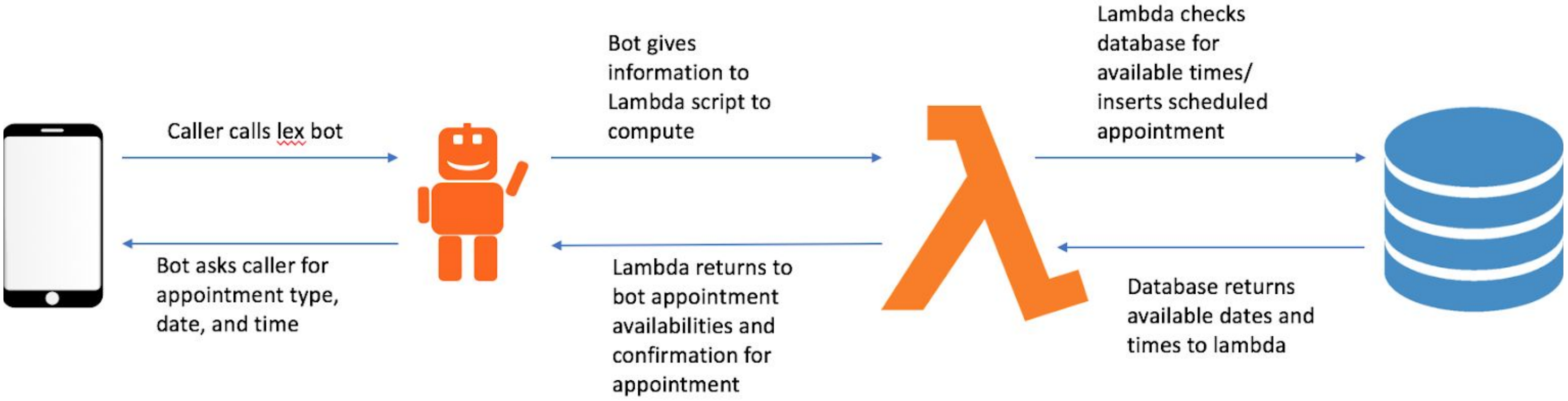
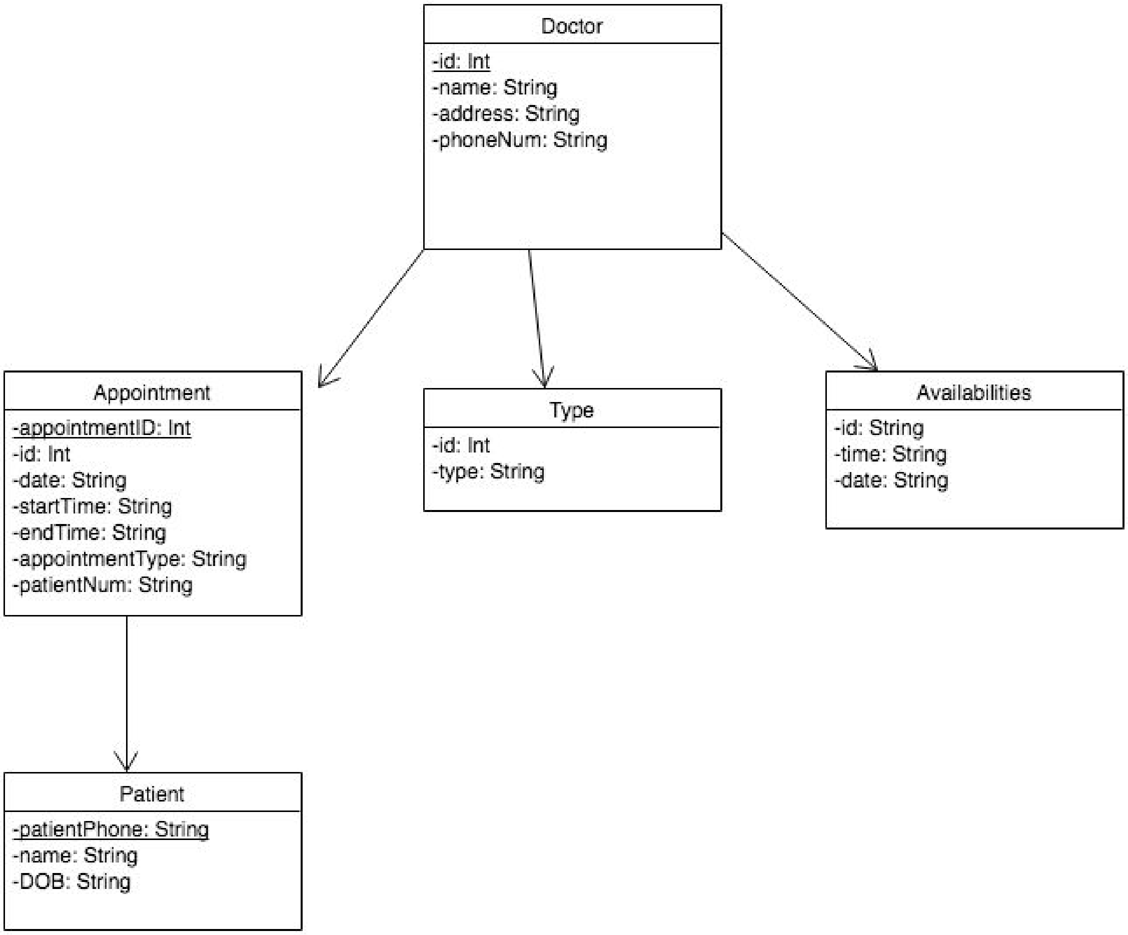
CS317 Envera Virtual Agent Design Document
Aaron Buehne
Davis Tran
Andy Huynh

We are developing an automated virtual agent for incoming calls to process the customer's needs like scheduling an appointment. Amazon Connect will be the call center to perform inbound calls. We integrate Amazon Connect to Amazon Lex to process the information that the customer is saying and passes it as input data to a Amazon Lambda function. Which is then translated through code to search for the corresponding result from a database. The result is then translated back into Amazon Lex which will respond back to the user with that result.


Caller calls lex bot
Bot asks caller for appointment type, date, and time
Bot gives information to Lambda script to compute
Lambda returns to bot appointment availabilities and confirmation for appointment
Lambda checks database for available times/ inserts scheduled appointment
Database returns available dates and times to lambda

Amazon Connect provides a number that a client may call. When the client calls, a contact flow is initiated. It would start by telling the customer who they have reached, ("Hello, you have reached Envera Health") then would go through a conversation flow with Amazon Lex. This is where the Lex chatbot will ask what the client needs and give sample utterances. If an utterance is correct the chatbot will proceed to gather more details, else the chatbot will say that it does not understand the utterance and will ask the question again. As the client goes through all the questions, Amazon Lex saves the data and then sends it to a Lambda function. We are writing the Amazon Lambda in NodeJS which connects to a database. The database will be created using Amazon Relational Database Service. Amazon Relational Database Service is a web service running in the cloud to simplify operations and scaling of a relational database.



The Lambda function uses the data from Amazon Lex and searches for the data in the database. If the piece of data exists, it returns it else it would give a different available option. For example, if a client is trying to schedule a dentist appointment on Tuesday. The lambda function will go through the database and join the Type table with the Doctor table to find appointments that are of type “dentist”. It will then join that table with the Availabilities table where it matches the requested date by the user. Then, it will look at the time slots to see if its free or not. If there are no available appointments that day, the chatbot would respond with, “Sorry there are no available appointments that day, but there is an opening on Wednesday at 4pm, would you like to schedule for that?”. When the customer accepts that date and time, it will remove that date, id, and time slot from the Availabilities table and then add the id, date, start and end time to the Appointment table. The rest of the information is then gathered from the user through Amazon Lex which then would be grabbed from Amazon Lambda and then inputted into the Appointment table.

If we have time, we are going to extend the project by implementing geolocation. To achieve this, we are going to use Google Maps. The lambda function will retrieve the addresses from the database and plots all the address on to Google Maps. When the customer calls and specifies a location when they want to set up an appointment, we return the top 3 locations that are closest to the user by selecting the locations with the least miles away.

Another way we can extend the project is by implementing Google Calendar. When the customer calls and book an appointment. The lambda function will add the appointment date and time into the database. It will then, get that date and add it on to Google Calendar. The Google Calendar will have all the appointment that the doctor has. This will be visually pleasing for the doctors and easily accessible for the doctors as they will most likely have their phones on them.

To measure the customer’s satisfaction, we would use sentiment analysis to measure how the customer is feeling. To do this, we would measure the volume of the user to determine if the caller is angry or happy. We could also, translate the users messages into text and feed them into a sentiment analysis API to see if the user message was overall angry, sad, happy, etc. This would help those who are managing the chatbot, gather information if the chatbot is doing well or how the customers overall feelings are.