MultiTenant Chatbot with Dialog Flow and NodeJS

Chatbot for Educational Institutes, Movie Theatres and Insurance Companies

Keval Shah *(Author)*
Department Of Computer Engineering
San Jose State University
San Jose, CA, US
keval.shah@sjsu.edu

Sunny Udhani *(Author)*
Department Of Computer Engineering
San Jose State University
San Jose, CA, US
sunny.udhani@sjsu.edu

Chandan Paranjape *(Author)*
Department Of Computer Engineering
San Jose State University
San Jose, CA, US
chandan.paranjape@sjsu.edu

Mohit Shah *(Author)*
Department Of Computer Engineering
San Jose State University
San Jose, CA, US
mohit.shah@sjsu.edu

***Abstract*— This software is a multi tenant chatbot system for various industries, which provide conversational chat bots for specific purposes , keeping the mind the data security and confidentiality. This is a standalone chat bot with no external web hooks , to maintain data confidentiality. Each bot will provide basic features by conversing and engaging with the end user, to enhance the user experience in performing tasks, with the help of Dialog Flow for the natural language understanding.**

***Index Terms*— Multi Tenant , chatbot, data security, dialog flow**

I. Introduction

This software is a conversational chatbot for various industries, and can even be extended further than these three industries, with minimal effort. The software uses database level multitenancy to serve multiple clients from various industries. The chatbot aims to handle specific functionalities for each of the industries and will enable the user to easily query and perform activities. For example, it will enable user of educational institute to see their enrollment dates, it will enable the user of movie theatre website to view shows and book tickets and it will enable the user of insurance companies to know the things covered under a particular insurance plan, or to renew insurance.

The system uses Dialog Flow api for understanding the user input. This system is secure and does not store any sensitive data of the clients , which passes through the chatbot interface. The technology stack used is MERN stack(MongoDB, Express, ReactJS, NodeJS).

II. SERVICES OFFERED

The services offered by this chatbot to each client differs and depends on the functionality and services required. For the educational institutes, the services include but are not limited to payment of fees, viewing course information, viewing admission information, viewing graduate advisor information, viewing professor information. For the movie theatres, the tasks are a bit more specific , and the user can view shows, view movie ratings and book movie tickets. The services for insurance companies include comparing insurance plans, buying insurance, renewal of insurance plan, query regarding existing plan.

III. Ease of Use

This software mainly solves the optimization problem, of solving user queries and functions quickly and easily than the existing system. On measuring the time required for accessing the service and performing a specific functionality with the chatbot service and comparing it with the current system which requires accessing the services through the website, we found that the time taken to perform the same activity with the chatbot is 43% less than the time required for performing the same activity through the web interface,

IV. Comparison with Existing Chatbots

The main issues which are encountered in the use of the current chatbot systems is the breach of security. Many popular chatbot webhooks like Facebook, Telegram store sensitive information which is passed through their webhook and thus become a bad choice for the industries who deal with sensitive user data. The below chart identifies some of the vulnerabilities of other chatbot interfaces:

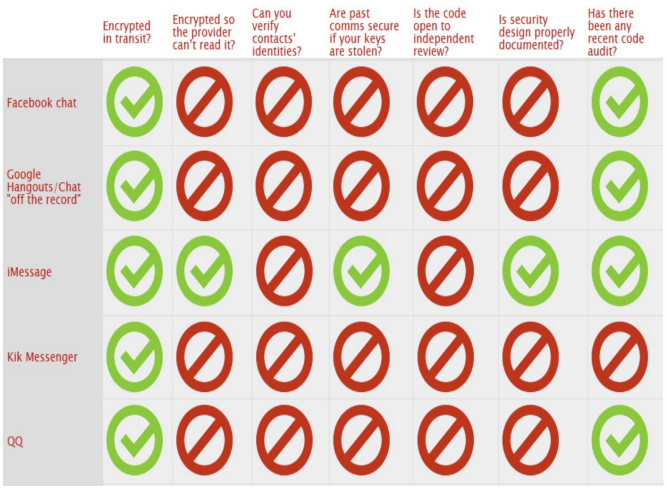
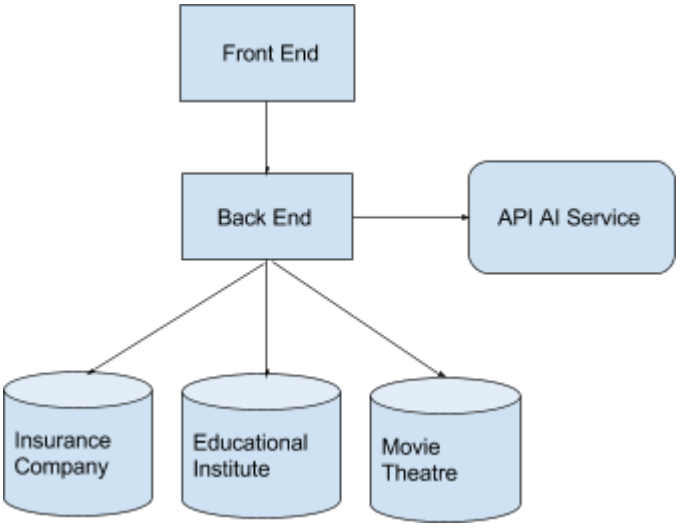


Fig-Security vulnerabilities of chatbot interfaces[1]

Our system provides security and ensures that client data cannot be read by the chatbot interface. Database level multi tenancy also assures that one client cannot access the data of the other client.This system can further be extended to include many clients for each category of insurance, education, movie theatre clients.

V. Our Chatbot Architecture

As our chatbot is a multi-tenant chatbot we have followed a very simple architecture for our project. Our frontend chat window client is made up using ReactJS and as it is a multi-tenant bot it can be easily customized as per the requirements of various companies like educational institutes, movie theatres or insurance companies. When a user types in a phrase in chat client it goes to the Node server. The Chatbot server or the NodeJS server further forwards the user phrase to machine learning NLP engine. In our case we are using Dialog Flow as a NLP engine. We have provided the functionality of making a multitenant NLP engine. It is easily customizable and can be trained as per use. NLP engine i.e Dialog Flow extracts user's intent and entities from the given phrase and sends it back to the node server. We have also provided multitenancy in using database. The entity extracted by the NLP engine is thereby forwarded to the Node server where the business logic is written. The intent is used to call upon proper service using the entity information to find the proper data. We have used MongoDB as our database. We have used MongoDB only because of the features which it provides. It gives us the functionality of sharding and replication at ease. Also by using MongoDB it is easy to scale our application for many users. So MongoDB returns the data back to the node server. The Node server packages the data into proper response and returns it back to the chat client. This was the simple architecture we followed which helps us to develop an efficient multi-tenant chatbot.



VI. Use Of Dialog Flow

We created a Dialog Flow agent which is used in our project for providing Natural Language Processing (NLP) for the user conversation input. The user can input anything into our chatbot, but we need to make sense out of it and give him an appropriate response. Moreover the response should also be such that the user doesn't feel he is talking to a machine but a real person. Natural Language Processing is required to understand what the user is trying to say and generate an appropriate response accordingly.
The first step in working with Dialog Flow was to create Intents. Intent are used for mapping what a user says to what a user is expecting as a response. For creating an intent we first have to set variations of text that a user can say. This helps the machine learning algorithm to train the agent. We return a customized json to the Node.js backend that processes it. In this json we send a parameter called query whose value will help the backend to generate a query to the MongoDB database. We send another parameter value whose value gives the value of the entity that was mapped in the user input. This entity value also helps in formation of the database query. Dialog Flow also has two built in intents called Default Welcome Intent and Default Fallback Intent. The Default Welcome Intent maps to the user input when he says things like "Hi", "Hello", "Good Morning" etc. The Default Fallback Intent matches to the user input when he says something that is
not matched by any of the other intents set by the developer

hence the name default fallback intent. It lets the user know with varied responses each time that it was not able to understand what the user is trying to communicate to it. Entities are created by the system as well as the developer

to get values from the user input and put it in the response. In our case we created a few custom entities like @course-name, @address-type etc. These entities if appear in user input can be
extracted and sent to the backend through custom json object. Some of the intents have these entities as required. This makes the Dialog Flow agent repeatedly ask the user for this entities in
different ways which is dependent on the number of prompt responses set. Entities are set by setting them in Dialog Flow. One way to do is to set reference values which have corresponding synonyms. These reference values are what is given when we try to access the values of entities matched in the request. The synonyms are what the Dialog Flow uses to match these entities in the first place. Another way is to just set
the words to match for the entity. Here we do not use any synonyms. We can also set automated expansion which is a feature given by Dialog Flow to automatically match any words
that are similar to those we have put in. We have done this for the entity called @course-names in which it will automatically match other course names.
Contexts is the concept by which a bot remembers what the user had said previously and what he is expecting as a response
now. Contexts can be set with the help of Dialog Flow. They are
set inside the intents. When a user says something it is matched
to the intent by matching what he has said with the user says section of the intent and matching the context that is currently set to the input context of the intent. Now the current context that is set is set by the output context in an intent. When a user says something and we want to continue the conversation then we set an output context in the intent. Then the Dialog Flow will
match only that intent which contains the same input context that the output context was currently set by the previous intent.

Hence we can take what we want from the user and generate appropriate responses accordingly.
When a context is set the response is already set in the agent. That response is returned as is by the back end to the user i.e the front end. When the context is not set then the response that is to be given to the user is generated by the back end. This is done by first matching the query to be generated that will query the database. Taking the data returned by the database a response is created by the backend and sent to the user i.e the front end.

Acknowledgment

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much pleasure. We would like to show our gratitude to **Prof. Rakesh Ranjan, Course Instructor, San Jose State University** for giving us a good guideline for project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this project.

Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

References

[1] https://dialogflow.com/docs/getting-started/basics

[2] https://tutorials.botsfloor.com/tagged/nodejs

[3] https://medium.com/@electrobabe/privacy-and-data-security-of-chatbots-6ab87773aadc

[4] https://www.csoonline.com/article/3226737/privacy/how-to-protect-chatbot-data-and-user-privacy.html.

[5] https://tutorials.botsfloor.com/tagged/nodejs

[6] http://www.information-age.com/rise-chatbot-security-concerns-123467697/

[7] The Chatbot Imperative: Intelligence, Personalization and Utilitarian Design