**INSTALLATION & EXECUTION GUIDE**

The entire code was done in python 3.7 version. So it is preferable to have python 3.5 or later versions for this particular project.

In order to execute the code, following libraries must be installed first in the python3 environment.

**STEP 1:**

**Install the following python libraries using pip as follows:**

pip install tensorflow == 1.15

pip install tflearn

pip install nltk

pip install numpy

pip install json

pip install random

pip install pickle

pip install sqlite3

pip install datetime

Here we’ve particularly used **tensorflow version 1.15**, instead of the latest tensorflow version i.e 2.2, because the library ‘**tflearn’** **is not yet supported on tensorflow 2x version.**

**STEP 2:**

Extract the **executable\_codes.zip** file using any zip extractor.

It will have the following files.

**- HR\_Trained\_Bot.py**

**- app.py**

**- create\_db.py**

**- intent (Interview).json**

- model.tflearn.data-00000-of-00001

- model.tflearn.index

- model.tflearn.meta

- training\_data.pkl

By running the **‘HR\_Trained\_Bot.py’** filewill automatically create the last 4 files (not written in bold). Still provided the 4 files already, because running the ‘**app.py’** will be enough without training model repeatedly. The ‘**app.py’** loads the already trained model.

**STEP 3 (Optional): Run the** ‘**HR\_Trained\_Bot.py’ file**

As we’ve provided the already trained model files, it won’t require to run ‘**HR\_Trained\_Bot.py’**

file.

Yet, for the proof of concept we’ve provided the training file. Run the file in the python SHELL or IDE (eg: spyder) or even from command prompt(Windows)/Terminal(linux), run the file.

It will create the following files in the same directory,

- model.tflearn.data-00000-of-00001

- model.tflearn.index

- model.tflearn.meta

- training\_data.pkl

**STEP 5:**

Run the ‘**create\_db.py**’ file, inorder to create a table named ‘**applicants\_data**’ within the database

**‘HR\_bot.db’** within the same directory where the project is being done.

This is a **‘SQLite’** database, which is created within the local system and not in the cloud server.

However, in the next round we’ll be using cloud server type database like **MySQL** or **POSTGre,** for complete deployment of the model in the cloud infrastucture.

For the sake of **proving our concept of this project,** We’ve used  **SQLite** to keep it simple and locally executable.

**STEP 6:**

Run the ‘**app.py**’ file, which executes the main objective of the project.

The chatbot will start running. It’ll ask the job\_applicant his/her full name and mail id, and then it starts the conversation by greeting.

The flow of the conversation has been given in the **Test case document.** It’ll ask a couple of questions and will evaluate the candidate accordingly. At the end it will ask to type **‘exit’,** it is important to complete the interview process. The ‘exit’ will save the details of the candidate in the database.

**STEP 6:**

To check the database, if the details of the candidate is been saved or not, one can retrieve the data by Using simple SQL commands.

But to check the data in a GUI clean Tabular format,

Download ‘**DB Browser for SQlite**’ from the link

https://download.sqlitebrowser.org/DB.Browser.for.SQLite-3.11.2-win64.msi

and install it.

Now,

>> Open it

>> Click on the open database, and

>> Browse for **‘HR\_bot.db’** file and open it

>> Click on the tab, Browse Data.

All the candidates’s details will be there.

The purpose of the project was to ease the screening process of Job\_applicants’ resume. The database here shows the total point of each candidates. Based on company’s requirements and vaccancy (say 20 seats vaccant), that will select top performing candidates ( say Top 20), and will invite them for personal interview rounds.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_