has been used without due acknowledgement. I have not made my work available to anyone else."

- Use the exact text above. No substitutions.
- Replace John Doe with your name and the number 123456 with your student ID.
- Place this text as one line as close to the top of each document as possible.
- Include StAuth10065: at the beginning of each statement.
- Failure to include this statement will cause your work to be ineligible for grading.

Marking scheme

Slackbot library	10
YELP API	10
Commands	70
Test log	10
Total	/100

Labs submitted late will receive a 15% penalty per day. Labs more than 2 days late will receive a grade of zero. For example, a lab submitted at 11:01pm on the due date that was marked 80/100

would receive a mark of 80 - 15 = 65/100. A lab submitted at 11:01pm the day after the due date that was marked 65/100 would receive a mark of 65 - 30 = 35/100. A lab submitted at 11:01pm two days after the due date would receive a mark of zero.