**Amrita Vishwa Vidyapeetham**

**Amrita School of Engineering, Amritapuri Campus**

Project Title : **CUSTOMER ASSISTANT CHATBOT**

Course Name : Android Application and Development

Course Code: 18CA386

Batch : S4MCA

Submitted By: Abhilash Saj George (AM.SC.P2CSC19002)

Harikrishnan P S (AM.SC,P2CSC19018)

**Contents**

1. **Abstract**

We all know that chat bots are self help tool that help us for better communications. Many companies are now using these chatbots for better communication with their customers and also to increase their sales.In this Project we are creating a customer Assistance chatbot with the help of IBM Watson. Watson Assistant is a question answering computer system capable of answering questions in natural languages.

1. **System requirements/ features**

The features include:

1. NLP:

It consist of services like Intent, Entity Recognition, Dialog Flow, Analytics.

**Intents** are purposes or goals that are expressed in a customer's input. By recognizing the intent expressed in a customer's input, theWatsonAssistant service can choose the correct dialog flow for responding to it.

**Entities** are like nouns or keywords. By building out your business terms in entities your assistant can provide targeted responses to queries.

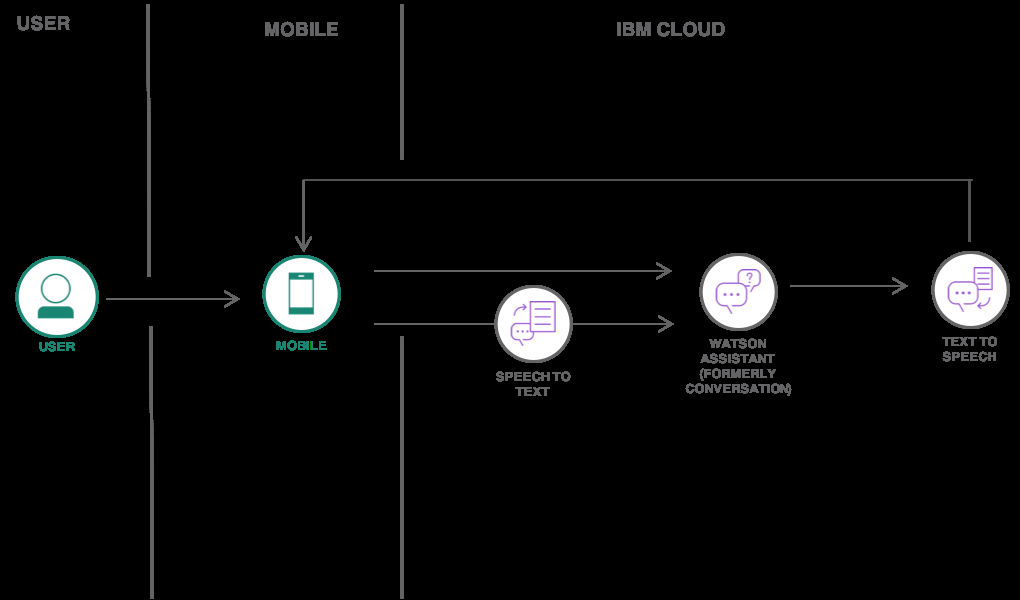
**Dialog** defines what our assistance says in response to the customer.

**Analytics** will give full details about the users like what the user mostly talk, How many users chat etc. It also give information about total conversations, average message per customer, Max. conversations, Weak understanding, top intents, etc.

1. Complaints

Here

1. **System architecture**



1. **User interface designs**



1. **Android components/services/API used for different functions**

For Creating our Project we mainly used 3 API. They are:

1. Text to Speech
2. Speech to Text
3. Watson Assistant
4. **Data storage**

Since we are using IBM Watson for creating our chat Bot, the messages will get stored in the Assistance itself.

1. **Design Blue print/code**

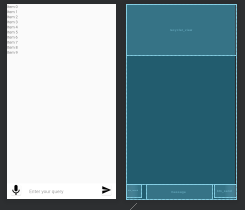
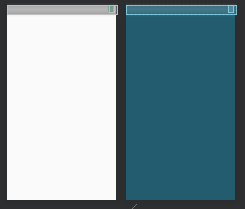
Activity\_main.xml

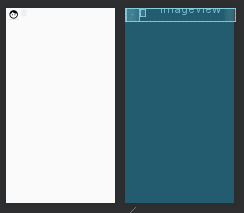
*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:context="com.example.vmac.WatBot.MainActivity"**>  
  
 *<!-- <android.support.design.widget.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"/>  
  
 </android.support.design.widget.AppBarLayout>-->* <**include layout="@layout/content\_chat\_room"** />  
  
</**android.support.design.widget.CoordinatorLayout**>

MainActivity.java

**package** com.example.vmac.WatBot;  
  
**import** android.Manifest;  
**import** android.content.Context;  
**import** android.content.pm.PackageManager;  
**import** android.graphics.Typeface;  
**import** android.net.ConnectivityManager;  
**import** android.net.NetworkInfo;  
**import** android.os.Bundle;  
**import** android.support.annotation.NonNull;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v4.content.ContextCompat;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.DefaultItemAnimator;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.ImageButton;  
**import** android.widget.Toast;  
  
**import** com.ibm.watson.developer\_cloud.android.library.audio.MicrophoneHelper;  
**import** com.ibm.watson.developer\_cloud.android.library.audio.MicrophoneInputStream;  
**import** com.ibm.watson.developer\_cloud.android.library.audio.StreamPlayer;  
**import** com.ibm.watson.developer\_cloud.android.library.audio.utils.ContentType;  
**import** com.ibm.watson.developer\_cloud.assistant.v1.Assistant;  
**import** com.ibm.watson.developer\_cloud.assistant.v1.model.InputData;  
**import** com.ibm.watson.developer\_cloud.assistant.v1.model.MessageOptions;  
**import** com.ibm.watson.developer\_cloud.assistant.v1.model.MessageResponse;  
**import** com.ibm.watson.developer\_cloud.speech\_to\_text.v1.SpeechToText;  
**import** com.ibm.watson.developer\_cloud.speech\_to\_text.v1.model.RecognizeOptions;  
**import** com.ibm.watson.developer\_cloud.speech\_to\_text.v1.model.SpeechRecognitionResults;  
**import** com.ibm.watson.developer\_cloud.speech\_to\_text.v1.websocket.BaseRecognizeCallback;  
**import** com.ibm.watson.developer\_cloud.text\_to\_speech.v1.TextToSpeech;  
**import** com.ibm.watson.developer\_cloud.text\_to\_speech.v1.model.SynthesizeOptions;  
  
**import** java.io.InputStream;  
**import** java.util.ArrayList;  
  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
  
 **private** RecyclerView recyclerView;  
 **private** ChatAdapter mAdapter;  
 **private** ArrayList messageArrayList;  
 **private** EditText inputMessage;  
 **private** ImageButton btnSend;  
 **private** ImageButton btnRecord;  
 *//private Map<String,Object> context = new HashMap<>();* com.ibm.watson.developer\_cloud.assistant.v1.model.Context context = **null**;  
 StreamPlayer streamPlayer;  
 **private boolean** initialRequest;  
 **private boolean** permissionToRecordAccepted = **false**;  
 **private static final int** REQUEST\_RECORD\_AUDIO\_PERMISSION = 200;  
 **private static** String TAG = **"MainActivity"**;  
 **private static final int** RECORD\_REQUEST\_CODE = 101;  
 **private boolean** listening = **false**;  
 **private** SpeechToText speechService;  
 **private** MicrophoneInputStream capture;  
 **private** SpeakerLabelsDiarization.RecoTokens recoTokens;  
 **private** MicrophoneHelper microphoneHelper;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 inputMessage = findViewById(R.id.message);  
 btnSend = findViewById(R.id.btn\_send);  
 btnRecord= findViewById(R.id.btn\_record);  
 String customFont = **"Montserrat-Regular.ttf"**;  
 Typeface typeface = Typeface.createFromAsset(getAssets(), customFont);  
 inputMessage.setTypeface(typeface);  
 recyclerView = findViewById(R.id.recycler\_view);  
  
 messageArrayList = **new** ArrayList<>();  
 mAdapter = **new** ChatAdapter(messageArrayList);  
 microphoneHelper = **new** MicrophoneHelper(**this**);  
  
  
 LinearLayoutManager layoutManager = **new** LinearLayoutManager(**this**);  
 layoutManager.setStackFromEnd(**true**);  
 recyclerView.setLayoutManager(layoutManager);  
 recyclerView.setItemAnimator(**new** DefaultItemAnimator());  
 recyclerView.setAdapter(mAdapter);  
 **this**.inputMessage.setText(**""**);  
 **this**.initialRequest = **true**;  
 sendMessage();  
  
 *//Watson Text-to-Speech Service on IBM Cloud* **final** TextToSpeech textService = **new** TextToSpeech();  
 *//Use "apikey" as username and apikey values as password* textService.setUsernameAndPassword(**"apikey"**, **"AsnHNiFUV8hf65M6jGsUXe1bOgv9IK3DFb2AyNrW33wi"**);  
 textService.setEndPoint(**"https://api.au-syd.text-to-speech.watson.cloud.ibm.com/instances/16213b11-3f07-4519-97e1-e93a29e563e3"**);  
  
 **int** permission = ContextCompat.checkSelfPermission(**this**,  
 Manifest.permission.RECORD\_AUDIO);  
  
 **if** (permission != PackageManager.PERMISSION\_GRANTED) {  
 Log.i(TAG, **"Permission to record denied"**);  
 makeRequest();  
 }  
  
  
 recyclerView.addOnItemTouchListener(**new** RecyclerTouchListener(getApplicationContext(), recyclerView, **new** ClickListener() {  
 @Override  
 **public void** onClick(View view, **final int** position) {  
 Thread thread = **new** Thread(**new** Runnable() {  
 **public void** run() {  
 Message audioMessage;  
 **try** {  
  
 audioMessage =(Message) messageArrayList.get(position);  
 streamPlayer = **new** StreamPlayer();  
 **if**(audioMessage != **null** && !audioMessage.getMessage().isEmpty()) {  
 SynthesizeOptions synthesizeOptions = **new** SynthesizeOptions.Builder()  
 .text(audioMessage.getMessage())  
 .voice(SynthesizeOptions.Voice.EN\_US\_LISAVOICE)  
 .accept(SynthesizeOptions.Accept.AUDIO\_WAV)  
 .build();  
 streamPlayer.playStream(textService.synthesize(synthesizeOptions).execute());  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 thread.start();  
 }  
  
 @Override  
 **public void** onLongClick(View view, **int** position) {  
 recordMessage();  
  
 }  
 }));  
  
 btnSend.setOnClickListener(**new** View.OnClickListener(){  
 @Override  
 **public void** onClick(View v) {  
 **if**(checkInternetConnection()) {  
 sendMessage();  
 }  
 }  
 });  
  
 btnRecord.setOnClickListener(**new** View.OnClickListener() {  
 @Override **public void** onClick(View v) {  
 recordMessage();  
 }  
 });  
 };  
  
 *// Speech to Text Record Audio permission* @Override  
 **public void** onRequestPermissionsResult(**int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults) {  
 **super**.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 **switch** (requestCode){  
 **case** REQUEST\_RECORD\_AUDIO\_PERMISSION:  
 permissionToRecordAccepted = grantResults[0] == PackageManager.PERMISSION\_GRANTED;  
 **break**;  
 **case** RECORD\_REQUEST\_CODE: {  
  
 **if** (grantResults.length == 0  
 || grantResults[0] !=  
 PackageManager.PERMISSION\_GRANTED) {  
  
 Log.i(TAG, **"Permission has been denied by user"**);  
 } **else** {  
 Log.i(TAG, **"Permission has been granted by user"**);  
 }  
 **return**;  
 }  
 **case** MicrophoneHelper.REQUEST\_PERMISSION: {  
 **if** (grantResults.length > 0 && grantResults[0] != PackageManager.PERMISSION\_GRANTED) {  
 Toast.makeText(**this**, **"Permission to record audio denied"**, Toast.LENGTH\_SHORT).show();  
 }  
 }  
 }  
 *// if (!permissionToRecordAccepted ) finish();* }  
  
 **protected void** makeRequest() {  
 ActivityCompat.requestPermissions(**this**,  
 **new** String[]{Manifest.permission.RECORD\_AUDIO},  
 MicrophoneHelper.REQUEST\_PERMISSION);  
 }  
  
  
 *// Sending a message to Watson Conversation Service* **private void** sendMessage() {  
  
 **final** String inputmessage = **this**.inputMessage.getText().toString().trim();  
 **if**(!**this**.initialRequest) {  
 Message inputMessage = **new** Message();  
 inputMessage.setMessage(inputmessage);  
 inputMessage.setId(**"1"**);  
 messageArrayList.add(inputMessage);  
 }  
 **else** {  
 Message inputMessage = **new** Message();  
 inputMessage.setMessage(inputmessage);  
 inputMessage.setId(**"100"**);  
 **this**.initialRequest = **false**;  
 Toast.makeText(getApplicationContext(),**"Tap on the message for Voice"**,Toast.LENGTH\_LONG).show();  
  
 }  
  
 **this**.inputMessage.setText(**""**);  
 mAdapter.notifyDataSetChanged();  
  
 Thread thread = **new** Thread(**new** Runnable(){  
 **public void** run() {  
 **try** {  
  
 Assistant assistantservice = **new** Assistant(**"2018-02-16"**);  
 *//If you like to use USERNAME AND PASSWORD  
 //Your Username: "apikey", password: "<APIKEY\_VALUE>"* assistantservice.setUsernameAndPassword(**"apikey"**, **"b4Vo1f\_gHgEHJ33VH1NG0Zn2p4gXLmR6fdHQv1WvLJEH"**);  
  
 *//TODO: Uncomment this line if you want to use API KEY* assistantservice.setApiKey(**"b4Vo1f\_gHgEHJ33VH1NG0Zn2p4gXLmR6fdHQv1WvLJEH"**);  
  
 *//Set endpoint which is the URL. Default value: https://gateway.watsonplatform.net/assistant/api* assistantservice.setEndPoint(**"https://api.au-syd.assistant.watson.cloud.ibm.com/instances/5487d2eb-6e50-48ed-b031-87cdd5a4e796"**);  
 InputData input = **new** InputData.Builder(inputmessage).build();  
 *//WORKSPACES are now SKILLS* MessageOptions options = **new** MessageOptions.Builder().workspaceId(**"0a213a28-53e2-4a4c-9cd9-0eac6dafc787"**).input(input).context(context).build();  
 MessageResponse response = assistantservice.message(options).execute();  
 Log.i(TAG, **"run: "**+response);  
  
 String outputText = **""**;  
 **int** length=response.getOutput().getText().size();  
 Log.i(TAG, **"run: "**+length);  
 **if**(length>1) {  
 **for** (**int** i = 0; i < length; i++) {  
 outputText += **'\n'** + response.getOutput().getText().get(i).trim();  
 }  
 }  
 **else** outputText = response.getOutput().getText().get(0);  
  
 Log.i(TAG, **"run: "**+outputText);  
 *//Passing Context of last conversation* **if**(response.getContext() !=**null**)  
 {  
 *//context.clear();* context = response.getContext();  
  
 }  
 Message outMessage=**new** Message();  
 **if**(response!=**null**)  
 {  
 **if**(response.getOutput()!=**null** && response.getOutput().containsKey(**"text"**))  
 {  
 ArrayList responseList = (ArrayList) response.getOutput().get(**"text"**);  
 **if**(**null** !=responseList && responseList.size()>0){  
 outMessage.setMessage(outputText);  
 outMessage.setId(**"2"**);  
 }  
 messageArrayList.add(outMessage);  
 }  
  
 runOnUiThread(**new** Runnable() {  
 **public void** run() {  
 mAdapter.notifyDataSetChanged();  
 **if** (mAdapter.getItemCount() > 1) {  
 recyclerView.getLayoutManager().smoothScrollToPosition(recyclerView, **null**, mAdapter.getItemCount()-1);  
  
 }  
  
 }  
 });  
  
  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 });  
  
 thread.start();  
  
 }  
 *//Record a message via Watson Speech to Text* **private void** recordMessage() {  
 speechService = **new** SpeechToText();  
 *//Use "apikey" as username and apikey as your password* speechService.setUsernameAndPassword(**"apikey"**, **"dxnYGVQPWjeMr2oVDsSOOpgiev-zcwcf78sXz1rd1D4B"**);  
 *//Default: https://stream.watsonplatform.net/text-to-speech/api* speechService.setEndPoint(**"https://api.au-syd.speech-to-text.watson.cloud.ibm.com/instances/5244418d-4e3d-4496-b058-232882a671f6"**);  
  
 **if**(listening != **true**) {  
 capture = microphoneHelper.getInputStream(**true**);  
 **new** Thread(**new** Runnable() {  
 @Override **public void** run() {  
 **try** {  
 speechService.recognizeUsingWebSocket(getRecognizeOptions(capture), **new** MicrophoneRecognizeDelegate());  
 } **catch** (Exception e) {  
 showError(e);  
 }  
 }  
 }).start();  
 listening = **true**;  
 Toast.makeText(MainActivity.**this**,**"Listening....Click to Stop"**, Toast.LENGTH\_LONG).show();  
  
 } **else** {  
 **try** {  
 microphoneHelper.closeInputStream();  
 listening = **false**;  
 Toast.makeText(MainActivity.**this**,**"Stopped Listening....Click to Start"**, Toast.LENGTH\_LONG).show();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
  
 }  
 }  
  
 */\*\*  
 \* Check Internet Connection  
 \** ***@return*** *\*/* **private boolean** checkInternetConnection() {  
 *// get Connectivity Manager object to check connection* ConnectivityManager cm =  
 (ConnectivityManager)getSystemService(Context.CONNECTIVITY\_SERVICE);  
  
 NetworkInfo activeNetwork = cm.getActiveNetworkInfo();  
 **boolean** isConnected = activeNetwork != **null** &&  
 activeNetwork.isConnectedOrConnecting();  
  
 *// Check for network connections* **if** (isConnected){  
 **return true**;  
 }  
 **else** {  
 Toast.makeText(**this**, **" No Internet Connection available "**, Toast.LENGTH\_LONG).show();  
 **return false**;  
 }  
  
 }  
  
 *//Private Methods - Speech to Text* **private** RecognizeOptions getRecognizeOptions(InputStream audio) {  
 **return new** RecognizeOptions.Builder()  
 .audio(audio)  
 .contentType(ContentType.OPUS.toString())  
 .model(**"en-US\_BroadbandModel"**)  
 .interimResults(**true**)  
 .inactivityTimeout(2000)  
 *//TODO: Uncomment this to enable Speaker Diarization  
 //.speakerLabels(true)* .build();  
 }  
  
 **private class** MicrophoneRecognizeDelegate **extends** BaseRecognizeCallback {  
  
 @Override  
 **public void** onTranscription(SpeechRecognitionResults speechResults) {  
 System.out.println(speechResults);  
 *//TODO: Uncomment this to enable Speaker Diarization  
 /\*SpeakerLabelsDiarization.RecoTokens recoTokens = new SpeakerLabelsDiarization.RecoTokens();  
 if(speechResults.getSpeakerLabels() !=null)  
 {  
 recoTokens.add(speechResults);  
 Log.i("SPEECHRESULTS",speechResults.getSpeakerLabels().get(0).toString());  
  
  
 }\*/* **if**(speechResults.getResults() != **null** && !speechResults.getResults().isEmpty()) {  
 String text = speechResults.getResults().get(0).getAlternatives().get(0).getTranscript();  
 showMicText(text);  
 }  
 }  
  
 @Override **public void** onConnected() {  
  
 }  
  
 @Override **public void** onError(Exception e) {  
 showError(e);  
 enableMicButton();  
 }  
  
 @Override **public void** onDisconnected() {  
 enableMicButton();  
 }  
  
 @Override  
 **public void** onInactivityTimeout(RuntimeException runtimeException) {  
  
 }  
  
 @Override  
 **public void** onListening() {  
  
 }  
  
 @Override  
 **public void** onTranscriptionComplete() {  
  
 }  
 }  
  
 **private void** showMicText(**final** String text) {  
 runOnUiThread(**new** Runnable() {  
 @Override **public void** run() {  
 inputMessage.setText(text);  
 }  
 });  
 }  
  
 **private void** enableMicButton() {  
 runOnUiThread(**new** Runnable() {  
 @Override **public void** run() {  
 btnRecord.setEnabled(**true**);  
 }  
 });  
 }  
  
 **private void** showError(**final** Exception e) {  
 runOnUiThread(**new** Runnable() {  
 @Override **public void** run() {  
 Toast.makeText(MainActivity.**this**, e.getMessage(), Toast.LENGTH\_SHORT).show();  
 e.printStackTrace();  
 }  
 });  
 }  
  
}

BluePrint



1. **Results**