Assistance Chatbot for Android Developers

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree

Master of Science in Software Engineering

By

Manasi Milind Joshi
Purvesh Kothari
Neil Thaker
Abhishek Upadhyay

Dec 2018

Manasi Milind Joshi
Purvesh Kothari
Neil Thaker
Abhishek Upadhyay

APPROVED

Charles Zhang

Digitally signed by Charles Zhang
Date: 2018.12.07 20:42:44 -08'00'

Professor Charles Zhang, Project Advisor

Professor Dan Harkey, Director, MS Software Engineering

Professor Xiao Su, Department Chair

ABSTRACT

Assistance Chatbot for Android Developers

By
Manasi Milind Joshi
Purvesh Kothari
Neil Thaker
Abhishek Upadhyay

Android is the most popular operating system for the smartphone. Recently, Google announced that there are 2 billion monthly active devices which are running Android. According to Statista, number of devices using Android has been increasing since 2009. These facts encourage the companies and Android developers to build innovative and useful apps on the platform to reach broader user base.

The current web based search engines require Android users and developers to search with specific and simple keywords rather than in a full question format. This restricts the android developers to express their queries as they like. Secondly, the web based search engines don't maintain a context based conversational search which the user would like to have. Also, Typical site search won't let the user to provide a feedback in the midst of his activity, which an FAQ based chatbot can do. Users are craving for easy, fast and efficient search with complex details present in the online search engines, a chatbot provides them an easy approach to find the right information.

In this project, we propose a solution to crawl existing Android questions and related answers available on Stack Exchange website. This dataset is used to train our models using machine learning algorithms. The goal is to provide the best possible solution to the Android questions within a few seconds. A user will have interactive experience with the chatbot using the web client interfaces.

Acknowledgments

The authors are deeply indebted to Professor Charles Zhang, for his invaluable comments and assistance in the preparation of this study.

Table of Contents

Chapter 1: Project Overview

1.1 Introduction

Search engines are the types of a bot. We use Google Assistance or Apple's Siri or Microsoft Cortana, they take our command or request in the form of input and return several results based on the user's question. This Virtual assistant or bot make use of machine learning or artificial intelligence algorithms and make things very easier and accessible to the user. The virtual assistants are used to perform several tasks like scheduling a meeting, sending an email, setting up of reminder and so on. The chatbot is trained in such a way that it interacts with the user using the natural language. These days chatbots or the virtual assistants can be used at several places such as information retrieval over the world wide web instead of searching over the web. Chatbots are a type of virtual assistant which can help you save your time so that you can utilize your time in some other productive way rather than wasting it searching on the Internet.

Chatbots are used these days for several purposes like the information retrieval, business purposes, education, e-commerce, entertainment and thereby reducing the human time and efforts. Chatbots are trained to understand the natural language and they are trained in such an intelligent manner that they take inputs in various forms like text, voice, or both and provide the conversational response to the user. Various chatbots are available in the market but the most reliable chatbot is the one which stores the previous conversations, responds back quickly and efficiently. Thus for this purpose various research are conducted to ensure the quality issues and attributes of the chatbots. There are various building blocks for developers to create chatbots inexpensively. While developing the chatbot the parameters like extensibility, scalability, and maintenance are required to take care of. AWS lambda serverless processing is used by the developers for the stateless functions. The serverless architecture is based on the functions and these functions are used to coordinate the cognitive microservices.

The serverless model improves the extensibility of our chatbot. The main objective of our project is to build an advanced conversation AI bot for assisting the Android developers and users. They can leverage it when they need to get a quick solution for their query related to any Android development and technologies rather than spending quite a reasonable amount of time looking for the solution on sites like "StackOverflow" or "StackExchange".

1.2 Proposed Areas of Study and Academic Contribution

The idea behind developing this chatbot is to make the life of Android developers community easier by providing them the fast and correct response to their queries. We are developing the bot with niche tools and technologies. While starting to implement our very idea about making a reliable, humane, efficient chatbot we had to look around and study about various tools and techniques. Finally, the areas of study that we proposed comprised of understanding the Google's Dialog Flow component, the Amazon Alexa Skill Kit, the Natural Language Processing, AWS Lambda, Website Crawling tools such as Scrapy, and Machine Learning algorithms for finding the best matching response to a question. The main motivation for us to make this study proposal by delving ourselves into studying the nuances of these tools and techniques was to provide an engaging voice and text-based conversational interfaces for Android users.

The below points draw out the main reasons why we are proposing these areas of study for each of the above tool and techniques

DialogFlow- It is one such API by Google which allows developers to build "engaging voice and text based conversational interfaces powered by AI" [5]
Amazon Alexa Skill Kit - We wanted to create a rich, voice-text based interface that includes voice, visual, and touch interfaces. By leveraging the ease to design and develop Alexa skill we wanted to provide users with the real-life human interaction experience for resolving their queries.
AWS Lambda - AWS Lambda is a compute service that lets you run code without provisioning or managing servers [6]. We wanted to build a scalable system and handle millions of requests without affecting the performance of our bot. Lambda let us build API that can execute the logical code only when needed and can scale our system to handle few requests per day to thousand requests per second.
Crawling the Website - We planned to build a bot with the capability to answer the queries that one can look for on the website. Our bot should have to be capable of answering those queries if any user asks them rather than visiting the website and search over there. That's when we thought of training our chatbot application by making it crawl all the questions and answers from the website that one Android user could possibly ask.

Even though the study areas that we proposed helped us to start building this Chatbot, we faced few challenges also to make the chatbot. Like, we had to look for ways to provide response within the shortest possible time. The other challenge here is that the proper response should be returned in minimum time and most accurate. For this purpose the setting of the intents and entities in the DialogFlow should be very accurate. The data crawling should be done in a most optimized way. The length of the response should be really small and to the point. The other challenge here is to understand the various accent. That is to make sure we have the internationalization in our system. The other things than should be taken care in our project are providing the default response to all the user requests. So, these are some of the research areas on which we are going to work on and apply the most accurate machine learning algorithm to it to return the most accurate, short and quick response. The other areas to study are related to how the intents and entities can be set for the huge set of questions and answers.

Academic Contribution - Even though there are numerous bots in the market providing different services for the industry and academic institutions, our bot specifically targets the Android developers community by helping them with their queries for which they usually go to the Stack Exchange website. In today's world, chatbots are becoming popular on many universities and colleges' websites. They provide the runtime assistance to any query that a student or any other person might have. Our Android Assistant Chatbot can also make an impact on the students lives once it is included on the university portal assisting students seeking technical solutions to anything related to android queries. The visual API of a chatbot can even help engage students to their subject by learning from the bot itself while they are preparing for any exam or test or just seeking help in their school project. People from different sections of the university mainly the professors, students and the Computer/Software engineering departments' technical staff can leverage the features of this bot. The professors or the teaching assistants can grade their papers by taking help from the bot. As our bot will be coming with a quick fire question-answer feature, professors can take those questions to set-up a quick interactive in-class quiz using this bot.

Industry Contribution - Not only these Chatbots will provide numerous advantages to make the university class programs more effective and engaging for students and professors, it will also provide many benefits to the technology industry. Our Android Assistant Chatbot can come handy for the tech industry providing real-time services to the Android users. Off late, we have seen an immense increase in the number of users expecting a Chat assistant feature for any website they use on a daily basis. Many customer services oriented businesses believe that Artificial Intelligence tool could help their companies. But are not sure if their business is sophisticated enough to implement Chatbots in their systems.[8] Chatbots can help gain the number of customers for any tech company in our case by implementing the AI feature effectively and providing an engaging platform to the user. Helping users by responding to their questions in real time can bring more trust to the user for that website and its services. There are many android engineers and developers who seek help for their technical queries going to many technical websites like Stack Exchange or Stack Overflow, our Chatbot can save their effort and time by providing them a quick and relevant response to their queries. These bots also come with an Alexa feature which can be deployed on Amazon Alexa as a new skill and can help millions of users in the Android community. Many tech industries working on Android projects can take benefit out of these bots by including them in their day to day work life. Consequently, they will be able to provide fast, trustworthy solutions to their clients and gain business revenue.

Chapter 2: Project Architecture

2.1 Introduction

The design implementation of the chatbot is a cloud hosted platform that receives the Android related questions in the textual or voice format from the end user and finds out the most similar match of the question from the database. The answer of the question is searched through the available dataset and answer is provided in either text or voice format based on the client from which the request has been made. We will host the entire service stack on the Amazon Web Service cloud.

Following is the high-level architecture diagram of our implementation of chatbot application:

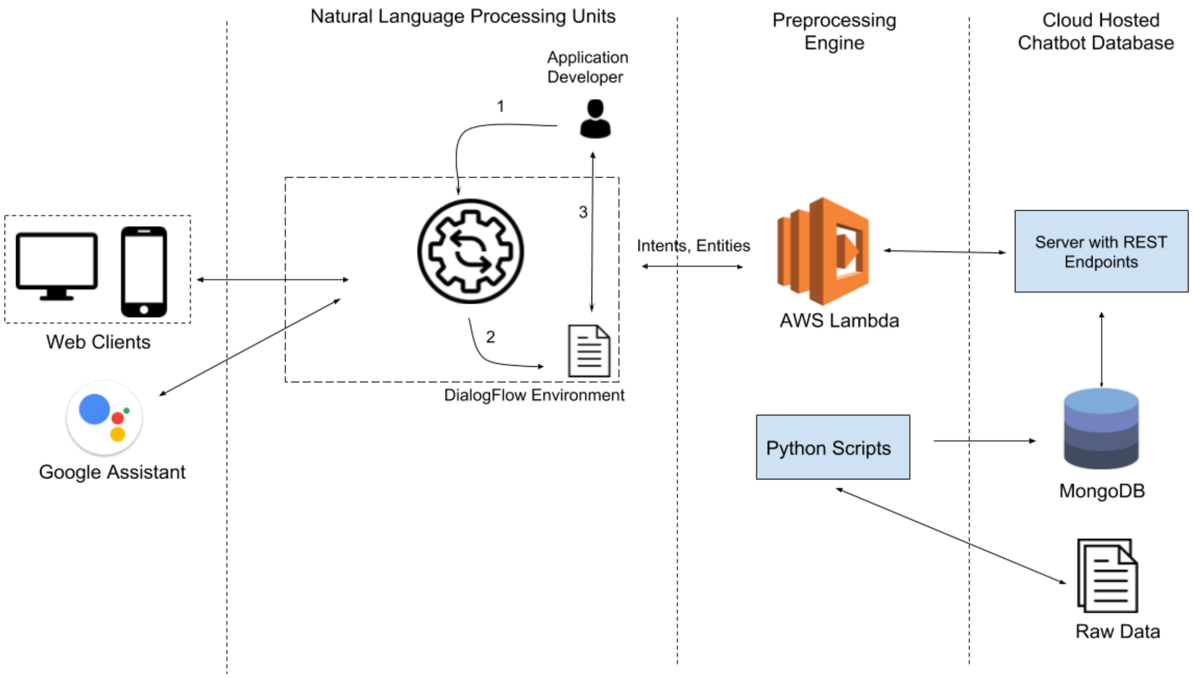


figure 1: Project Architecture

We have four main components in the system. Following is the high-level description of their working:

Clients

These are the end user applications which users will be using to interact with the chatbot. Web application and mobile application will have rich user interfaces. For the voice assistant, we will use the Google Assistant to have a voice-based natural conversation through Google Assistant enabled devices.

Natural Language Processing Unit

This is the heart of the project. To extract the information from the question provided by the user, it is important to process the question and figure out what type of information is requested by the user. We will use the DialogFlow provided by Google for the natural language processing.

Serverless Processing

Once the natural language processing phase has been completed and important information has been extracted from the question, that information needs to be sent to the backend server to get the correct answer. The natural language processing units DialogFlow requires the connection with backend in certain requirements. Serverless Processing provided by AWS Lambda is the recommended way to get the information processed by these services as it satisfies these requirements for the security. It works as a bridge between the third-party natural language processing services and custom built backend server.

Backend APIs and Database

The dataset with Android related questions and answers has been taken from the StackExchange website. It has several hundred thousand questions related to the Android development. To search through the dataset with the information retrieved from the natural language processing, we will develop APIs. These APIs aim to search through the questions and provide the most relevant answer to the question. Using scripts we fetch the data from csv files and after structuring it, data is stored in the database.

2.2 Architecture Subsystems

Client Applications

These are the applications with which end user will be interacting to access the chatbot. We are going to have a web application, mobile application, and a voice assistant application. Having the clients across multiple platforms helps a product to reach to many customers with different devices.

a. SaaS-based Web Application:

For any chatbot application, a web application is a necessary component so that user can access the chatbot across any standard browser on different OS. We will have the web application developed in ReactJS. ReactJS is a JavaScript framework based on NodeJS that helps to build an application with a dynamic user interface. We will host this application in the cloud so that users can access it from anywhere. Client-specific information will be stored locally on the device as browser sessions and cookies.

Following is the architecture of the web application:

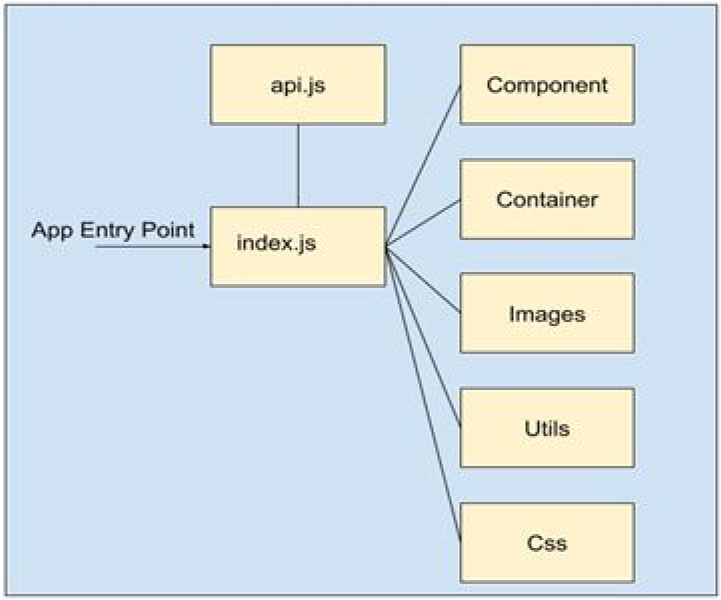


figure 2: ReactJS based web application architecture

- index.js: It is the application entry point. This file will initialize the application.
- api.js: This file will contain the code to make the API calls to the backend. In this case, it will use the DialogFlow SDK to make the call and get the response.
- Component: This folder contains the all the presentational components. These components will be stateless.
- Containers: This folder contains the state full components of the project.
- Utils: This folder contains the source for different utilities.
- images/CSS: These folders contains static resources like images and CSS to be displayed on the front-end.

Natural Language Processing Units

This is one of the most important components of the project. Development of this component directly impacts the accuracy and success of the chatbot. To understand the meaning of the questions users have asked, it is necessary to process the question to get key information like type of the questions and other specific information to which user is seeking the answer. We are going to use two different natural language processing and understanding services to implement the same task. One is DialogFlow by Google. DialogFlow is used for the text-based application of chatbot and voice-based application of the chatbot.

There are two important factors in these NLP services. One is the intent and other is the entity:

- **Intent**: Intent is something that a customer wants to do with application and we as a developer want to handle using the chatbot application. In other words, intents are the intentions of the end-user.
- **Entity**: Entity is a term or object in user's input that provides more information about the context of the intent. An entity could be a class of the objects which could be possible values of that entity. An entity is a metadata of an intent.

For example, if the user asks that "What is the name of Android version 7?". Then "version 7" could be a possible entity with an intent of knowing about the Android version.

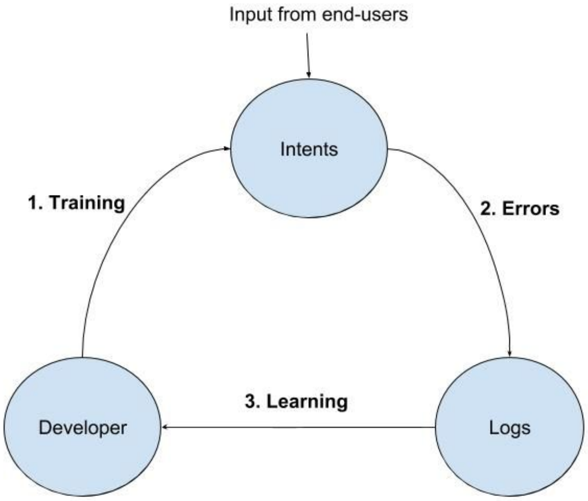


figure 3: Training of Natural Language Processing units

For NLP units DialogFlow, working and implementation are done in three steps:

Training: The first step is to figure out the correct intent from the user's input. These systems can do this function on their own, but they need to be trained beforehand. As developers, we figure out the types of the questions user can ask our chatbot. This information is also extracted from the dataset. Once question types are decided, we figure out the different ways a single question can be asked. We also figure out the possible entities for each type of intent. DialogFlow and Alexa Skill Kit have admin panels and UI where these configurations are done. They learn by themselves once this information is provided.

Errors: Even if we have provided many ways of asking a question, and systems have learned from that, the ways are not exhaustive. There will be many cases where the intent couldn't be figured out or the expected intent is not triggered based on the input given. These scenarios will be logged by the NLP services.

Learning: Developers look at the logs and figure out the inputs which were failing previously because of missing intents, and train the models again so that appropriate intent can be triggered next time when a question is asked in a similar way.

After this process, once the user provides a question in the input, an intent is triggered and relevant metadata has been extracted from the intent. This marks the end of the work done by the NLP units. Their next task is to provide the extracted input to next step. These services require that endpoint receiving this information must be running over HTTPS for the security reasons. Instead of setting up the custom server and getting the certificate for authority, AWS provides the service to execute the code on-demand known as AWS Lambda.

NLP units send structured JSON with intent, entities and other related information like the actual text of the question, language detected, unique session id etc.which can be used for the application to the configured endpoint of AWS Lambda. Following is an example of the request made by the NLP unit to backend server[3]:

POST https://url/of/AWS/Lambda

Headers:
Content-type: application/json

Request Body:

```
{
  "responseId": "ea3d77e8-ae27-41a4-9e1d-174bd461b68c",
  "session":
"projects/your-agents-project-id/agent/sessions/88d13aa8-2999-4f71-b233-39cbf3a824a
0",
  "queryResult": {
    "queryText": "user's original query to your agent",
    "parameters": {
      "param": "param value"
    },
    "allRequiredParamsPresent": true,
    "fulfillmentText": "Text defined in Dialogflow's console for the intent that
was matched",
    "fulfillmentMessages": [
      {
        "text": {
          "text": [
            "Text defined in Dialogflow's console for the intent that was matched"
          ]
        }
      }
    ],
    "outputContexts": [
      {
        "name":
"projects/your-agents-project-id/agent/sessions/88d13aa8-2999-4f71-b233-39cbf3a824a
0/contexts/generic",
        "lifespanCount": 5,
        "parameters": {
          "param": "param value"
        }
      }
    ],
    "intent": {
      "name":
"projects/your-agents-project-id/agent/intents/29bcd7f8-f717-4261-a8fd-2d3e451b8af8
",
      "displayName": "Matched Intent Name"
    },
    "intentDetectionConfidence": 1,
    "diagnosticInfo": {},
    "languageCode": "en"
  },
  "originalDetectIntentRequest": {}
}
```

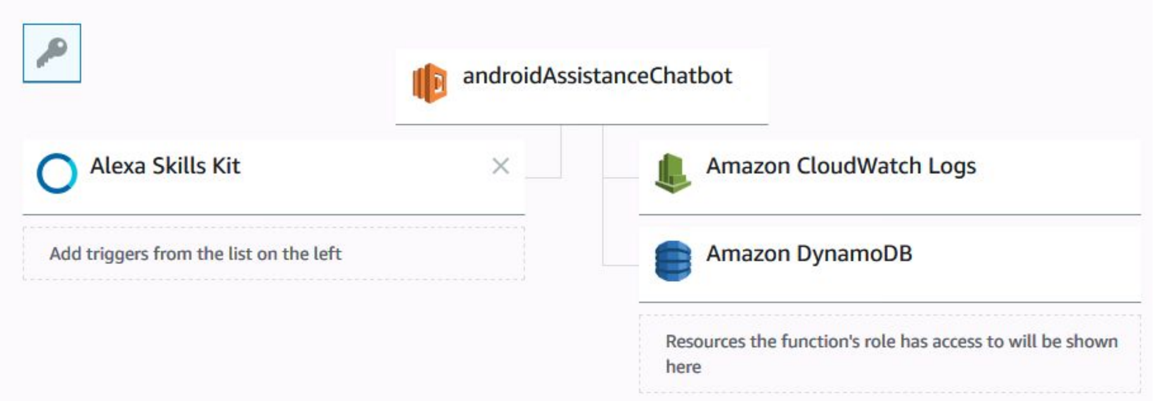
Serverless Processing

AWS Lambda is service provided by Amazon to run the custom code on the cloud without setting up any servers. It is an event-based system where the code is executed when a certain event occurs and triggers the execution. We don’t need to monitor the servers and only need to pay for the time taken for computation of the code. This service is also easily scalable and secure.

Purpose of using the serverless computing provided by AWS is the requirements by the NLP units. DialogFlow require that the backend service where they send the information must follow the rules provided by them. The important thing is they require the backend service to be running on HTTPS for security reasons.

Setting up the custom service over HTTPS needs a signed certificate from the certificate authority and requires certain configurations. AWS Lambda fulfills these requirements by default.

We are going to set up two different Lambda functions for DialogFlow and Alexa Skill Kit since they have different request and response formats. Following is the diagram of the Lambda function configured with Alexa Skill Kit:



Add triggers from the list on the left

Resources the function's role has access to will be shown here

figure 4: AWS Lambda configuration with Alexa Skill Kit

In the development of Lambda function, we will use the SDK provided by DialogFlow and Alexa Skill Kit so that we can focus on writing the application logic rather than the boilerplate code required for handling the POST requests made by the NLP units. The errors and information are logged in Amazon CloudWatch Logs. Once, we have the required information in the Lambda function, we will call the backend API to get the actual answer from the dataset.

Backend APIs and Database

This is another important component of the project. This component focuses on the storing and retrieving the data. Implementation of this part determines the success of the project. Queries need to be efficient enough to provide the accurate response within the acceptable time limit.

a. Datasource

We are using the StackExchange website of Android and other official android developer forum websites to get the hundreds of thousands of questions and answers related to the Android development. This datasource has been taken from these websites in XML and csv format. These files contain all the information like questions, all the answers, correct answer, tags and title. Upvotes are also provided in the datasource which can help to find out the most relevant answers. There are many open source pages which has information about android related questions and answers like developer uses code for any task or ui elements or exception information.

b. Data Preprocessing

Since the data from StackExchange and other official android websites has many questions which haven't been answered at all or the answers are too long then such questions need to be removed from the dataset. Such questions are not suitable for the chatbot application. We are going to write the Python script to find out such questions and removing them from the datasource.

c. MongoDB

Mongodb is NoSQL database used for storage of data in the form of documents. The data format for the chatbot matches the configuration of the mongodb. Node js integration and querying mongodb is very fast using json format data transfer and it supports horizontal scalability.

d. Deep Learning Technique

Using tensorflow and its libraries like tflearn, model can be trained. That model does the natural language processing and figures out important terms like intents and entities. From that answer can be searched. Additionally sequence to sequence model using RNN can be implemented and it doesn't need to figure out keywords. But the model is trained in such a way that model returns the best answer. In such cases, neural network, natural language processing and deep learning algorithms are very important.

e. REST APIs

These are the endpoints developed in Java Spring Boot and hosted in the cloud. AWS Lambda functions will call these APIs with appropriate parameters to get the answers to the questions. Our APIs will have mostly GET endpoints as we plan to only read the information from the database.

Chapter 3: Technology Description

This chapter provides brief description about the technologies that are used in the implementation of Assistant Chatbot for Android Developer. Key benefits and justification for using these technologies are also given.

3.1 Client-Tier Technologies

NodeJS

NodeJS is one of the most popular languages for developing the web-based applications. The main reason for choosing the NodeJS is that it is uniform for all the layers. We can use the same language in all of the three tiers thereby making it easy to understand and follow. It also allows the team to share the knowledge internally.

On the client-tier, NodeJS is useful because it can be easily integrated with other Javascript based frameworks and eliminates the need to write the server and client sides in different programming languages. NodeJS is an all JavaScript framework, it complements the other Bootstrap and jQuery frameworks used to design the interactive web client. Since we are using the DialogFlow in Middle-Tier, the NodeJS SDK provided by DialogFlow easily integrates with the web client. This also makes it easy to make web service calls to the DialogFlow rather than handling the network calls in the NodeJS explicitly.

HTML/CSS/jQuery

These are the standard technologies used for any web client development. Using HTML, CSS and jQuery, we design the interactive web based client using which users can ask questions to the chatbot. These technologies are supported by all the major web browsers.

Speech-To-Text

The cloud based service provided by Google to convert the speech to text is used to facilitate the benefit for end users to ask questions via audio as well. This service helps to convert the audio provided by the end user in the text format in real-time. This service enables to apply the powerful neural networks on the audio and get the transcript for the short audio in .wav format.

3.2 Middle-Tier Technologies

DialogFlow

DialogFlow is the Natural Language Processing and understanding (NLP) framework provided Google. It serves as the main component of the project. DialogFlow helps to detect the intent from the text provided by the end user. It also extracts the entities from the text provided. DialogFlow provides the better accuracy for the NLP as compared to a new NLP processor built from the scratch.

AWS Lambda

Serverless Processing provided by AWS Lambda is the recommended way to get the information processed by DialogFlow as it satisfies requirements for the security. DialogFlow requires the fulfilment service to be running on HTTPS with valid security certificate. AWS Lambda runs on HTTPS and eliminates the need to maintain the certificate. It works as a bridge between the third-party natural language processing services and custom-built backend server.

NodeJS

NodeJS is used in writing the functions in AWS Lambda. AWS Lambda provides the on-screen editor for NodeJS. We are using the DialogFlow SDK provided in NodeJS to receive the intents and entities extracted by the DialogFlow. Using the intents and entities, NodeJS functions call the backend API to get the response from the backend server.

3.3 Data-Tier Technologies

MongoDB

MongoDB is the document-based database with high performance on very large datasets. The mongoDB is used for storing the data of related questions and answers

NodeJS

NodeJS here is used to expose the APIs that AWS Lambda can call to fulfil the request from the backend. Here, the NodeJS Application serve dual purpose of exposing the APIs and connecting to the MongoDB server for data extraction.

Python

Python is a programming language and has very rich library set. Python can be used to make scripts and connect to the database. Using various libraries, data processing is very easy to do.

Chapter 4: Project Design

This chapter includes the UML diagrams such as sequence diagram, use case diagram, class diagram and entity diagram for the chatbot system

4.1 System Design

4.1.1 Sequence Diagram

Sequence diagram is shown below. There are many lifelines and actors shown in the diagram. All actors are user, component for UI react, request module, dialog flow, intent, entity in dialog flow and alexa skill kit, raw data, mongodb client and elasticSearch.

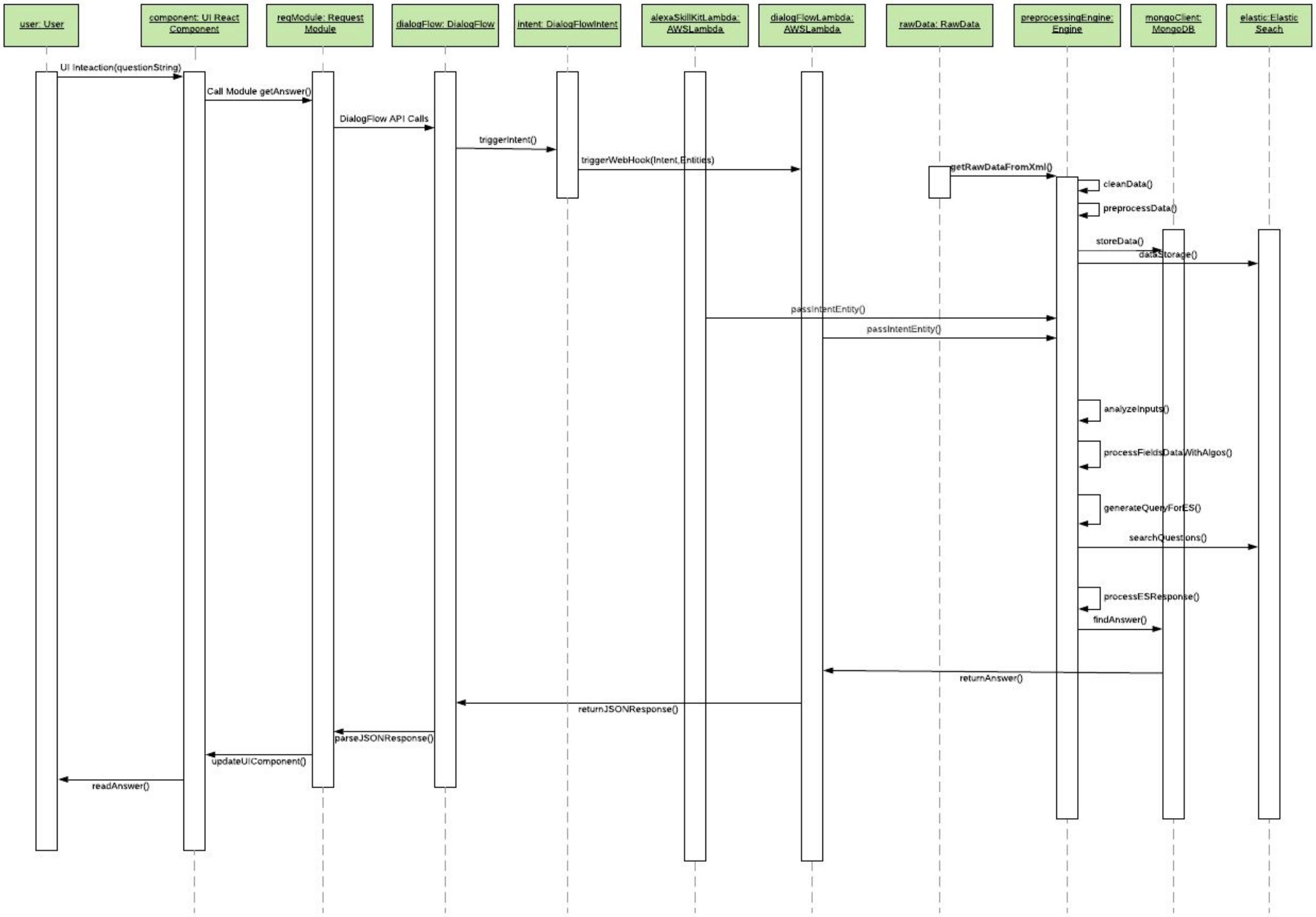


figure 5: Sequence Diagram

The sequence diagram shows the flow from user input to processing of input. After processing the input request and answer is returned. Steps are user interface interaction, calling module to get answer. Continuous steps are dialog flow api call, triggering event, use webhooks and many more. Parallely raw data is available in xml format and that is stored in database after preprocessing steps. Some actions are in the backend part. Those actions are passing entity, passing intent, analyze input, process fields with input data, generate query for the database. Search data in the elasticSearch for questions. Then processing the response.

4.1.2 Use case Diagram

There are total 2 use case diagrams shown below: Use case diagram is used for representing various use cases of the systems. Use case diagram is used for showing various functionality of the system.

Main components of the use case diagram are actor, use case, relation, boundary and many more. Actor can be seen as a user of the system. First use case diagram shows Basic use cases of the chat bot. There will be multiple use interfaces so interactions can include mobile app, web app. All are extending the interface use case. Other use case are text, voice case and extending ask question, get answer use cases. These use cases are connected to the chat bot user.

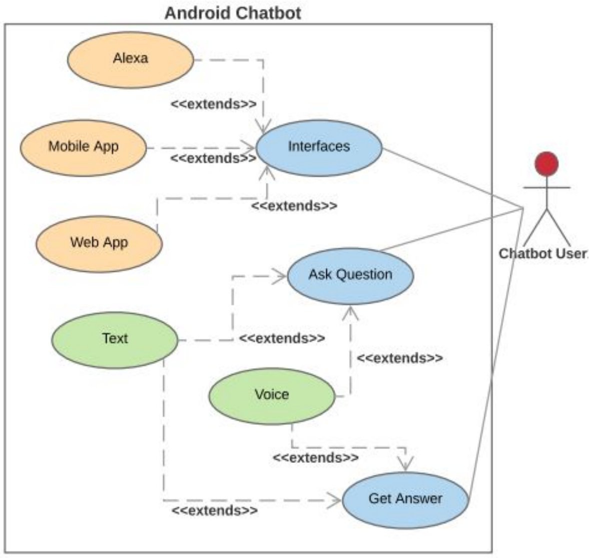


figure 6: Use Case Diagram

Below use case diagram is about various use cases which are performed internally. The dialog flow and Google Assistant services extend the natural language processing to figure out entity and intent. Amazon lambda, back end, preprocessing question answer data, storage are various use cases for data processing. Pick the best answer is also a use case. So, all these use cases belong to the processing engine and live in the boundary of android chat bot.

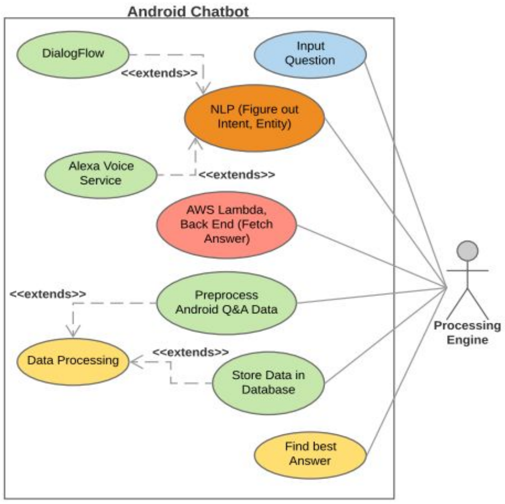


figure 7: Use Case Diagram

4.1.3 Class Diagram

In the class diagram, we generally depict the executable objects that are being used while we develop the application. The class diagram shows the objects and their relations, how they are inter related to each other. We can draw an idea of creating a class diagram by making an interpretation from the entity-relationship diagram. So, we can see that entities being used in the entity-relationship diagram have become the classes in the class diagram.Class diagram serves as a primary artifact in software development tools based on the idea of MDA (Pavlova & Nikiforova, 2009)

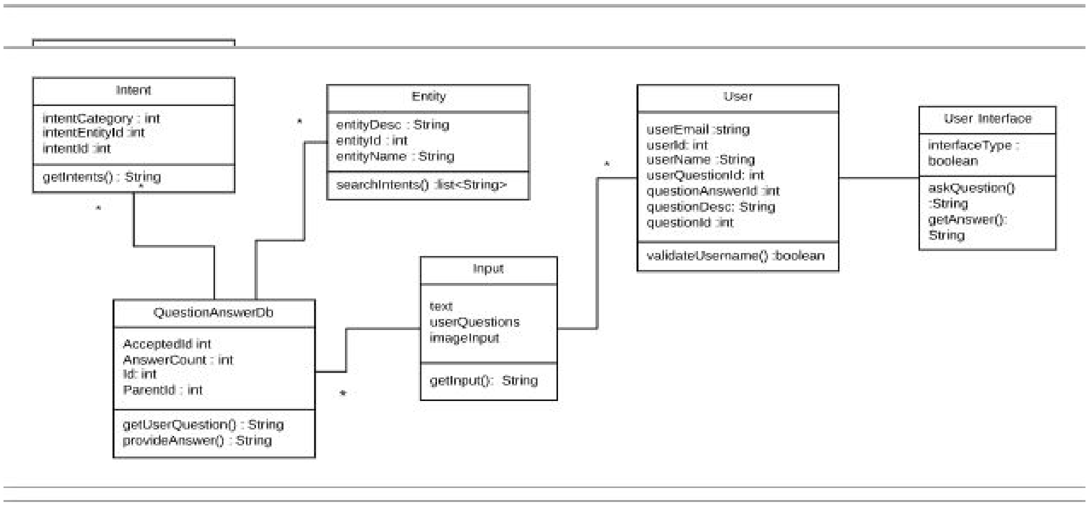
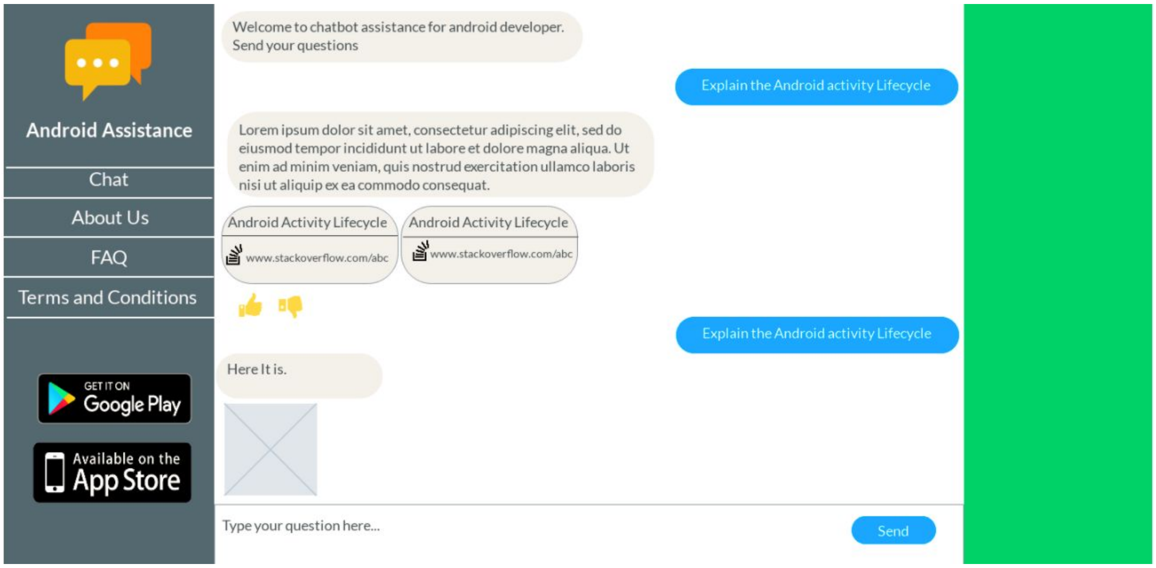


figure 8: Class Diagram

4.2 Client Design

4.2.1 Web Application

This is the mockup of the landing page of the web interface. This screen shows the textbox to send the message to the server. In the response, the web interface will show the text, hyperlinks, and images returned from the server. We also ask the user to provide the feedback for answers with thumbs up/down symbols.



Welcome to chatbot assistance for android developer. Send your questions

Explain the Android activity Lifecycle

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Android Activity Lifecycle www.stackoverflow.com/abc

Android Activity Lifecycle www.stackoverflow.com/abc

Explain the Android activity Lifecycle

Here It is.

Type your question here...

figure 9: Web Interface Mockup

4.3 Data-Tier Design

4.3.1 Entity Diagram

We are using MongoDB NoSQL database to store the data downloaded and extracted by the website crawler. Considering the huge amount of data being crawled and downloaded, we have used MongoDB noSql database to provide fast and efficient querying of data. MongoDb is a database that provides scalability and flexibility. It integrates well with large scale web applications.

Below is the entity and mapping diagram for MongoDB collections also known as the entities. We are storing the main details of the crawled data (which has the questions and answers from the stack exchange site) in QuestionsAnswersDb. This entity is being mapped to other entities based on our system design requirement. The “Entity” and “Intent” are the other two important tables which are going to be used extensively and particularly for mapping the intent fetched from the question asked in the chatbot by the user to the correct entity and then to a correct question to get the correct response from QuestionsAnswersDb.

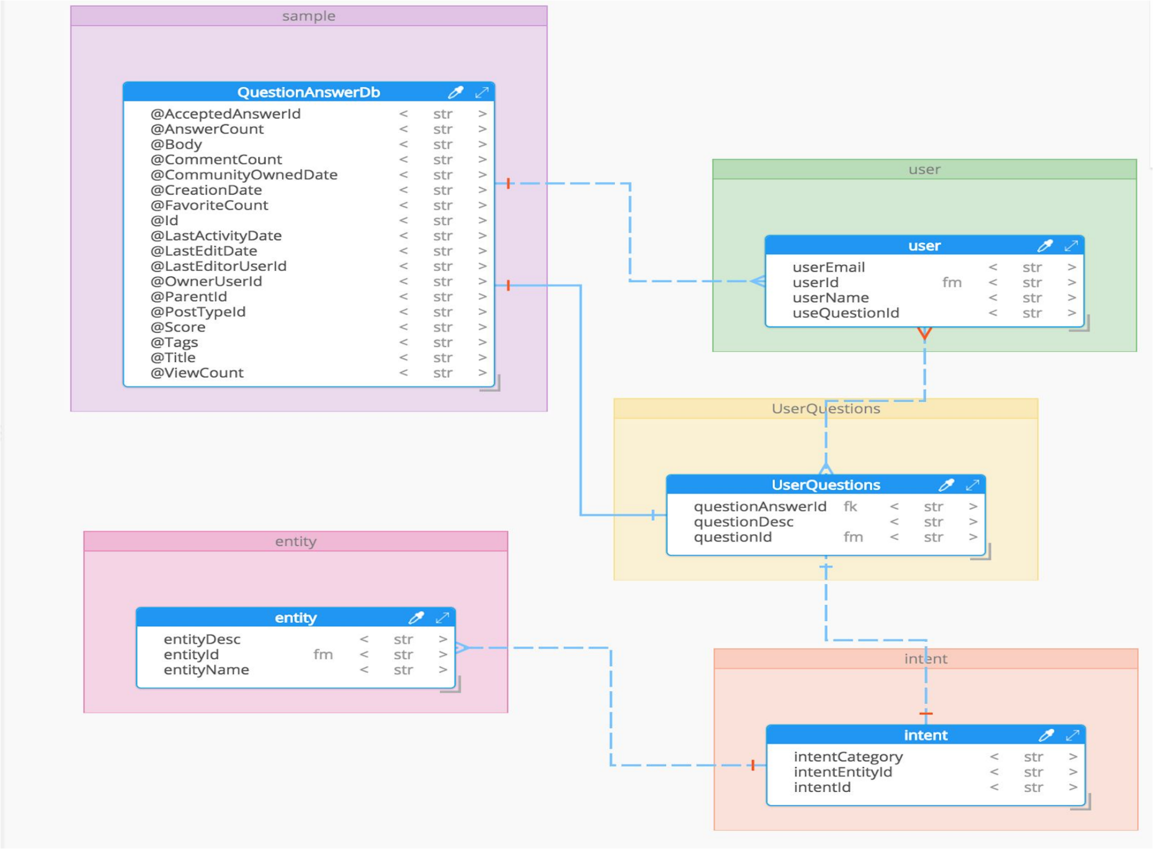


figure 10: Entity Diagram

Chapter 5 : Project Implementation

5.1 Client Implementation

We have implemented the web client application which connects with the backend server via the DialogFlow and AWS Lambda which fetches the response from the backend by making restful web service calls and delegates it back to the client chatbot interface.

Following are the key requirements common to both the web application that we fulfilled:

- Users are able to ask questions in text format.
- The application provides the response by the server in the text format.
- The application has help/FAQ page describing how to interact with the application.
- The application will handle the error scenarios like long response time, unavailability of a server, invalid formats of input etc. in a user-friendly manner.

Development Environment:

Following are the key components we used in the development environment:

- NodeJS 8.11.1 LTS
- Express 4.16
- npm 5.8
- Babel
- Webpack package

Following are the main components of the client-side application developed on NodeJS:

package.json:
It is a manifest file of the project. This file stores the information about the external packages and their versions. It helps the Node Package Manager (NPM) to identify that a project has a dependencies on these modules and it configures the packages listed there prior to execute the project.

app.js:
It is an entry point of the project and it initializes the required module of the project so that it keeps everything glued together for the project.

/routes/:
This folder contains the routing modules which maps the web service requests to the functions implemented.

/views/:
This folder contains the front-end views which are displayed to the end user. We are using ejs template language for the front-end views.

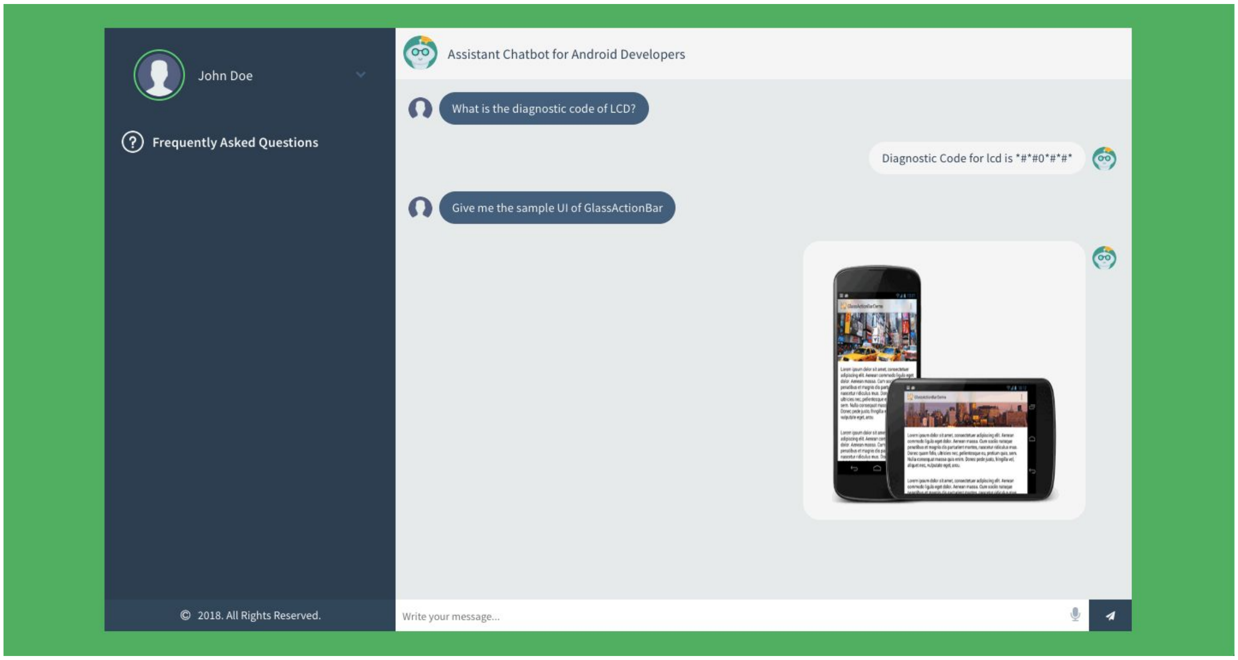
/public/:
This folder contains the static resources like images, css, and javascripts which are delivered to the end user along with the views.

Logical breakdown of tasks we performed to develop the front-end application:

1. User Interface Design:

a. We prepared the UI for a Web application client using a bootstrap template available under MIT license for designing a chat interface. The bootstrap template uses HTML, CSS, JavaScript and Jquery to perform the necessary actions. We have modified the basic HTML provided in the template to accommodate our requirements such as removing the unnecessary HTML components and modifying the CSS to make it look like shown in the wireframe.

Following is the screenshot of the Web client:


John Doe
Frequently Asked Questions
© 2018. All Rights Reserved.
Assistant Chatbot for Android Developers
Write your message...

What is the diagnostic code of LCD?

Diagnostic Code for lcd is *#*#0*#*#*

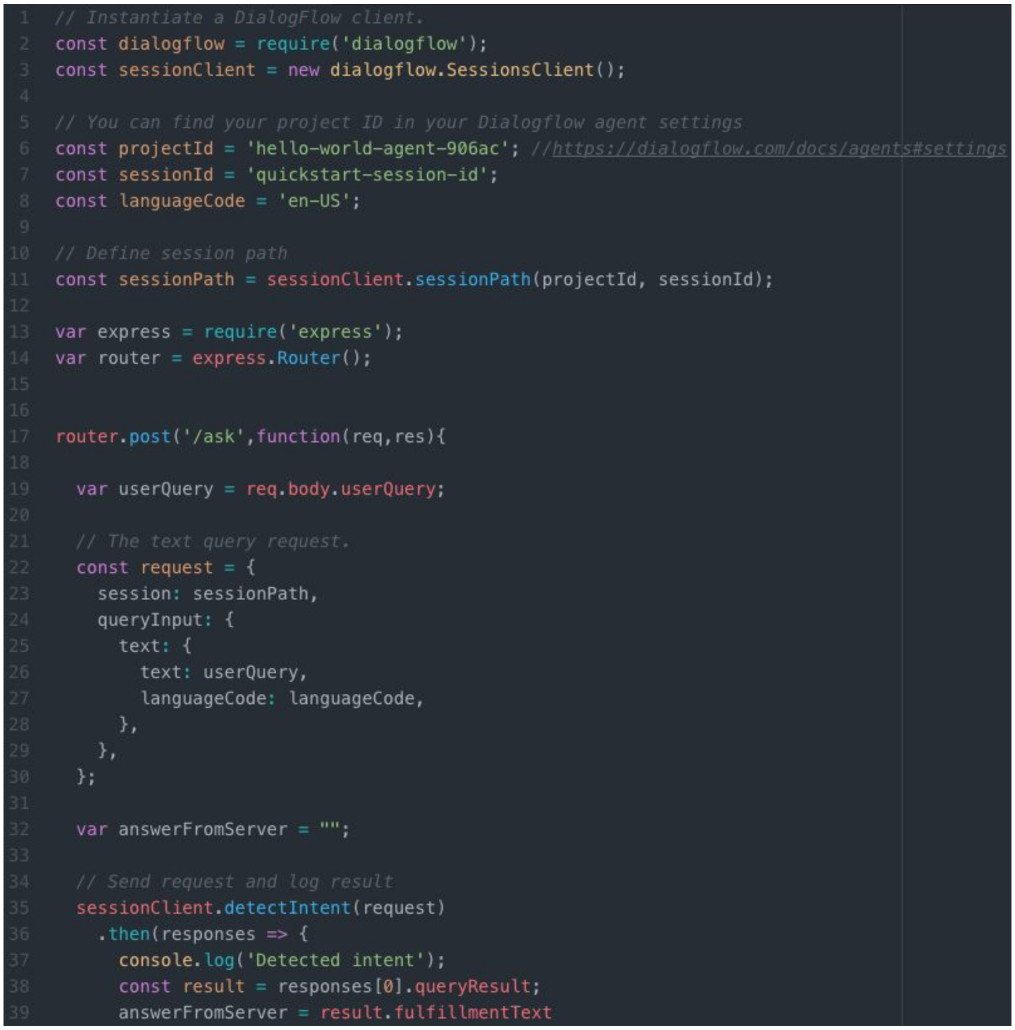
Give me the sample UI of GlassActionBar

figure 11: Web Client Interface

2. User Interface Preparation with Static Data:
a. Prepared the browser supported web user interface for the chatbot.

3. Integration with the DialogFlow APIs:
a. Integrated the DialogFlow Node JS API to make calls to the DialogFlow and get the response.

The following code snippet shows the DialogFlow configurations and request to be sent the DialogFlow using the NodeJS client SDK:



```
// Instantiate a DialogFlow client.
const dialogflow = require('dialogflow');
const sessionClient = new dialogflow.SessionsClient();

// You can find your project ID in your Dialogflow agent settings
const projectId = 'hello-world-agent-906ac'; //https://dialogflow.com/docs/agents#settings
const sessionId = 'quickstart-session-id';
const languageCode = 'en-US';

// Define session path
const sessionPath = sessionClient.sessionPath(projectId, sessionId);

var express = require('express');
var router = express.Router();


router.post('/ask',function(req,res){

  var userQuery = req.body.userQuery;

  // The text query request.
  const request = {
    session: sessionPath,
    queryInput: {
      text: {
        text: userQuery,
        languageCode: languageCode,
      },
    },
  };

  var answerFromServer = "";

  // Send request and log result
  sessionClient.detectIntent(request)
    .then(responses => {
      console.log('Detected intent');
      const result = responses[0].queryResult;
      answerFromServer = result.fulfillmentText
```

b. Dynamically updated the user interface with response data.
c. Provided support for the different types of responses like images and hyperlinks.

4. Voice input support on web application:

In addition to providing the input via text, we added the support of voice input on the web application. DialogFlow extracts the intents and entities from the audio and provides the

response back to the application. We are using DialogFlow SDK methods that takes an audio file as a parameter. Internally, DialogFlow uses the Speech-To-Text API to transcribe the audio and then detect the intents from it.
As on the browser side, the client application takes the microphone permission from the user and records the audio. It then converts it into .wav format with sample rate of 44100 Hz and sends it back to the NodeJS backend. The backend saves the file on the disk and uploads it to the DialogFlow. Upon successful upload, the file is removed from the disk and the app will behave similar to the text input.

5. Error handling:
a. Implemented code logic for handling improper input data by showing proper client side error response understandable by the user.

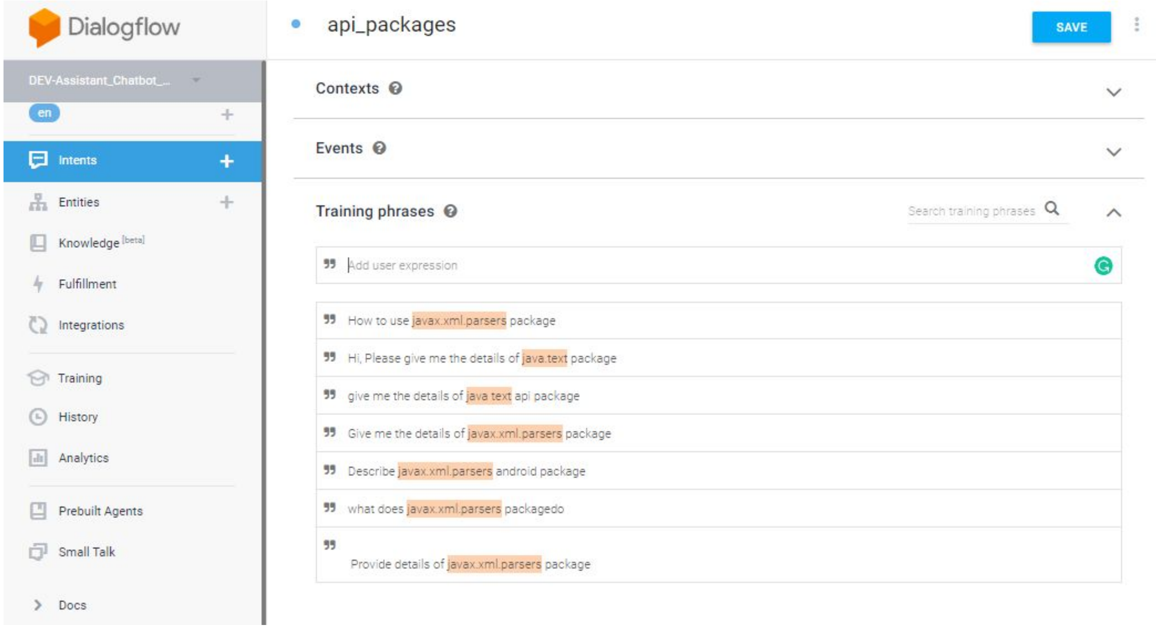
6. Testing and Bug fixing:
a. Developed and executed the unit and integration test cases of the containers for the application.
b. Identified the bugs and fixed them.

5.2 Middle-Tier Implementation

In the middle-tier we have our DialogFlow api which performs the ML part and also the AWS Lambda serverless processing of the business logic to fetch the query and fetch the response from the backend server api. We created our account in the Google DialogFlow and created the agents, intents and entities based on the QA dataset present on our backend mongodb server.

Below are the snippets and images showing the implementation of the DialogFlow agent creation and also the creation of intents, entities with proper webhook to call the AWS lambda to process the business logic. Implementation of DialogFlow agent setup with Intents and training phrase to fetch results of android device based queries.

As shown in the figure below, we have setup intents and entities in the following fashion. We have created a package for the various api and the trained the model to ask questions as shown in the below figure. We also created the entities associated with the intents. The training of the chatbot is done using the DialogFlow.



api_packages

Training phrases

Add user expression

How to use javax.xml.parsers package

Hi, Please give me the details of java.text package

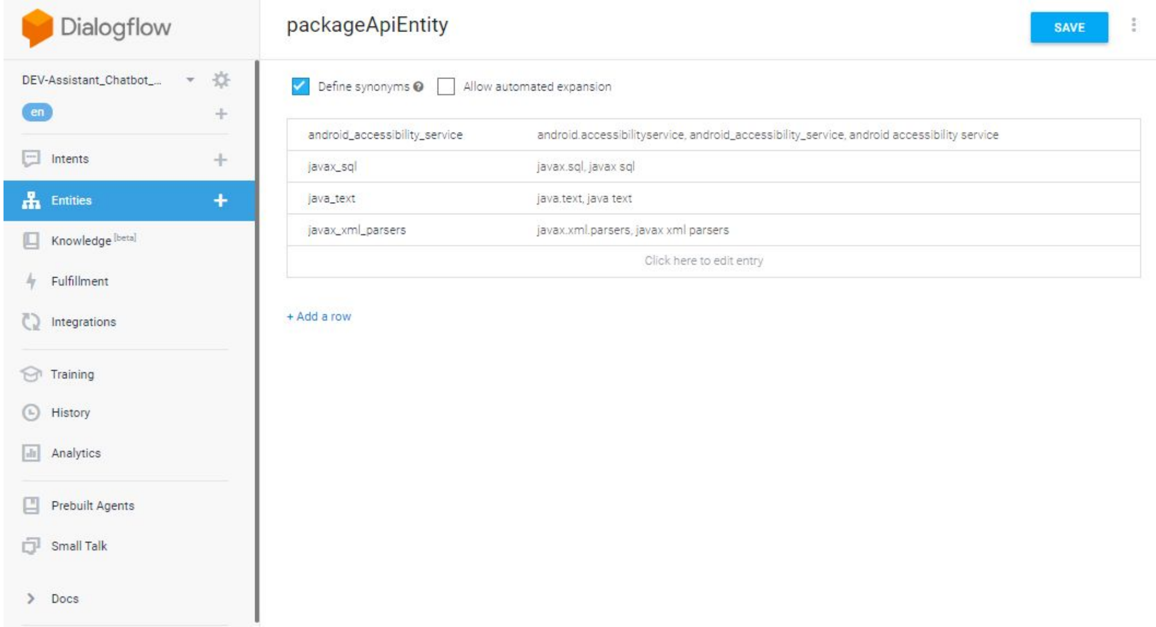
give me the details of java text api package

Give me the details of javax.xml.parsers package

Describe javax.xml.parsers android package

what does javax.xml.parsers packagedo

Provide details of javax.xml.parsers package

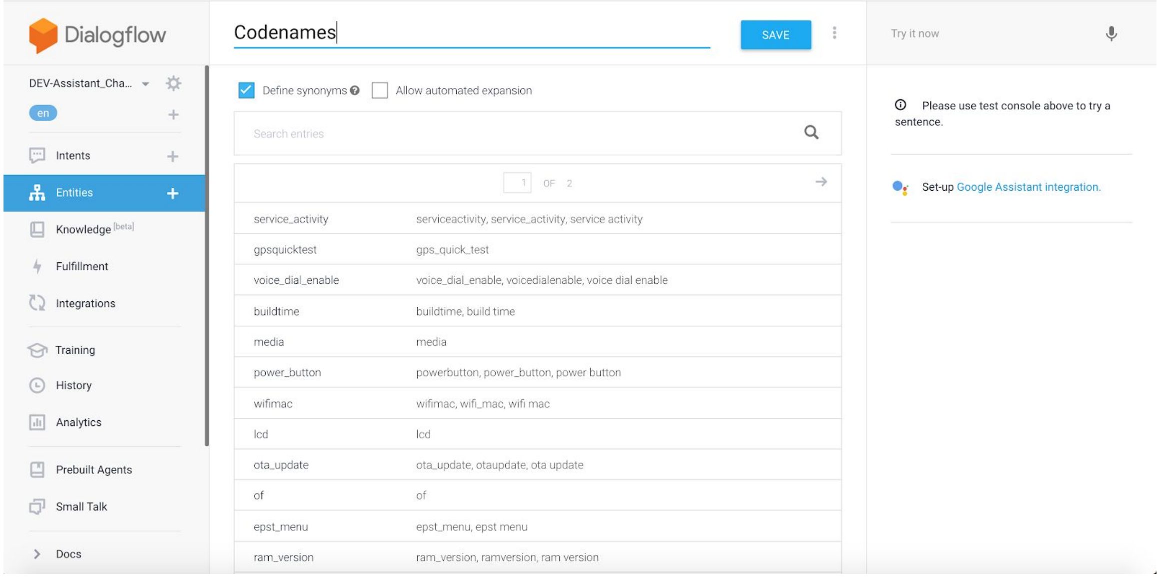


packageApiEntity

☑ Define synonyms ☐ Allow automated expansion

android_accessibility_service	android.accessibilityservice, android_accessibility_service, android accessibility service
javax_sql	javax.sql, javax sql
java_text	java.text, java text
javax_xml_parsers	javax.xml.parsers, javax xml parsers
	Click here to edit entry

+ Add a row



Codenames

☑ Define synonyms ☐ Allow automated expansion

service_activity	serviceactivity, service_activity, service activity
gpsquicktest	gps_quick_test
voice_dial_enable	voice_dial_enable, voicedialenable, voice dial enable
buildtime	buildtime, build time
media	media
power_button	powerbutton, power_button, power button
wifimac	wifimac, wifi_mac, wifi mac
lcd	lcd
ota_update	ota_update, otaupdate, ota update
of	of
epst_menu	epst_menu, epst menu
ram_version	ram_version, ramversion, ram version

Please use test console above to try a sentence.

Set-up Google Assistant integration.

figure 12: Dialogflow Agent, Intent, Entity Creation

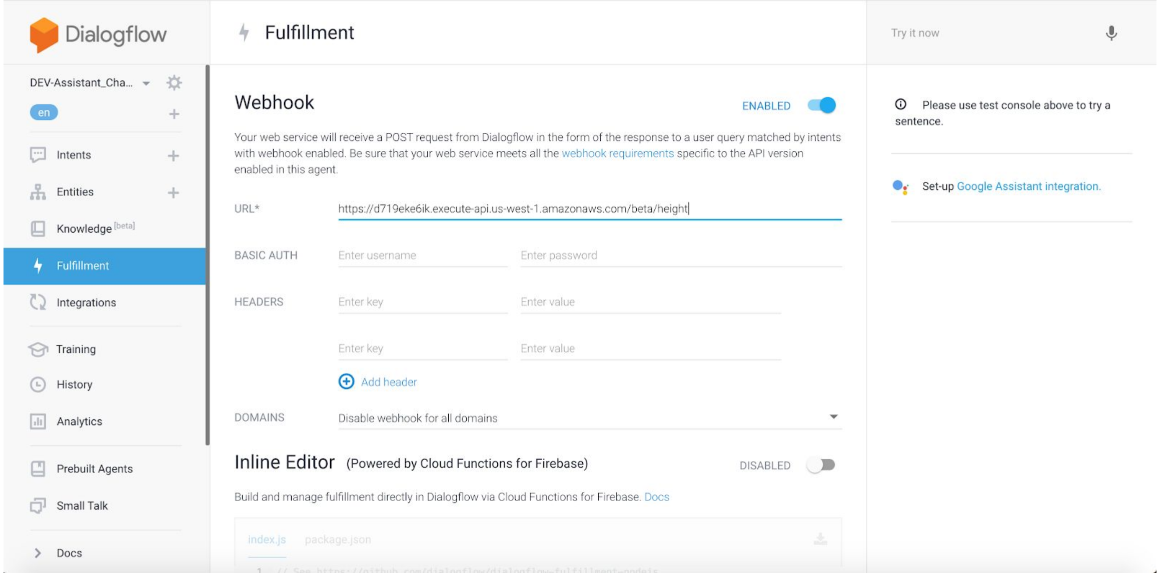
In this project we used the dialogue flow for the conversation. We have set the intents, entities and training in the following manner:

INTENT	ENTITIES	TRAINING	RESPONSE
api_classes	zoomcontrols zoombuttonscontroller xpathfunctionexception	Give me the details of ZoomControl class How to use WifiP2pManager ConnectionInfoListener interface Provide details of ZoomControl class What is the use of VolumeShaper.Configuration class Describe ZoomControls	The Zoom Controls class displays a simple set of controls used for zooming and provides callbacks to register for events.

		android class	
keyboard_shortcuts	install android_studio Meet Android Studio emulator Emulator Fast Emulator Projects	What is the keyboard shortcut for Synchronize Hi, Could you please provide the keyboard shortcut for Synchronize on Windows keyboardShortcut for Synchronize MacOs keyboard shortcut for Synchronize Windows Hi, Could you please provide the keyboard shortcut for Synchronize on MacOs	Command + Option + Y Command + Option + z Command + Option + R
studio	Fast emulator Android App BundleAndroid Kotlin Developer workflow	What is Emulator Describe the feature of visual layout editor Describe installation Process of installation of emulator	Install and run your apps faster than with a physical device and simulate different configurations and features, including ARCore, Google's platform for building augmented reality experiences. Click <a href=https://develop

			er.android.com/studio/run/emulator target="_blank">here</a> for more information.
api_packages	android_accessibility_service javax_sql javax_xml_parsers java_text	How to use javax.xml.parsers package Hi, Please give me the details of java.text package give me the details of java text api package Give me the details of javax.xml.parsers package	https://developer.android.com/reference/javax/xml/parsers/package-summary.html
image	ui_element image GlassActionBar	what is the ui element of GlassActionBar give me the image of GlassActionBar give me the sample of GlassActionBar ui_element	5bf8b6436a84f550230c1b1b It will return the Image of the UI elements
basicsEntity	theme manifest Android Architecture languages_support	what is content_provider in android? what is a view group in android? what is content_provider in android?	A content provider is used to share information between Android applications.

We setup the webhook here for the request to be further delegated to the AWS Lambda api to process the intent and entity that our dialog flow api extracted from the user query text.



Fulfillment

Webhook ENABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the webhook requirements specific to the API version enabled in this agent.

URL* https://d719eke6ik.execute-api.us-west-1.amazonaws.com/beta/height

BASIC AUTH Enter username Enter password

HEADERS Enter key Enter value

Enter key Enter value

Add header

DOMAINS Disable webhook for all domains

Inline Editor (Powered by Cloud Functions for Firebase) DISABLED

Build and manage fulfillment directly in Dialogflow via Cloud Functions for Firebase. Docs

index.js package.json

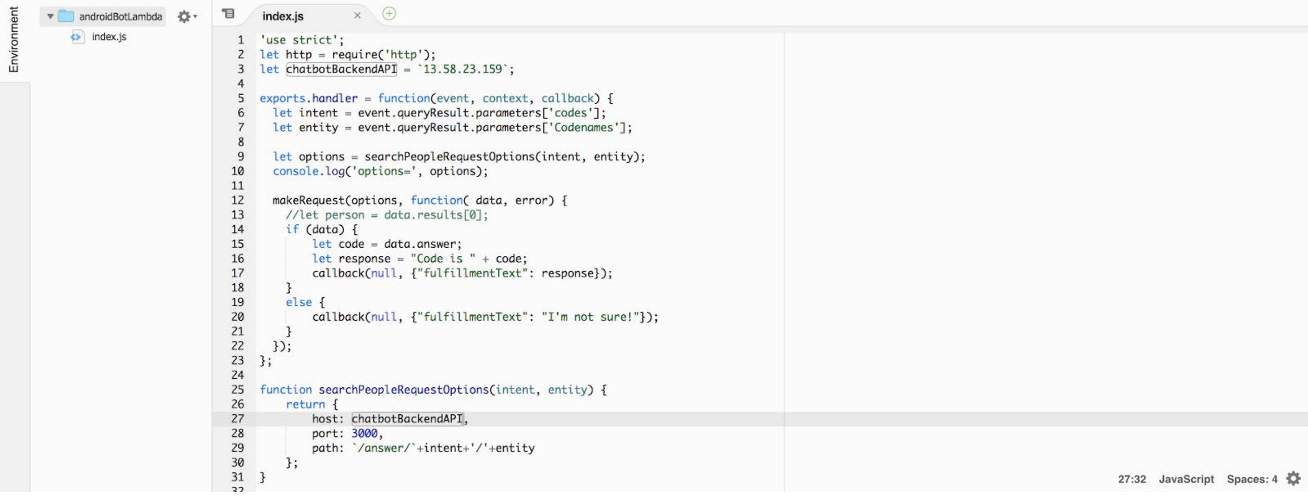
Please use test console above to try a sentence.

Set-up Google Assistant integration.

figure 13: Dialogflow Webhook Implementation

Once we did all these set up and implementation of the intents, entities in our dialogflow we moved to the AWS Lambda serverless api to process our business logic and fetch the response based on the intents and entities passed from the dialogflow api.

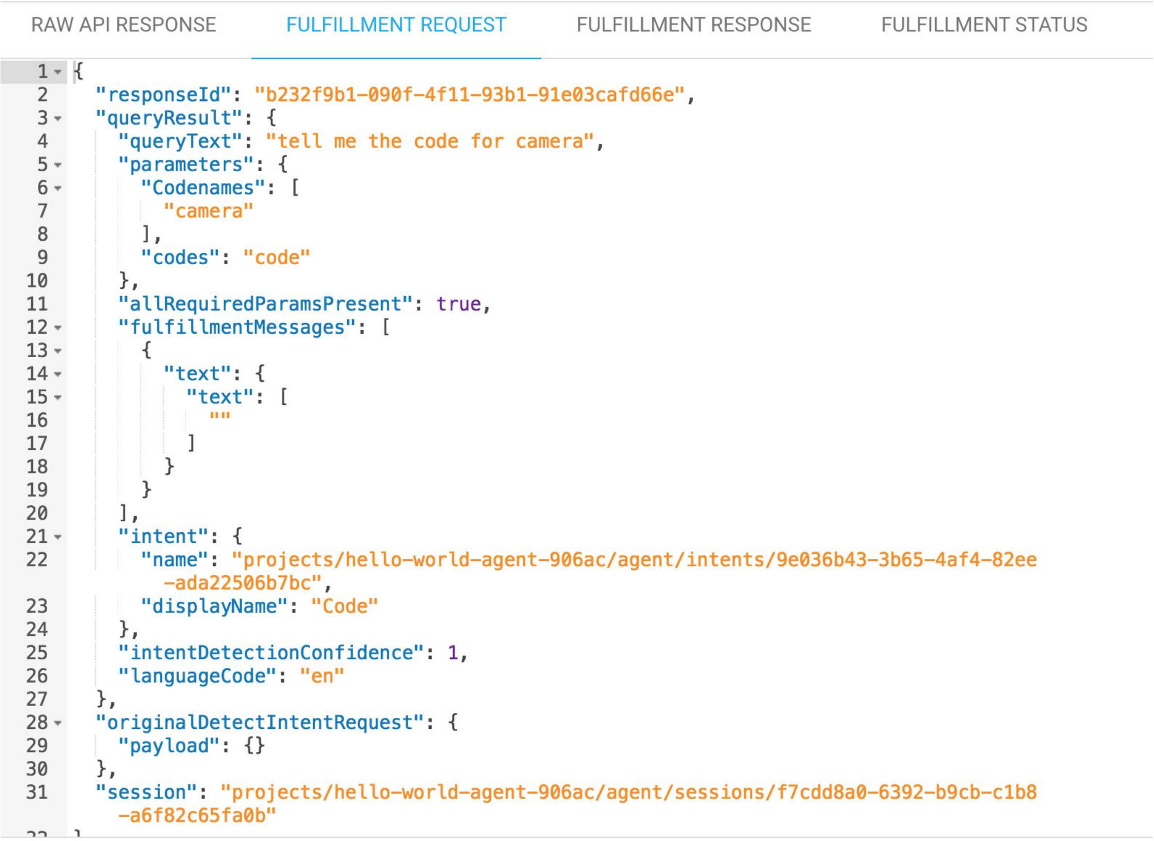
Below is the code snippet of the lambda api created using node js language and deployed on the AWS for fast response and better scalability. The below code snippet prepares a connection with the backend server restful web service api and passes the intents, entities to the server to fetch the user query response in return.



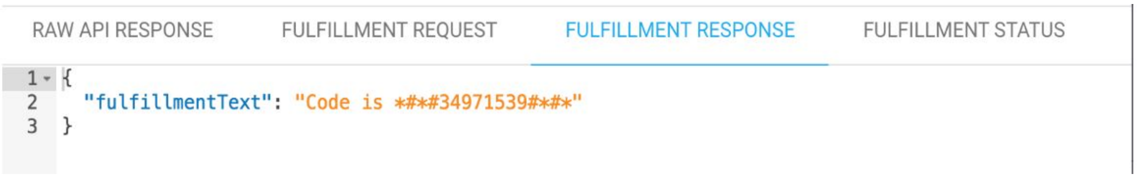
```
'use strict';
let http = require('http');
let chatbotBackendAPI = `13.58.23.159`;

exports.handler = function(event, context, callback) {
  let intent = event.queryResult.parameters['codes'];
  let entity = event.queryResult.parameters['Codenames'];

  let options = searchPeopleRequestOptions(intent, entity);
  console.log('options=', options);

  makeRequest(options, function( data, error) {
    //let person = data.results[0];
    if (data) {
        let code = data.answer;
        let response = "Code is " + code;
        callback(null, {"fulfillmentText": response});
    }
    else {
        callback(null, {"fulfillmentText": "I'm not sure!"});
    }
  });
};

function searchPeopleRequestOptions(intent, entity) {
    return {
        host: chatbotBackendAPI,
        port: 3000,
        path: `/answer/`+intent+'/'+entity
    };
}
```

figure 14: AWS Lambda implementation

The diagnostic info shows us the proper request parameters that are required by the dialog flow api client request and response parameters coming from the lambda api. This helped to better configure our intent and entity in the dialog flow model to get proper response.



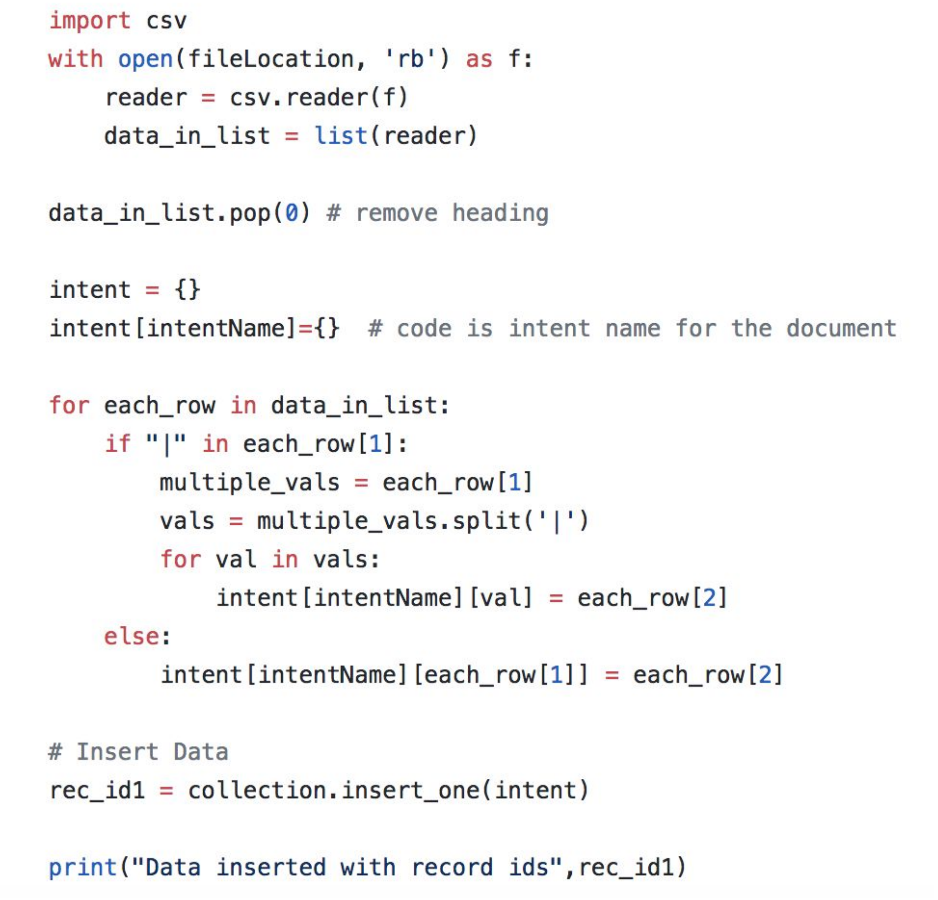
```
{
  "responseId": "b232f9b1-090f-4f11-93b1-91e03cafd66e",
  "queryResult": {
    "queryText": "tell me the code for camera",
    "parameters": {
      "Codenames": [
        "camera"
      ],
      "codes": "code"
    },
    "allRequiredParamsPresent": true,
    "fulfillmentMessages": [
      {
        "text": {
          "text": [
            ""
          ]
        }
      }
    ],
    "intent": {
      "name": "projects/hello-world-agent-906ac/agent/intents/9e036b43-3b65-4af4-82ee
        -ada22506b7bc",
      "displayName": "Code"
    },
    "intentDetectionConfidence": 1,
    "languageCode": "en"
  },
  "originalDetectIntentRequest": {
    "payload": {}
  },
  "session": "projects/hello-world-agent-906ac/agent/sessions/f7cdd8a0-6392-b9cb-c1b8
    -a6f82c65fa0b"
}
```



```
{
  "fulfillmentText": "Code is *#*#34971539#*#*"
}
```

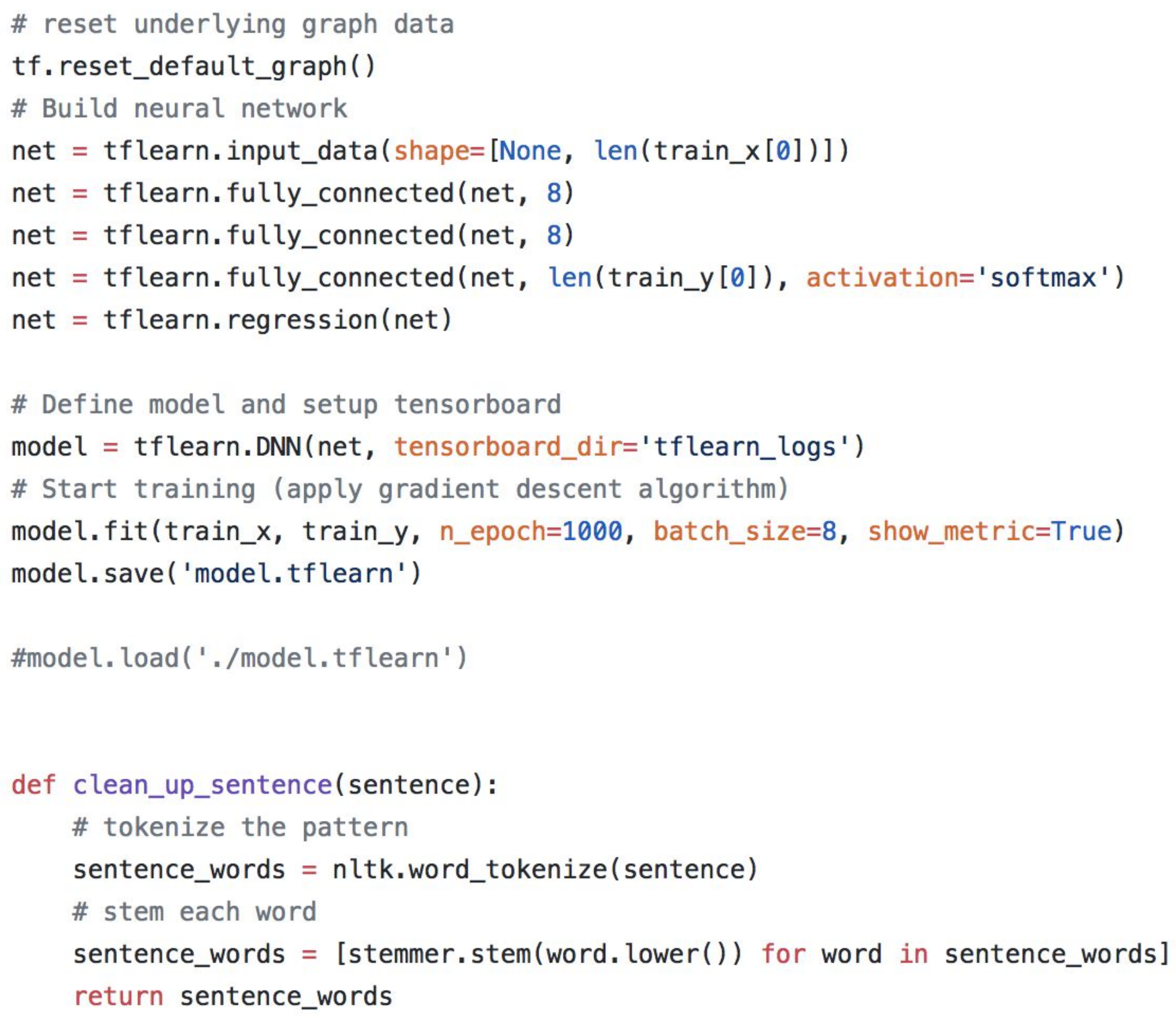
5.3 Data-Tier Implementation

The main purpose of the backend is to provide accurate answer of the question. Middle tier and data tier are linked to each other. In the backend REST API's are implemented in node js. AWS lambda from the middle tier directly calls the API and fetches the result. Internally in the data tier, it is composed of database, input data processing and finding the result. Mongodb database is used to store the data as documents. Android question answers for developers are available on stack exchange and other popular websites. The focus is to provide answers of android domain with specific subjects.



```
import csv
with open(fileLocation, 'rb') as f:
    reader = csv.reader(f)
    data_in_list = list(reader)

data_in_list.pop(0) # remove heading

intent = {}
intent[intentName]={}  # code is intent name for the document

for each_row in data_in_list:
    if "|" in each_row[1]:
        multiple_vals = each_row[1]
        vals = multiple_vals.split('|')
        for val in vals:
            intent[intentName][val] = each_row[2]
    else:
        intent[intentName][each_row[1]] = each_row[2]

# Insert Data
rec_id1 = collection.insert_one(intent)

print("Data inserted with record ids",rec_id1)
```

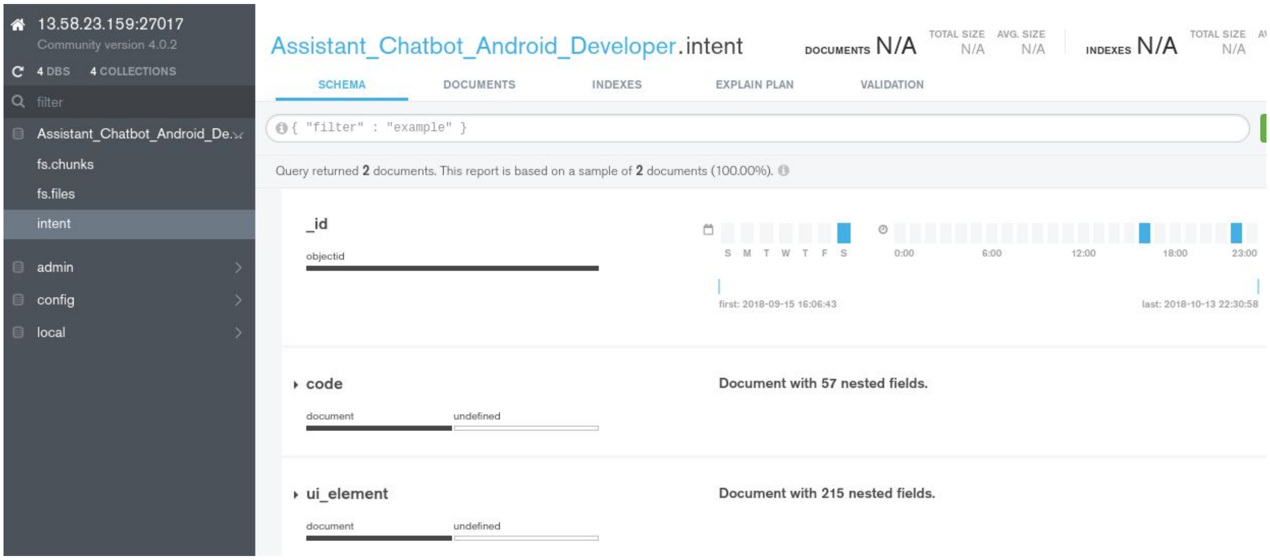
First step was to do data preprocessing, data cleaning and forming data to store in the database. That scripts are written in python and from the csv files, data is fetched and scripts cleans the data and stores in the Mongodb. There are many intents and under each intents multiple entities are stored. For each entity the answer is stored in the database. As per the input intent and entity proper answer is fetched from the database and returned. Images are also stored in the mongodb using gridfs. It internally stores images in pieces and uses id for image uniqueness.This way get api is implemented using node js. Database and backend server is deployed on the amazon cloud ec2 instance.



```
# reset underlying graph data
tf.reset_default_graph()
# Build neural network
net = tflearn.input_data(shape=[None, len(train_x[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(train_y[0]), activation='softmax')
net = tflearn.regression(net)

# Define model and setup tensorboard
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=8, show_metric=True)
model.save('model.tflearn')

#model.load('./model.tflearn')


def clean_up_sentence(sentence):
    # tokenize the pattern
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word
    sentence_words = [stemmer.stem(word.lower()) for word in sentence_words]
    return sentence_words
```

Some modules are implemented in such a way that dialog flow api identifies intent and entities using machine learning and deep learning algorithms. Some modules are implemented using tensorflow, tflearn and deep learning libraries for model to identify intent and entity. In this pattern, answer is fetched from the database. This way whole processing in the data tier works and data flows from amazon lambda to the api to the database and returned back to the lambda.

Below is the mongodb database screenshot. It shows how the data is stored as documents. Main components are intents and entities. Below image shows mongodb, which has database setup and many intents like code, ui_element, many more.



Assistant_Chatbot_Android_Developer.intent

{ "filter" : "example" }

Query returned **2** documents. This report is based on a sample of **2** documents (100.00%).

_id
objectid

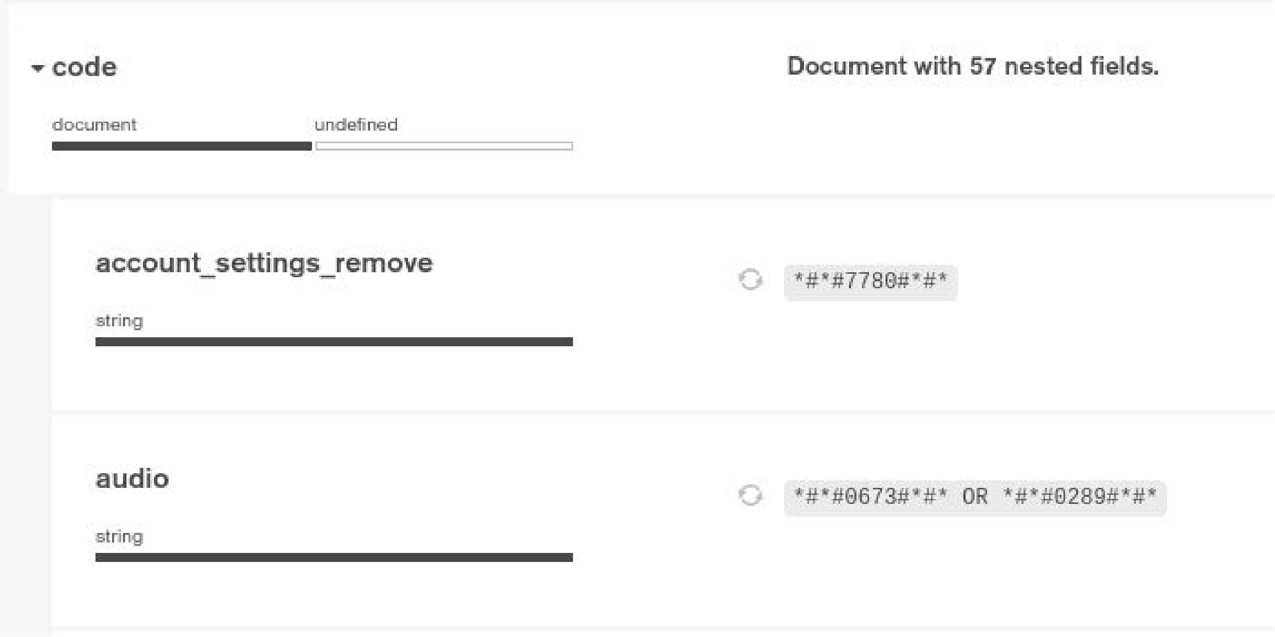
first: 2018-09-15 16:06:43
last: 2018-10-13 22:30:58

code
document undefined
Document with 57 nested fields.

ui_element
document undefined
Document with 215 nested fields.

figure 15: mongoDB implementation

This image shows that how multiple entities are stored in the intent to fetch the answer. Each intent has many entities and each entity has an answer. That is storage pattern.

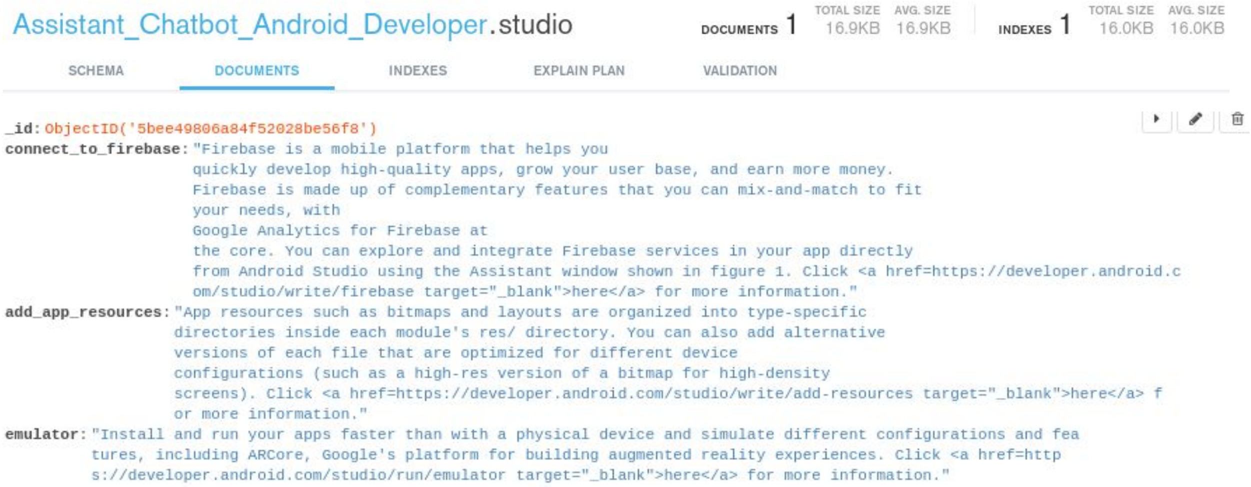


code
document undefined
Document with 57 nested fields.

account_settings_remove
string
##7780#*#*

audio
string
##0673#*#* OR *#*#0289#*#*

figure 16: intent, entity storage in mongoDB

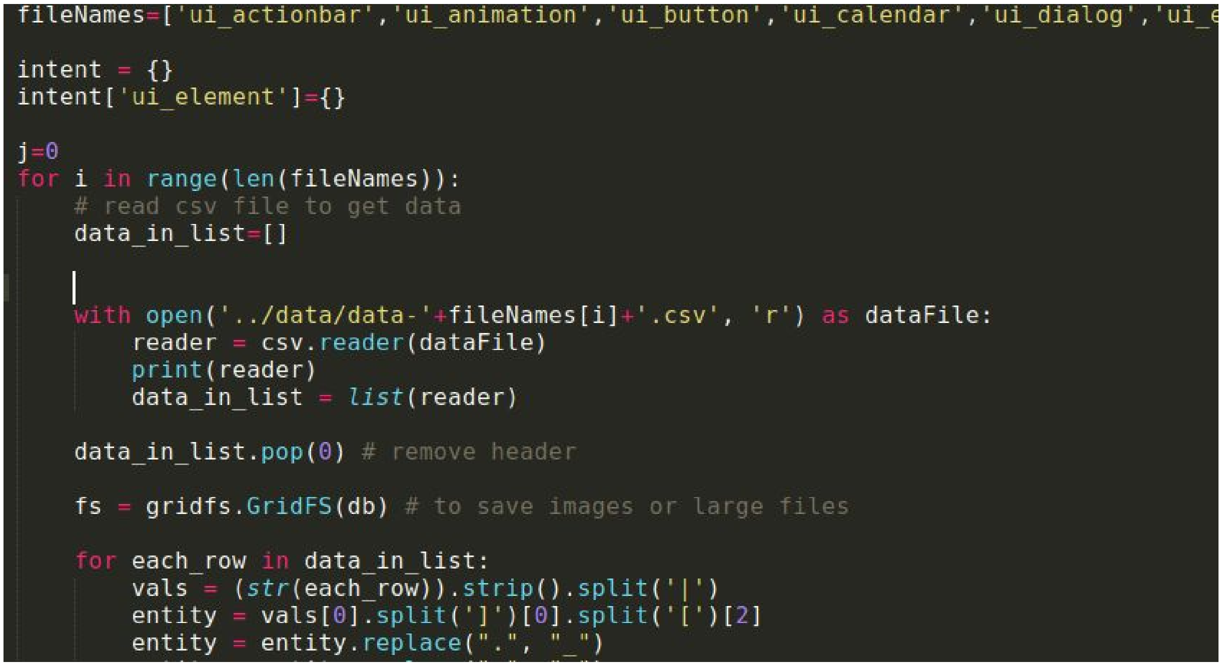


```
_id: ObjectID('5bee49806a84f52028be56f8')
connect_to_firebase: "Firebase is a mobile platform that helps you
                     quickly develop high-quality apps, grow your user base, and earn more money.
                     Firebase is made up of complementary features that you can mix-and-match to fit
                     your needs, with
                     Google Analytics for Firebase at
                     the core. You can explore and integrate Firebase services in your app directly
                     from Android Studio using the Assistant window shown in figure 1. Click <a href=https://developer.android.c
                     om/studio/write/firebase target="_blank">here</a> for more information."
add_app_resources: "App resources such as bitmaps and layouts are organized into type-specific
                   directories inside each module's res/ directory. You can also add alternative
                   versions of each file that are optimized for different device
                   configurations (such as a high-res version of a bitmap for high-density
                   screens). Click <a href=https://developer.android.com/studio/write/add-resources target="_blank">here</a> f
                   or more information."
emulator: "Install and run your apps faster than with a physical device and simulate different configurations and fea
          tures, including ARCore, Google's platform for building augmented reality experiences. Click <a href=http
          s://developer.android.com/studio/run/emulator target="_blank">here</a> for more information."
```

figure 17: storage pattern in mongoDB

Python

Python scripts are used to insert the data from raw files. Data is structured after fetching data from the raw files and then inserted in the database with proper format for faster search results. Data is in different format and to handle and store it various types of scripts are written.



```
fileNames=['ui_actionbar','ui_animation','ui_button','ui_calendar','ui_dialog','ui_e

intent = {}
intent['ui_element']={}

j=0
for i in range(len(fileNames)):
    # read csv file to get data
    data_in_list=[]

    with open('../data/data-'+fileNames[i]+'.csv', 'r') as dataFile:
        reader = csv.reader(dataFile)
        print(reader)
        data_in_list = list(reader)

    data_in_list.pop(0) # remove header

    fs = gridfs.GridFS(db) # to save images or large files

    for each_row in data_in_list:
        vals = (str(each_row)).strip().split('|')
        entity = vals[0].split(']')[0].split('[')[2]
        entity = entity.replace(".", "_")
```

Chapter 6: Testing and Verification

This chapter describes the testing efforts which has been made so far to test the development of the chatbot. We are mainly focused on the common test scenarios and not testing the edge cases which are also complex to test.

Before carrying out the functionality testing, we made sure the AWS lambda was working properly and was able to process the intents and execute the logic as expected. For making sure this happened properly we took a general approach which is described as below:

1. See what the Lambda function received.
2. Convert that information into valid JSON.
3. Paste the JSON into the “Input test event” window. This can be done by using the util library to console log the entire event invoking the Lambda.
4. Then run the “aws lambda invoke” command through lambda CLI to invoke that particular function

After doing this unit testing of lambda function we integrated the lambda api with dialogflow and also the backend restful api and then performed the end to end testing by interacting with our chatbot web interface.

Below is the high-level test case suite for the Chatbot application with expected results and status:

6.1 Unit Test Cases

This section describes the unit test cases for front-end client, middleware written on AWS Lambda, and backend API written to extract the data from the MongoDB database.

The main purpose of creating and executing the unit test cases is to validate the logic of individual functions without being dependent on the external systems.

Web Client

Test Case Description	Expected Result	Actual Result	Status
Verify that if a response contains an image, it is	Image is displayed in the chat area with	Correct image is shown in the chat	Pass

displayed in the chat area	proper height and width	area	
Verify that a voice recording is stored on the disk	Voice recording with length of user's question is stored in the folder	.wav file is created and stored in the /uploads folder	Pass
Verify that a sound file is deleted after a request has been completed	File saved on the disk for recording is deleted	.wav file is deleted after request is completed with success/failure	Pass

AWS Lambda Middleware

Test Case Description	Expected Result	Actual Result	Status
Verify that a proper response is formed if no answer found in the database	Lambda function should form a proper response if no response received from Backend server	Upon receiving no information from the Backend, Lambda function creates a message that "We couldn't find anything"	Pass
Lambda function re-tries the request to database on network failure	Lambda function makes request again within the DialogFlow timeout to attempt to get response	Lambda function gives the error to front-end client saying that it couldn't find they are looking for	Fail

Backend

Test Case Description	Expected Result	Actual Result	Status
Verify that intents and entities are converted to lower case	Lower cased intent and entity names	Upper cased intents and entities are	Pass

	are used to search the response from DB	converted into lowercase before making a request	
Verify that an image file is written to the disk if a response contains an image	A file is written to the disk and its name is returned in the response	An image file with proper extension is written and its name is sent to client in the response	Pass

6.2 System Integration Test Cases

Test Case Description	Expected Result	Actual Result	Status
Verify that the Alexa has been configured properly	Alexa should return a response to a “Hello World” query	Alexa responds “Hello. How may I help you?”	Pass
Verify that chatbot return the shortcut code of the Android device	Chatbot should return a message with the text code	Chatbot returned a text message “Shortcut code of the LCD is xxxx”	Pass
Verify that chatbot returns the example image of Android UI component	Chatbot should present an example image of the functionality	Chatbot shows the image on the screen	Pass
Provide the question that chatbot cannot understand for sure	Default fallback intent should be triggered saying that chatbot didn’t understand the question	Default fallback intent is triggered with the message saying that chatbot couldn’t understand	Pass
Provide the invalid input character(s) to the chatbot	Default fallback intent should be triggered saying that	Chatbot returns the message that it didn’t understand	Pass

	chatbot didn’t understand the question	the question and asks user to repeat	
Verify that chatbot maintains the context of the conversation and identifies the entities in ongoing conversation	Chatbot should be able to understand the entity and intent in the ongoing conversation without user actually specifying it	After asking a question about the class, chatbot maintains the context and answers the next question asked without intent trigger phrase	Pass
Verify that the chatbot provides the appropriate message when it is down	When any of the system is down, chatbot should return an appropriate message for system failure	DialogFlow provides the default response, if the client itself is offline, it shows the error that user needs to be online for the service	Pass

Chapter 7: Performance and Benchmarks

We are evaluating the performance of a chatbot is the key operation to make sure whether users will use it. There are no defined standards for evaluation of the chatbot performance. However, there are some widely used metrics in the industry to measure its performance.

We will be using following metrics during the development to evaluate the performance of the chatbot periodically:

- Response Time

Response time refers to the fastest possible and accurate response from the chatbot. Getting the response in fraction of seconds is the ideal scenario but there is no ideal response time for the expected scenario. We are creating a chatbot with minimal response time so as to succeed as a ideal chatbot expectations. The ideal chatbot is the one with the fastest response time and the most accurate result pertaining the expected result.

- Length of answers

Length of answer is one of the most important measure while developing a chatbot. The length of the answer is not measured as the number of words but is measured in the most appropriate one word or one sentence answer provided to the user based on his questions. The tags in the question are matched for the appropriate result or answer to be returned in response to the question. In our project we are planning to integrate it will the Google voice assistant hence it is mandatory to take into consideration the size of the content that we are returning as a answer

- Accuracy

Accuracy is one of the characteristics which is measured from the intents and entities that we are setting while creating the chatbot. We provide the answers to the chatbot through the intents that we set to the chatbot. The model then tries to find the appropriate tags from the user provided questions and then bot provided the accurate answers.

- Conversational context and flow

Chatbot should be able to maintain the context of the conversation. From the previous chat the chatbot should be able to answer the next questions asked by the user. We are considering a scenario in which the user is taking to the chatbot regarding the installation of the android suite and then followed by a series of questions and in this manner the bot should respond to the user appropriately

- Scalability

The chatbot that we developed is designed in such a way that it can be scaled at later point in time. We are trying to integrate the cloud services technologies in the chatbot. Along with the unit testing,we will perform the performance testing for various users. Initially we are planning to keep the load within the free tier limits of the cloud services that we are planning to use.

- Error Handling

As the chatbot has the various capabilities like auto correcting the mistakes. The chatbot handles all the error and provides the correct response within short amount of time. The meaningful response is returned by the user in short amount of time.

Chapter 8: Deployment, Operations, Maintenance

This chapter talks about our deployment plan, operations and maintenance of the code and application on the executable environment.

8.1 Deployment

Our deployment architecture is very simple and straightforward as we are leveraging most of the cloud infrastructure and tools to manage our application lifecycle. For example- we have built our client application using node js and deployed it on AWS ec2 instance. In the middle ware we are using Google cloud dialogflow api.ai for performing our natural language processing tasks. This also does not require any thing to be set up and everything can be configured here on the fly without need of future maintenance. Also, our business logic is deployed using serverless AWS lambda which is again free of any such configuration and its management. At the backend, our restful web services are created and deployed on AWS ec2 using node js server. The backend apis are developed and deployed on the node js server with little or no configuration management. We are using noSQL mongodb for our data storage and this is also running on an AWS VPC on the cloud. This AWS VPC infrastructure helps to set up a mongodb cluster on demand.

The below sequence of activities are performed by the system manager as part of the initial deployment plan:

- Creating and managing the deployment plan schedule showing the individual tasks associated with installation of required software/hardwares.
- Pre-installation activities based on the system requirements and finalizing the installation plan and other deployment documents.
- Verification that each software/hardware are set-up and installed properly.
- Verification the multiple systems are communicating properly before deployment.
- Verification that any new features added to the system is not breaking the existing features.
- Monitoring the health of the system after the system module is deployed in the production environment.

Deployment steps for deploying MongoDb on the AWS cloud -

- Sign up for an AWS account. Then choose a region, create a key pair.
- **Launch the QuickStart** - Launch the AWS CloudFormation template into your AWS account, specify parameter values, and create the stack.

- Connecting to MongoDb nodes - Using SSH to connect to MongoDB nodes via the NAT instance, as the nodes are in a private subnet.

Deployment steps for client application on the AWS cloud -

- As the client application is developed using NodeJS, transfer the source code to the particular location on the EC2 instance using SSH connection.
- Update the running port of the client web application from the default NodeJS port 3000.
- From the security group applied to the EC2 instance, open the port for the incoming connection so that the client web application can be accessed from outside world.
- Use "npm start" command to start the client application.

8.2 Operations

We carried out the operations according to the deployment plan that we created in the planning phase of product development. The operations were carried out in the same sequence keeping in mind the timelines and each phases of our complement development strategy. Below are the major operations that are being carried out:

- Software/Hardware setup and installation
- Requirements gathering and preparing SRS document
- Creating System Design Document - High Level and Functional Document
- User Interface design for web and mobile application
- Gathering test data based on the website data and set up mongodb with that
- Developed intent processing api to search in database and provide response
- Creating DialogFlow agents, intents, entities and configuring lambda webhook
- Developed lambda api for processing the intents and entities and call the backend api to fetch the response
- Creating test agent to test the lambda api independently.
- Test cases implementation for backend processing engine
- Deployment of mongoDB on EC2 instance
- Deployment of UI application on the node server running in AWS

8.3 Maintenance

The maintenance of the application is quite easy due to the on the go cloud apis and cloud server and its infrastructure that we are using for building our chatbots. The UI client api is running on the node server which is deployed on AWS which maintains the scalability and multi tenancy of the application even under high load of user requests. The deployment activity is minimal as after change we don’t need to rebuild the whole application and deploy again. The middleware of the application is also quite maintainable due to the serverless lambda api that we are using. Again as the lambda doesn’t require any server to be configured for its working and just a mere click takes all the burden off of managing the code when its changed and then its deployment. The backend restful api is pretty agile and robust as its built on node server running on AWS and its connected to the database built using mongodb which is also running in the same AWS infrastructure which provides easy connectivity and maintenance of these applications. Only thing we need to take care of the AWS server instances that our client api, lambda and the backend mongodb server instance are running on, that those AWS infrastructure is running properly while the applications are in use by the end users.

Chapter 9: Summary, Conclusions, and Recommendations

9.1 Summary

Since the starting of the chatbot project, as per the requirements gathering, implementation, modification, results have been key components. User can ask many types of android questions to the chat bot. Chatbot return the answer in various formats. Simple format can be the text like sentence or words. Answer can include image or long description with url having detailed description of the topic. In web based or mobile based platforms all format are useful. In voice assistant like alexa or google home, only short answers are meaningful. In such cases only short answers are returned. User of the chatbot are going to be android developers. They can ask question like diagnostic code for factory-restore, shortcut in mac or windows for building the project, api classes for AbsListView, ui element like glass action bar, studio about emulator. Various type of answers will be returned. This is how users are going to use the chatbot and its use-cases.

Now, how chatbot functions internally is described. To provide the proper answer, mongodb database is used to store the answer for the questions. The data is stored in the form of collections and documents. There are apis developed in node js to return the answer as per the question from the database. Amazon lambda is used for secure http connection with dialog flow. Lambda does the input fields extraction and calls the apis to fetch the answer. All backend apis and database are deployed on the aws instance. The dialog flow is used for intent and entity extraction. Also it does the training the model for questions and figuring out the important fields. That flow goes to lambda. The web application is used for input questions. User can use microphone option for input questions and the web application calls the dialog flow api. For some type of questions, tensorflow, tflearn libraries are used. The model is getting trained from the data available in intent, entity format. In this type, the question is given to the model to fetch the answer from the database. Other approach used is sequence to sequence rnn model. In this approach, there are many phases. First phase is data gathering, cleaning and preprocessing for the model training. The matrix with term frequencies are created for the model. Then using long short term memory the model is trained and after training it memorize some keywords to answer. In this case when question is asked it is directly given to the model to provide the answer and no need to check the database. This way whole chatbot is developed end-to-end and deployed on the cloud.

9.2 Conclusions

Mainly chatbot supports questions related to android. The android developers can find the answers using the bot. Internal functions of the bot are included in the summary. It is very useful tool for android developers to quickly get the answer. To conclude, chatbot works with a wide range of questions and provides answers in many forms. As far as development and implementation is concern, the main motive is to answer accurately in closed domain. So for user it can be impactful.

9.3 Recommendations for Further Research

Current implementation of the chatbot is closed domain and generative approach. In the same category, further research area is that answer the question with same accuracy as current architecture but using rnn, sequence to sequence model. That model doesn’t look up into the database for fetching specific terms. But the training has been done in such a way that it figures out the keywords. So the research area can be to get the accurate result using the rnn model. The chatbot is implemented for web based application. That supports voice assistance as well using microphone. Now there can be various interfaces like alexa, google home, mobile application. That integration is possible with the current implementation of the project.

Glossary

Amazon Alexa	Alexa is a virtual assistant developed by Amazon, first used in the Amazon Echo and the Amazon Echo Dot smart speakers developed by Amazon Lab126.
Amazon Alexa Services	The Alexa Voice Service (AVS) enables you to access cloud-based Alexa capabilities with the support of AVS APIs, hardware kits, software tools, and documentation.
AWS Lambda	AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of the Amazon Web Services.
Deep Learning	Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.
DialogFlow	DialogFlow (formerly Api.ai, Speaktoit) is a Google-owned developer of human–computer interaction technologies based on natural language conversations.
ElasticSearch	Database which internally uses Lucene based search for fast searching.
Entity	Entities are powerful tools used for extracting parameter values from natural language inputs. Any important data you want to get from a user's request, will have a corresponding entity.
Intent	An intent represents a mapping between what a user says and what action should be taken by your software.
LSTM	Long short-term memory (LSTM) units are units of a recurrent neural network (RNN). An RNN composed of LSTM units is often called an LSTM network. A

	common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.
Machine Learning	Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
Natural Language Processing (NLP)	Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken. NLP is a component of artificial intelligence (AI).
Neural Network	A computer system modeled on the human brain and nervous system.
React JS	In computing, React (sometimes React.js or ReactJS) is a JavaScript library[2] for building user interfaces.
REST API	REST or RESTful API design (Representational State Transfer) is designed to take advantage of existing protocols.
RNN	A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence.
Sequence to sequence	A Sequence-to-Sequence model reads a sequence (such as a sentence) as an input and produces another sequence as an output
Tensorflow	TensorFlow is an open-source software library for dataflow programming across a range of tasks.
Tflearn	TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

Webhook	A WebHook is an HTTP callback: an HTTP POST that occurs when something happens; a simple event-notification via HTTP POST.

References

1. DialogFlow. (n.d.). Key Concepts. Retrieved May 11, 2018, from https://dialogflow.com/docs/fulfillment

2. Liza Daly, Chatbot Fundamentals, An interactive guide to writing bots in Python https://apps.worldwritable.com/tutorials/chatbot/

3. Chen, S. (2017, November 4), Getting started with DialogFlow and building my first bot, Retrieved from https://chatbotslife.com/getting-started-with-dialogflow-and-building-my-first-bot-newbies-guide-d025d4eed3b2

4. AWS Lambda. (n.d.). AWS Lambda Developer Guide, Retrieved May 11, 2018,from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

5. Sayed S, Jain R., & Lokhandwala, B. (0975 – 8887) Android based Chat-Bot International Journal of Computer Applications Volume 137 – No.10, March 2016 28

6. Yan M, Castro P., Cheng P., & Ishakian V, Building a Chatbot with Serverless Computing. MOTA '16. doi: 10.1145/3007203.3007217

7. Huang J, Zhou M., & Yang D, Extracting Chatbot Knowledge from Online Discussion Forums.

8. Sameera A., Kader A., & Woods J. (2013). Survey on Chatbot Design Techniques in Speech Conversation Systems. Paper published in (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 6, No. 7, 2015 72 | P a g e www.ijacsa.thesai.org

9. Abashev A, Grigoryev R, Grigorian K, Boyko V (2017) Programming Tools for Messenger-Based Chatbot System Organization . Paper published in BioNanoScience, Vol 7, issue 2, pp 403-407

10. Chatbot And Service Industry (n.d) , Retrieved May 11, 2018, from https://www.marutitech.com/chatbots-and-service-industry/

11. Hanh Nguyen, M (2017, October 20), The latest market research, trends & landscape in the growing AI chatbot industry,Retrieved from, http://www.businessinsider.com/chatbot-market-stats-trends-size-ecosystem-research-201710