BUSINESS-VISIBILITY BOT

This project is about building a chatbot with Natural language understanding and personalization using IBM Conversation API. It calculates values from a database.

It consists of the following:
public folder:
This folder consists of the following:
1. css folder : Consists of app.css, bootstrap.css and style.css
app.css is the css file for the main chatbot application, bootstrap.css and style.css are the css files for the layouts of the homepage and login pages.
2. fonts folder: It consists of different types of fonts.
3. img folder: It contains the chat button image.
4. js folder: It contains the files api.js, conversation.js, common.js, global.js that handle the interactions of the chat server with the users. The bootstrap.js file is the javascript file of bootstrap.

routes folder:
This folder consists of the following:

1. homepage.js : This is the js file that renders the homepage.
2. users.js : This is the file that manages the login, logout and the chat page. It passes the last 10 questions in the sql_customers table as suggestions to the chat page.

views folder:
This folder consists of the handlebars files. Handlebars is the view engine of this app.
It has the layouts folder that consists of home.handlebars which is default layout for this page.
index.handlebars: It is the view for the chat page.
login.handlebars: It is the view for the login page.
homepage.handlebars: It is the view for the homepage.

app.js :
It is the main file for this application. It handles the user input and responses from the chatbot. It also handles updating the table sql_customers when new questions are being added. If the size of the table sql_customers exceeds 100 the top entry in the table is deleted to ensure the table size doesn't exceed 100. Whenever the user types a question as an input, the user input is sent to the conversation service and it gives a positive response if the confidence level is greater than 75%.

update.js:
This is the file that executes the php file and updates the workspace. It has to be run whenever the database is updated.

chat.json:
This is the main json file that is imported into the workspace.

data.json:
This is the json file that contains the stored questions and the intents. The file chat.json is generated from this data.json file.

sql.php:
This is the php file that generates the chat.json file from the data.json file. It stores the account names from the database and adds them as entities to the chat.json file. It then creates dialog nodes for the 12 intents that are defined in the data.json file. If you want to add more questions to the chatbot then add intents to the data.json file and change the values in the sql.php file. The php file creates dialog nodes for each intent with the entities defined earlier as conditions. So for example for every user question an intent and entity is identified.
The value corresponding to the intent with the entity as condition is returned by the chatbot. These values are calculated in the sql.php file and are stored in the chat.json file.

Working:
Before starting this chatbot you need to create a database USERS with 2 tables sql_customers, sql_table.

```
CREATE TABLE sql_table(
        Ticket_Number VARCHAR(100),
        Assigned_To   VARCHAR(100) ,
        Account_Name  VARCHAR(100) ,
        Severity VARCHAR(100),
        Service_Offering VARCHAR(100),
        Additional_Info1 VARCHAR(100),
        Additional_Info2 VARCHAR(100),
        Category  VARCHAR(100),
        Status  VARCHAR(100),
        Last_Status_Modified  VARCHAR(100),
        Date_Created_Format  VARCHAR(100),
       Date_Last_Modified_Format  VARCHAR(100),
       Response_Date_Format  VARCHAR(100),
       Date_Closed_Format  VARCHAR(100),
       PER_Phase_Date_Format  VARCHAR(100),
       Parent_Ticket_Number  VARCHAR(100),
       Child_Ticket_Number  VARCHAR(100),
       PER_Phase  VARCHAR(100),
       Production_Implementation_Tracking  VARCHAR(100),
       Summary  VARCHAR(100),
       Time_Spent  INT,
```

```
    Age INT);

CREATE TABLE IF NOT EXISTS sql_customers(
    id INT,
    Question VARCHAR(100) );
```

The values in the chatbot will be calculated from the data in the table sql_table.
Now to run the application run node app.js
Then run node update.js whenever the table sql_table is updated.
Login with your ibmid to view the chatbot.
On the right hand side, the chatbot shows the frequently asked questions by other users.
You can click on these suggestions to ask those questions.

Instructions:

1. If you are importing a .csv file to the sql_table mentioned earlier make sure that it doesn't have entries with empty Account Name or empty rows or any extra columns.
2. Make sure that you don't change the column names in the sql_table mentioned earlier.If you change column names you will have to change the code in the php file.

Adding more columns to the table :

If you want to add more columns to the sql_table just add extra column names to the create table command. But make sure that you don't change the column names of the previous entries as it will affect the application.

Adding more questions to the chatbot:

The chatbot in this project has 14 intents.If you want to add more questions to the chatbot, you will have to create intents corresponding to each question in the data.json. You can store variations to the question in the intent. You then have to increase the value 12 in the sql.php file by the number of intents that you are adding. For example change the loop value for($j=0;$j<12;$j++) to for($j=0;$j<12+x;$j++) where x is the number of intents that you have added. You need to change values in line number 506, 561,571,581 of sql.php.
You will have to change the value 16 to 16+x in line number 512-531, 563,574,587,592,601,606,615,620,629,634,643,648,657,662,670,676,685,690,698,703,711,716 ,724,729,737,742. You then have to run update.js to update the workspace.