CHAPTER 1
INTRODUCTION

1.1 AIM

The scope of the project is to make easy interaction between the system and the user. The basic idea is to make the system recognize human speech and in turn respond in the form of speech. The aim is to build an android application based ticket booking through which people can interact with the system directly through speech.

1.2 PROBLEM STATEMENT

Spoken Dialog Systems (SDS) are speech based interfaces for Human Computer Interaction (HCI). There are systems which responds in the form of text when the user talks and there are also some systems which responds in the form of voice when user gives the input in the form of text. There are systems which are called Interactive Voice Response (IVR) which operates following set of commands. Our project aims at developing a system that eliminates the drawback of existing systems. So, hereby we are going to develop a model which makes the user and system to get interacted without following a set of commands and thereby getting interacted in a natural language. The existing systems have a disadvantage where the user has to remember all the instructions, commands and keywords that are inbuilt in it to carry a specific task. These systems are not user friendly for illiterates and physically challenged. Therefore, to overcome all these hurdles we are going to propose a model to book a ticket.

1.3 DESCRIPTION

SPOKEN DIALOG SYSTEMS

A Spoken Dialog System is a software tool allowing communication via voice in order to perform a certain task. They are interactive computer systems which allow the user and the system to communicate with each other in natural language. It acts as an interface between an application and the user. According to the goal of the task one can identify:

- Transaction-based systems which allow the user to perform a transaction. (e.g. reserving a seat in a train, buying or selling stocks)
- Information-provision systems which provide information as answer to a query. (e.g. weather information, timetables for a train)

The output can be in form of synthesised voice or some data. Most part of these systems are designed for telephones or wireless devices which:

- Have quite small screen if at all.
- Are used for communication in real time therefore typing the input is not a solution.
- Sometimes have to be used in hands-free mode (e.g. in the car).

Such systems can be used for people with certain disabilities where typing or screen reading is not possible. Speech is the most natural way of communication atleast at telephone.

A common spoken dialogue system consists of six components:

Automatic Speech Recognition component (ASR): The system takes audio input and returns a string of words. Common systems have high error-rate.

Natural Language Understanding component (NLU): Produces a semantic representation from the words using syntactic and semantic analysis.

Natural Language Generation component (NLG): NLG generates in each dialog state an appropriate expression.

Text-To-Speech synthesis component (TTS): The TTS component converts the generated text to audio which is received as output.

Dialog manager: It controls the structure of the dialog. It takes the output of the NLU module and passes it to the task manager. Then it takes the output of the task manager and passes it to the NLG module.

HUMAN DIALOG

It is important for developers of dialog systems to have a sound understanding of nature of human dialog. Even when conversational modelling is not the main aim, understanding the complexity of human dialog is important for knowing how to constrain systems in the interests of performance and avoidance of errors. Dialog is a process of exchanging views, sometimes with the purpose of finding a solution to a problem or to solve differences. It is a reciprocal conversation between two or more entities.

Characteristics of a dialog:

Turn and Turn-Taking: A dialog consists of many turns, where in every turn one participant speaks.

Grounding: Common ground is the set of mutual beliefs by the speakers. The speaker makes an utterance and the hearer indicates whether he has understood it or not.

Conversational implicature: The ability of inferring the (explicit and implicit) meaning from the sentences based on assumptions.

DIALOG SYSTEMS AND MARKET

Voice portals are becoming extremely used in e-Commerce. Call centres are replacing at least part of their human services through such systems. According to a statistic from 2004 there were only 250 million computers with Internet access but 1.3 billion telephones in the world. In recent trend, more and more services are integrated on the mobile phone chip.

1.4 BENEFITS

- Natural language based interaction
- Even illiterates can use computers
- Human Computer Interaction
- Can be used by physically challenged
- Reliability

1.5 OTHER APPLICATIONS

- Weather forecasting
- Agriculture
- Traffic
- Remote Banking

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEM

2.1.1 INTERACTIVE VOICE RESPONSE (IVR)

Interactive Voice Response System or IVRS is phone technology that allows a computer to automatically detect voice and touch tones using a normal voice phone. The objective of IVRS is to cut down customer service costs by providing self service to customers and guiding them to the department, person or information that they need.

Modern IVRS / IVR systems enable users to interact through a computer system via two interfaces – the **Interactive Voice Response System (IVRS)** and now an **Interactive Response Web-Based System (IWR)**.

DRAWBACKS

- Menus are too long. Experts recommend that no menu should exceed four choices.
- There's too much information. When writing a script for IVR systems, start with the least amount of extraneous information possible.
- Voice prompts are hard to understand. This could be caused by two different factors. To save money, the organization didn't hire professional voice talent and may have recorded the audio over the phone instead of in a studio.

2.1.2 TEXT TO SPEECH

Text-to-speech (TTS) is a type of speech synthesis application that is used to create a spoken sound version of the text in a computer document, such as a help file or a Web page. TTS can enable the reading of computer display information for the visually challenged person, or may simply be used to augment the reading of a text message. Current TTS applications include voice-enabled e-mail and spoken prompts in voice response systems. TTS is often used with voice recognition programs. There are numerous TTS products available, including Read Please 2000, Proverb Speech Unit, and Next Up Technology's Text Aloud. Lucent, Elan, and AT&T each have products called "Text-to-Speech."

In addition to TTS software, a number of vendors offer products involving hardware, including the Quick Link Pen from WizCom Technologies, a pen-shaped device that can scan and read words; the Road Runner from Ostrich Software, a handheld device that reads ASCII text; and DecTalk TTS from Digital Equipment, an external hardware device that substitutes for a sound card and which includes an internal software device that works in conjunction with the PC's own sound card.

2.1.3 SPEECH TO TEXT

Speech-to-text software is a type of software that effectively takes audio content and transcribes it into written words in a word processor or other display destination. This type of speech recognition software is extremely valuable to anyone who needs to generate a lot of written content without a lot of manual typing. It is also useful for people with disabilities that make it difficult for them to use a keyboard. Speech-to-text software may also be known as voice recognition software.

In terms of technical function, many speech-to-text software programs break

spoken-word audio down into short "samples" and associate those samples with simple phonemes or units of pronunciation. Then, complex algorithms sort the results to try to predict the word or phrase that was said. Speech-to-text software has improved quite a bit in accuracy and evolved in general functionality to play a larger role in modern communications over digital platforms.

2.2 PROPOSED SYSTEM

2.2.1 SPOKEN DIALOG SYSTEM

Spoken Dialog System is a software tool which allows communication through voice in order to perform a particular task. They are interactive computer systems which use natural language for interaction. It acts as an interface between an application and the user. The output can be in the form of synthesized voice or some data. Such systems can be used for people with certain disabilities where typing or screen reading is not possible.

The proposed idea is a spoken dialog system which is a dialogue between a train ticket booking computational agent and a user. At any instance starting with “Sys” the agent should decide which is the best action it can perform. System actions include greeting, ask-source, ask-destination, ask-day-date, offer-ticket, book-ticket, say-goodbye. The learning problem here is to find an optimal state to action mapping i.e. given the state of the dialogue the system should know (among all actions) which is the best action to perform. For instance the system is starting the dialogue by saying “hi” i.e. greeting-action. As a computational agent it should learn to behave like this. As part of this project we will apply reinforcement learning algorithms to learn a state to action mapping and thus solve the dialogue management problem.

2.2.2 EXAMPLE DIALOG

The following illustrates the dialog between the user and the system for booking the ticket.

System : Hi, my name is Liza and I can book train tickets for you.

User: Hello there, I would like to book a ticket now.

System: Which is your origin city?

User: I am going from Chennai.

System: Which is your destination city?

User: I am going to Trichy.

System: On what day and time would you like to travel?

User: I would like to travel next Friday night.

System: 2nd tier AC sleeper tickets are available in Brindavan Express.

User: That would be great. Please send the mobile ticket to me.

System: I have booked the ticket and sent the PNR along with train timings to your mobile.

Fig 2.1 Interaction through speech

CHAPTER 3

SYSTEM ANALYSIS

3.1 FEASIBILITY STUDY

The main objective of feasibility study is to test the Technical, operational and economical for adding new modules and debugging old running system. All system is feasible for adding new modules and debugging old running system.

There are aspects in the feasibility study portion of the preliminary investigation:

- Technical feasibility
- Economic feasibility
- Operational feasibility

3.1.1 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. To develop this system, we first worked with web applications and then found that Android would be technically feasible. The requirements here are very modest because the system here supports very basic android version which is 2.4 and above.

So, this system is technically feasible because it needs only android with a basic version.

3.1.2 ECONOMIC FEASIBILITY

Economic feasibility is the most frequently used method for evaluating the effectiveness of the proposed system. More commonly known as cost-benefit analysis, the procedure that costs for a proposed system and weights them against the tangible and intangible benefits of the system. The

system is cost effective because it is freely available as android application and can be downloaded at free of cost from the internet. The system is economically feasible because the users can freely download and install the app on their phones and can run it.

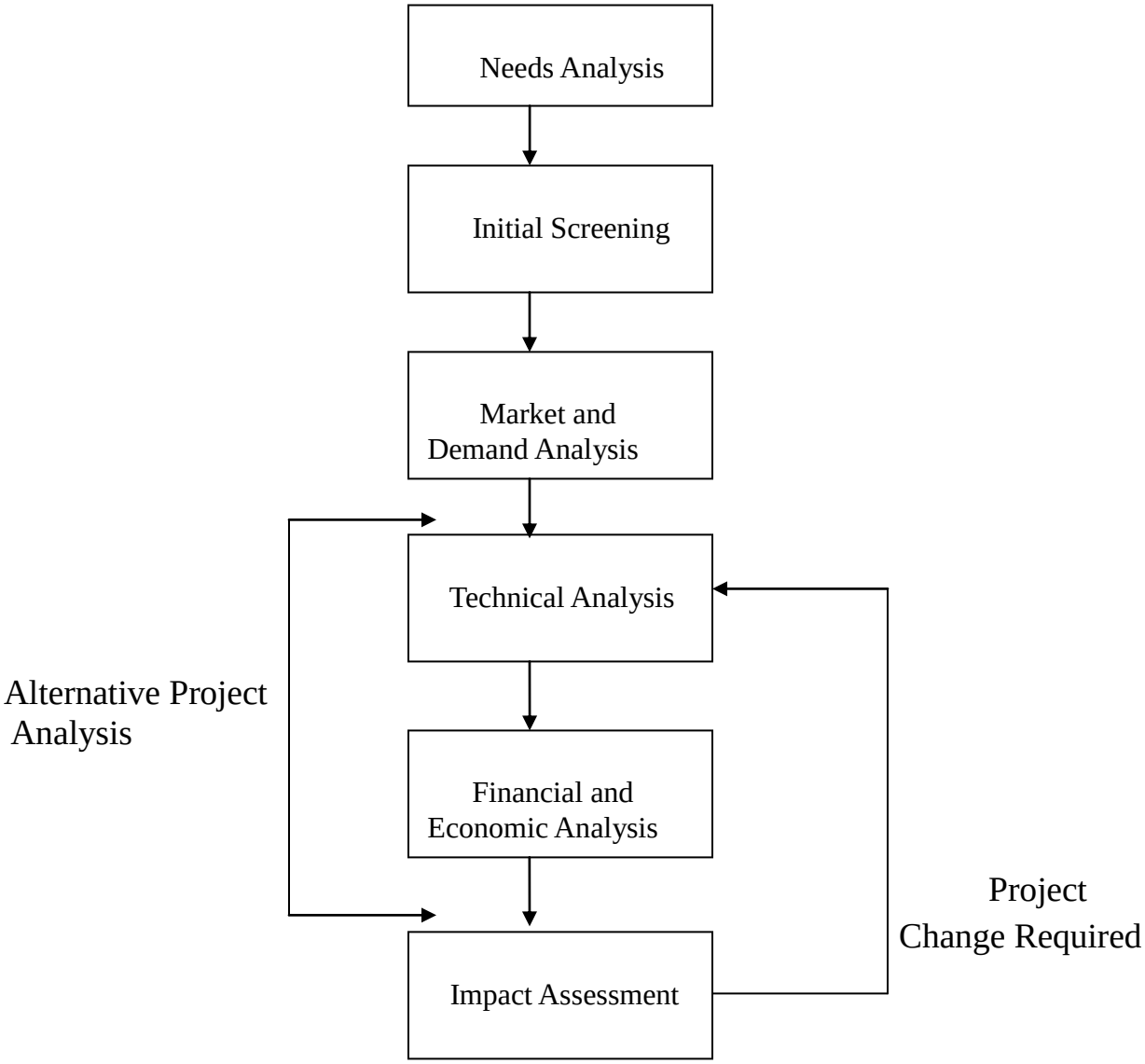

Fig. 3.1: Feasibility Analysis

3.1.3 OPERATIONAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. This system will not threaten the user instead it is friendly in its operation. All the user needs to have is an Internet connection provided to his mobile. The system is operationally feasible because the user can run the application just by clicking on the mic icon present there and then continue talking. The whole process is done only through the speech recognition.

3.2 HARDWARE USED

RAM	: 1GB and above
Processor	: Intel Processor (Core i3)

3.3 SOFTWARE USED

Operating System	: Android OS (Jelly Beans)
Version	: 2.4 and above
Frame Work	: JDK 1.6
Front End	: Java
Back End	: My SQL

CHAPTER 4

DETAILED DESIGN

4.1 SYSTEM ARCHITECTURE

The following is the architecture of spoken dialog system.

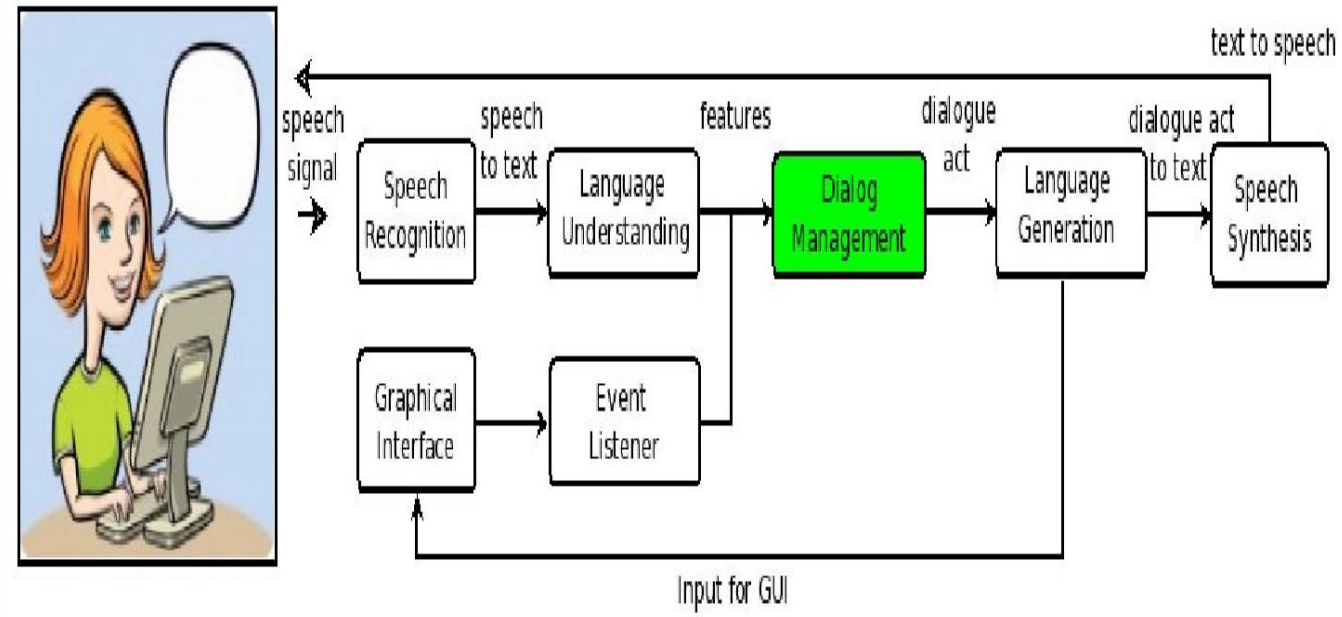

Fig 4.1: System Architecture

4.1.1 MODULES

The architecture of spoken dialog system has six modules and they are as follows:

a. **Speech Recognition**:

The input to this module is the speech signal generated by the user. It recognizes the signal and converts it into text format.

b. **Language Understanding**:

It receives the text generated by the speech recognition module. It analyses the text and sends the essential features to the dialogue manager.

Eg: When we say "I need a ticket from **Chennai** to **Delhi**" it takes only the necessary details(Source and Destination).

c. **Dialog Management:**

It decides what to do or ask after it receives the input from the user. The decision of the dialogue management will have only the necessary details.

Eg: It just asks "ask-date-time".

d. **Language Generation:**

The necessary details from the dialogue management module is framed to user conveyable text in language generator.

e. **Speech Synthesis:**

The text received from the language generation module is converted to speech signal. This speech signal is forwarded to the user as a response to the request.

f. **Graphical Interface:**

This helps the user to interact with the system using graphical icons rather than text.

FLOW DIAGRAM

The following diagram describes about the flow that takes place in the process of ticket booking.

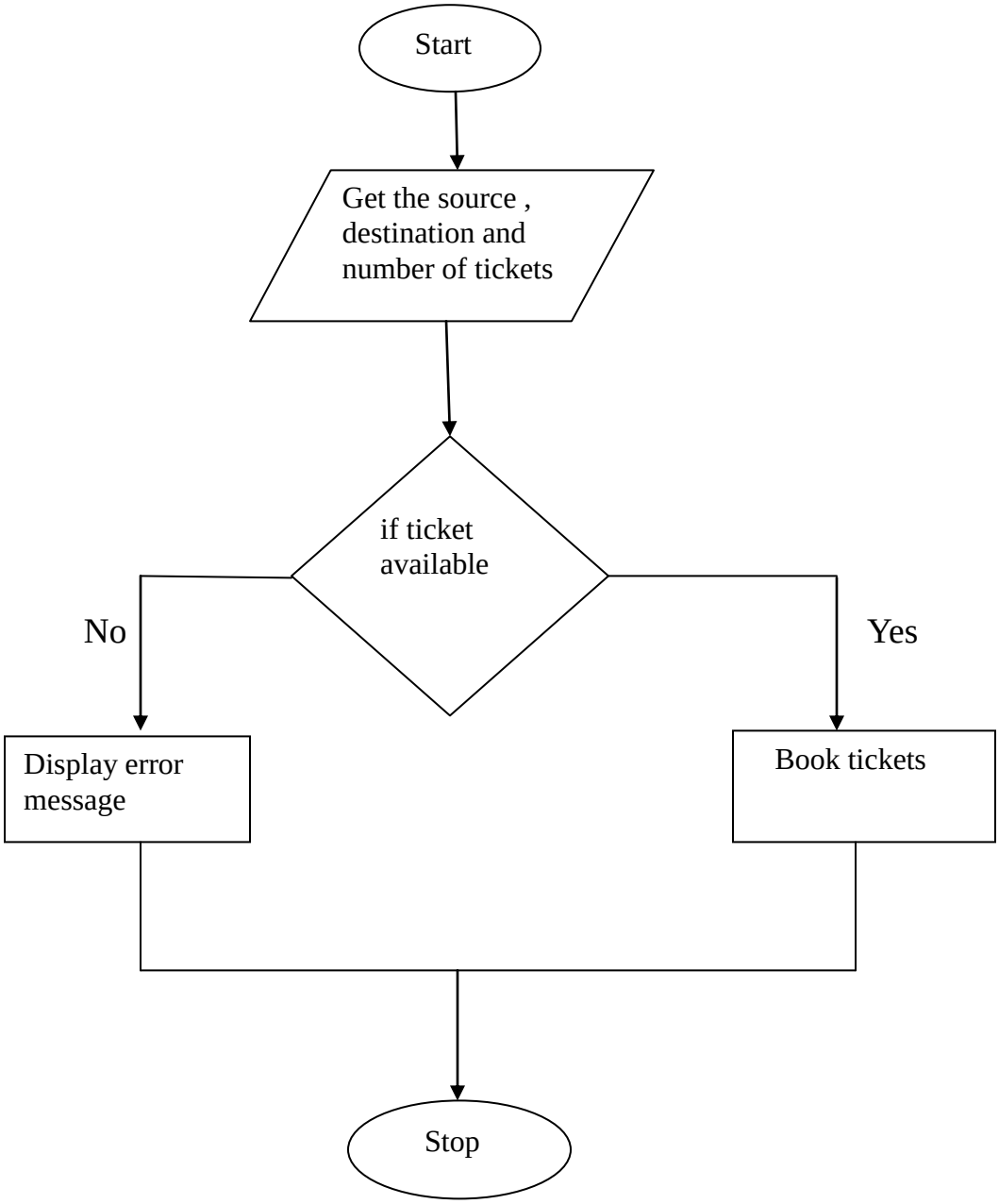

Fig 4.2: Flow Diagram for ticket booking

4.3 UML DIAGRAMS

The Unified Modelling Language (UML) is a general purpose modelling language in the field of software engineering. The basic level provides a set of graphic notation techniques to create visual methods of object-oriented software-intensive systems. Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects.

4.3.1 USE CASE DIAGRAM

Use case describes the interaction between one or more actors and the system itself, represented as a sequence of simple steps that take part in a sequence of activities in a dialog with the system to achieve goal.

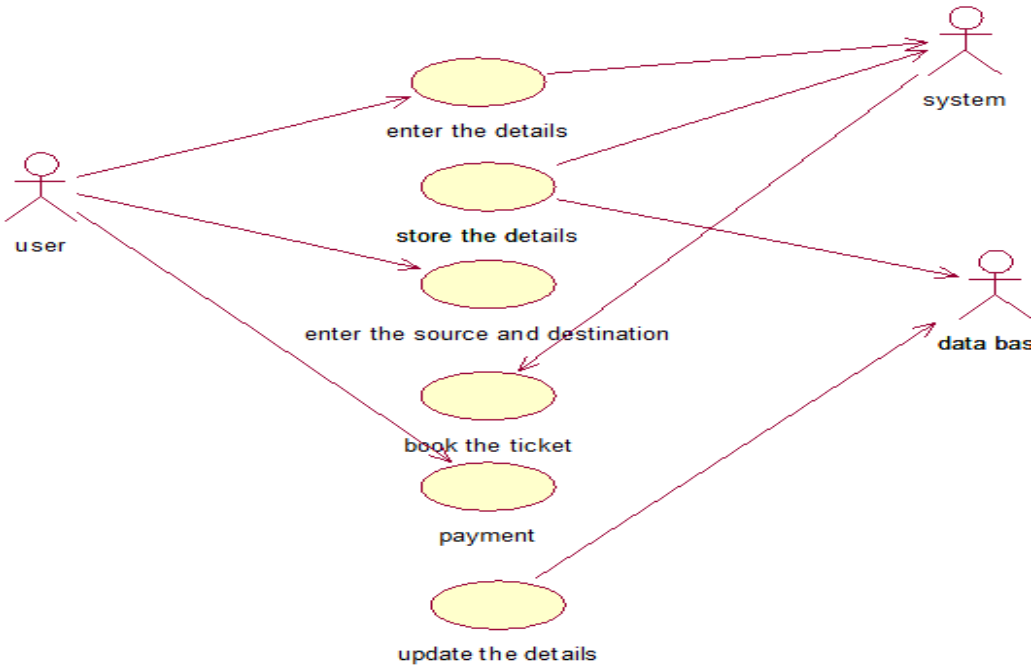

Fig 4.3: Use Case Diagram

There are three actors who are user, system and database necessary to run the application. The various functions of these actors like storing, entering

updating and retrieving the details are represented through use cases.

4.3.2 SEQUENCE DIAGRAM

A Sequence diagram shows, as parallel vertical lines different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

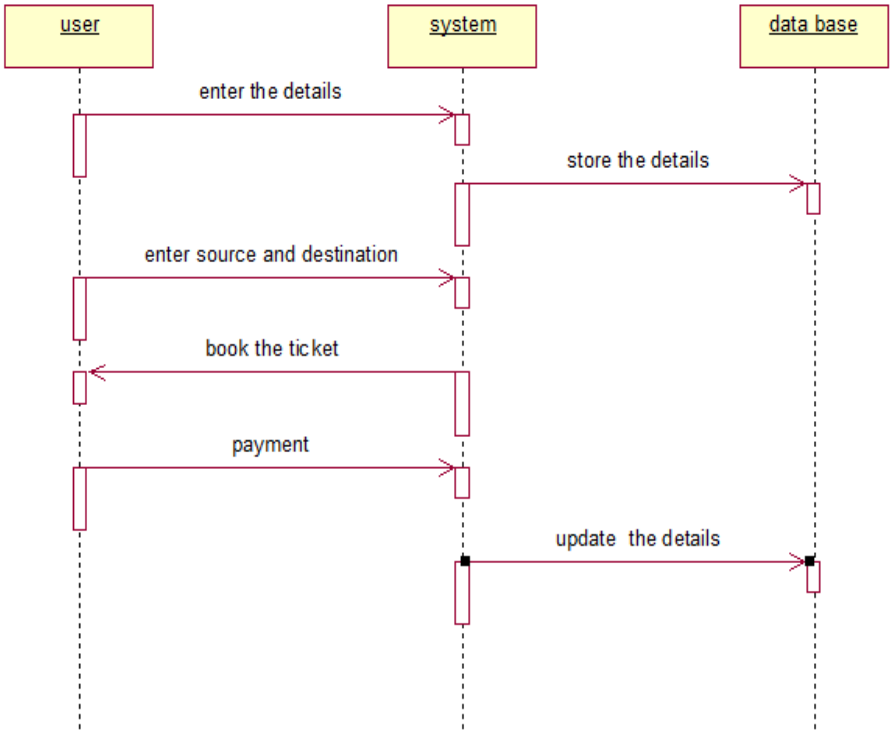

Fig 4.4: Sequence Diagram

The actions such as entering the details, storing and so on are expressed in a sequential order through vertical lines. The exchange of data between the actors(user, system and database) are represented by a set of horizontal lines.

4.3.3 ACTIVITY DIAGRAM

Activity diagram are graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

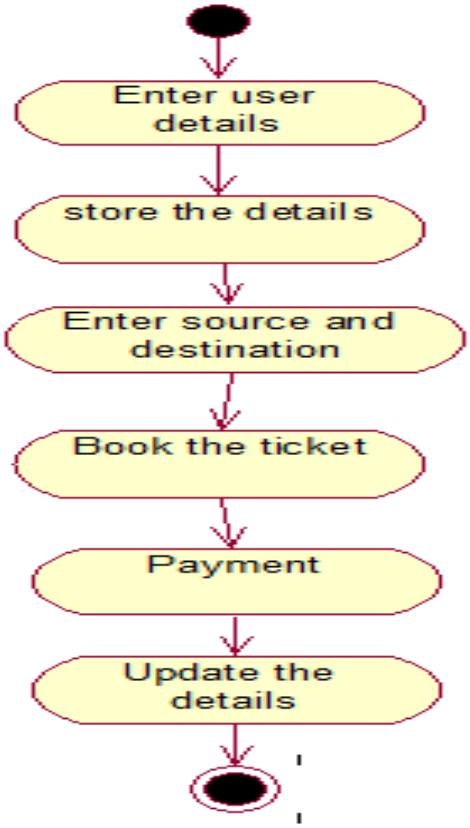

Fig 4.5: Activity Diagram

The set of activities that takes place among the actors is represented through activity diagram. It has a start and stop state which indicates the user to start and stop the activity respectively. The sequence of activities are represented through arrow marks.

4.3.4 COLLABORATION DIAGRAM

A Collaboration diagram is easily represented by modelling objects in a system and representing the association between the objects as links. The interaction between the objects is denoted by arrows. To identify the sequence of invocation of these objects, a number is placed next to each of arrows.

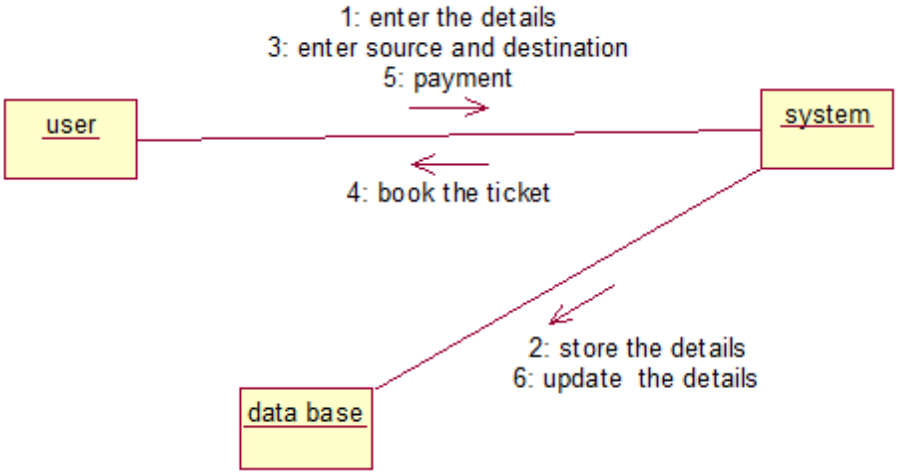

Fig 4.6: Collaboration Diagram

According to the sequence diagram drawn, an automatic collaboration diagram would be generated which again indicates the relationship among the actors.

4.3.5 CLASS DIAGRAM

Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from modelling the domain-specific data structure to detailed design of the target system. With the share model facilities, you can reuse your class model in the interaction diagram for modelling the detailed design of the dynamic behaviour. The Form Diagram allows you to generate diagram automatically with user-defined scope.

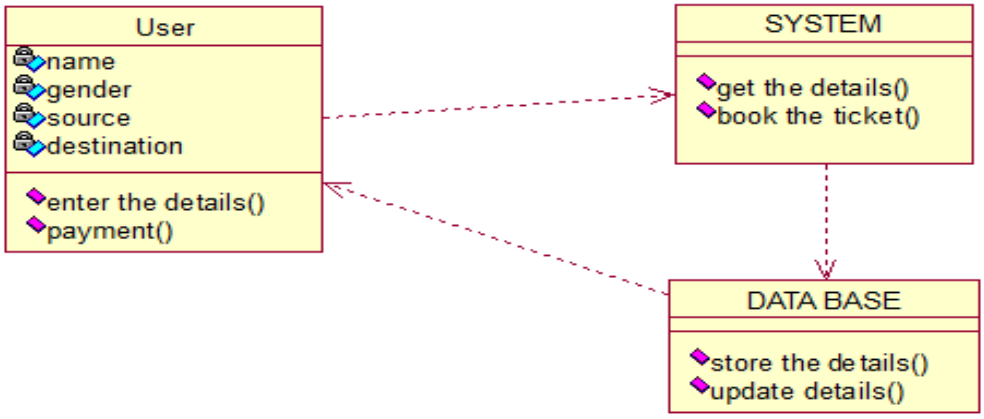

Fig 4.7: Class Diagram

Class diagram describes the attributes and operations of different actors.

Eg: For actor user the attributes are name, gender, source and destination and the operations are entering the details and payment.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION

This project deals with interaction between the user and the system through speech to book the tickets. It is an android based application. The very beginning step of our project is to enter the source, destination and number of ticket manually by the user. The data entered by the user is stored in the database for future processing. This is done to check if the number of tickets are available in the database before the user could book the ticket. Once this process is completed it leads the user to the next page using “click here to go” button where the user is provided with a mic icon which will recognize the user’s voice.

When the user clicks the mic icon the system will ask for the user’s name for which the user in turn has to respond in the form of speech. The system recognizes the user’s voice and stores the same in the database. Next, the system will ask for the source from where the user is going to start. This again would be recognized by the system and then would only be continued only if the entered source is present in the database else it would respond with an error message that the given input is an invalid source. After this, the system asks the destination the user is going to travel. The same procedure will be followed. Once the system recognizes all these inputs correctly the user gets the response that “Source to destination ticket is booked for the person user’s name” in the form of voice.

MODULES

There are different architectures for dialog systems. What sets of components are included in a dialog system, and how those components divide up responsibilities differs from system to system. Principal to any dialog system is the dialog manager, which is a component that manages the state of the dialog, and dialog strategy. A typical activity cycle in a dialog system contains the following phases:

Speech Recognition: The input to this module is the speech signal generated by the user. It recognizes the signal and converts it into text format.

Language Understanding: It receives the text generated by the speech recognition module. It analyses the text and sends the essential features to the dialogue manager.

Dialogue Management: It decides what to do or ask after it receives the input from the user. The decision of the dialogue management will have only the necessary details.

Language Generation: The necessary details from the dialog management module is framed to user conveyable text in language generator.

Speech Synthesis: The text received from the language generation module is converted to speech signal. This speech signal is forwarded to the user as a response to the request.

Graphical Interface: This helps the user to interact with the system using graphical icons rather than text.

5.2 TESTING

Testing is an important phase that focuses on an empirical investigation in which the results describe the quality of the system. It cannot confirm system functions properly under all conditions but can establish that it fails under specific conditions. The prime purpose of testing is to guarantee that system successfully built and tested in the development phase meets all the requirements and design parameters.

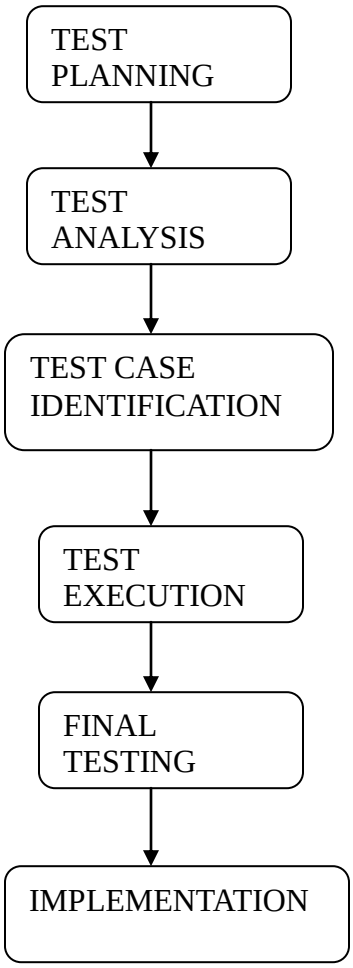

Fig 5.1 Process of Testing

5.2.1 UNIT TESTING

SPEECH TO TEXT

Table 5.1: Speech to Text

No	**Test Case**	**Expected Output**	**Observed Output**	**Result**
1	Input through voice without external noise	Display the spoken text	The spoken text is displayed	Pass
2	Input through speech with external noise	Display an error message	An error message is displayed	Pass

TEXT TO SPEECH

Table 5.2: Text to Speech

No	**Test Case**	**Expected Output**	**Observed Output**	**Result**
1	Input in the form of text	Read the entered text	Entered text is read	Pass

GRAPHICAL INTERFACE

Table 5.3: Graphical interface

No	**Test Case**	**Expected Output**	**Observed Output**	**Result**
1	Click the mic icon in online mode	Recognize the speech	Speech is recognized	Pass
2	Click the mic icon in offline mode	Display error message	Error message is displayed	Pass

DIALOG MANAGEMENT

Table 5.4: Dialog Management

No	**Test Case**	**Expected Output**	**Observed Output**	**Result**
1	Input through voice without external noise	Recognize the spoken text	The spoken text is recognized	Pass
2	Input through speech with external noise	Display an error message	An error message is displayed	Pass

5.2.2 INTEGRATION TESTING

Table 5.5: Integration testing

No	**Test Case**	**Expected Output**	**Observed Output**	**Result**
1	Input through text	Store and display data	Data is displayed and stored	Pass
2	Input through voice	Compare database and accept	Input is accepted	Pass

5.2.3 FUNCTIONAL TESTING

Table 5.6: Functional Testing

No	**Test Case**	**Expected Output**	**Observed Output**	**Result**
1	Required details are given through voice	Ticket should be booked	Ticket is booked	Pass

5.2.4 ACCEPTANCE TESTING

Table 5.7: Acceptance Testing

No	Test Case	Expected Output	Observed Output	Result
1	Tickets entered are available	Ticket should be booked	Ticket is booked	Pass
2	Tickets entered are more than availability	Ticket should not be booked	Ticket is not booked	Pass

5.3 TEST PLAN

The project is tested to verify its correctness and identify the bugs. The test plan includes the various test cases that acts as the set of conditions or variables that determine whether the corresponding feature in the system is working as it originally established to do so. When this test plan is executed, the errors spotted are rectified and the final testing yields following result.

5.4 TEST ANALYSIS

In this phase of testing, the requirements for software testing are analysed and later its feasibility is determined. In the feasibility study the possibility of project development is found through suitable test cases.

5.5 RESULT

The application is tested and found to function as expected with no errors. This application provides an interface for the users to book the ticket in an efficient way. Thus the ticket is booked as per the user’s request using speech recognition.

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The spoken dialog system provides an easy communication between the system and the user for booking tickets. It is capable of recognizing human speech and process the same to book the ticket. Moreover, the model allows us to communicate using natural language. It is more useful to the people with physical disabilities.

6.2 FUTURE ENHANCEMENT

This application can be developed in a way users can communicate with the system in different languages. The same idea can be applied in different fields such as weather forecasting, agriculture etc. This application can be adapted to various other platforms. The application can also be enhanced to process the dialog in the presence of external noise.

APPENDIX-A

SAMPLE SOURCE CODE

Admin

The below code helps us to enter into the page where the user has to give details regarding his travel.

```
package com.TicketBooking.logic;

import android.R.bool;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Settings extends Activity implements Tripinterface
{
	EditText source,destination,seat;
	Button sub;
	public void onCreate(Bundle bundle)
	{
		super.onCreate(bundle);
		setContentView(R.layout.setting);
		source=(EditText)findViewById(R.id.source);
```

```
destination=(EditText)findViewById(R.id.destination);
 seat=(EditText)findViewById(R.id.tseat);
 sub=(Button)findViewById(R.id.Submit);

  sub.setOnClickListener(new View.OnClickListener()
{
        public void onClick(View v)
        {
                try
                {
                    boolean status=true;
                  if(source.getText().toString().trim().equals(""))
                    {
                        status=false;
                        Toast.makeText(getApplicationContext(),
                        "EnterSource",
                        Toast.LENGTH_SHORT).show();
                    }
                    elseif(destination.getText().toString().trim().
                    equals(""))
                    {
                        status=false;
                      Toast.makeText(getApplicationConte
                      xt(),"EnterDestination",
                      Toast.LENGTH_SHORT).show();
```

```
                }
            else if(seat.getText().toString().trim().equals(""))
               {

                status=false;

        Toast.makeText(getApplicationContext(),"Enter Total number of seat",
                        Toast.LENGTH_SHORT).show();
                }
                if(status)
                {
                    Stringtseat=source.getText().toString().trim(
                    ).toLowerCase()+"*"+destination.getText().t
                    oString().trim().toLowerCase();
                    Tsource.put(source.getText().toString().trim(
                    ), destination.getText().toString().trim());
                    Tseat.put(tseat,Integer.parseInt(seat.getText(
                    ).toString().trim()));
                    Toast.makeText(getApplicationContext(),""
                    +Tsource, Toast.LENGTH_SHORT).show();
                    Toast.makeText(getApplicationContext(),""
+Tseat, Toast.LENGTH_SHORT).show();
Intent i=new Intent(Settings.this,First_Activity.class);
                    startActivity(i);
```

```
                    }
                }
                catch (Exception e)
                {
                    Toast.makeText(getApplicationContext(),  ""+e,
                    Toast.LENGTH_SHORT).show();}

                 }
            });
        }
}
```

Ticket Booking Activity

The below source code helps the user to interact with the system through speech to book the ticket.

```
package com.TicketBooking.logic;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.Locale;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;
public class TicketBookingActivity extends Activity implements Tripinterface
{
        private TextView txtText;
        protected static final int RESULT_SPEECH = 1;
        private ImageButton btnSpeak;
        TextToSpeech ttobj;
        ArrayList name=new ArrayList();
        String status="start";
```

```
String source="",name="";
int count=0;

LinkedHashMap Tsource=new LinkedHashMap();
LinkedHashMap Tseat=new LinkedHashMap();
public void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtText = (TextView) findViewById(R.id.txtText);
        btnSpeak = (ImageButton) findViewById(R.id.btnSpeak);
        btnSpeak.setOnClickListener(new click());
        ttobj=new TextToSpeech(getApplicationContext(), new
        TextToSpeech.OnInitListener()
        {
                public void onInit(int status)
                {
                        if(status != TextToSpeech.ERROR)
                        {
                                ttobj.setLanguage(Locale.UK);
                        }
                }
        });
}
```

```
        public class click implements View.OnClickListener
        {
                public void onClick(View v)
                {
                        try
                        {
                                if(name.equals(""))
                                {
                                        ttobj.speak("Your Name please",
TextToSpeech.QUEUE_FLUSH, null);
                                        Thread.sleep(1000);
                                        speech();
                                }
                        }
                        catch (Exception e)
                        {
                                Toast.makeText(getApplicationContext(),
""+e,Toast.LENGTH_SHORT).show();
                        }

                }

   }
   public void speech()
   {
```

```
        try
        {
                Intent intent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
                intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
"en-US");
                try
                {
                        startActivityForResult(intent, RESULT_SPEECH);
                        txtText.setText("");
                }
                catch (ActivityNotFoundException a)
                        {
                                Toast.makeText(getApplicationContext(),"Ops! Your
device doesn't support Speech to Text",Toast.LENGTH_SHORT).show();
                        }
       }catch (Exception e) {
         Toast.makeText(getApplicationContext(),
""+e,Toast.LENGTH_SHORT).show();
                }
    }
    protected void onActivityResult(int requestCode, int resultCode, Intent data)
   {
                super.onActivityResult(requestCode, resultCode, data);
                switch (requestCode)
```

```
{

        case RESULT_SPEECH:
        {
                if (resultCode == RESULT_OK && null != data)
                {
                        ArrayList<String> text =
                data.getStringArrayListExtra(RecognizerIntent.EXTRA_
                RESULTS);
                txtText.setText(text.get(0));
                if(txtText.getText().toString().trim()!=null &&
                txtText.getText().toString().trim()!="")
                {
                        if(status.equals("start"))
                        {
                                name=txtText.getText().toString().trim();
                                if(!name.equals(""))
                                {
                                        ttobj.speak("Source Please",
                                        TextToSpeech.QUEUE_FLUSH,
                                        null);
                                        try
                                        {
                                                Thread.sleep(1000);
                                                speech();
```

```
                    status="source";}
                    catch (InterruptedException e)
                    {
                    e.printStackTrace();
                }
        }

}
else if(status.equals("source"))
{
        if(Tsource.containsKey(txtText.getText().toS
tring().trim()))
        {
source=txtText.getText().toString().trim();
                ttobj.speak("Destination Please",
TextToSpeech.QUEUE_FLUSH, null);
                try
                {
                    Thread.sleep(1000);
                    speech();
                    status="Destination";
                }
                catch (InterruptedException e)
                {
                    e.printStackTrace();
```

```
                    }
                }
                else
                {
                    ttobj.speak("Invalid Source",
                TextToSpeech.QUEUE_FLUSH, null);
                try
                    {
                        Thread.sleep(1000);
                        speech();
                        status="source";
                    }
                    catch (InterruptedException e)
                    {
                        e.printStackTrace();
                    }
                }
            }
            else if(status.equals("Destination"))
            {
                String
checksoutce=Tsource.get(source.toString().trim()).toString();

                if(checksoutce.equalsIgnoreCase(txtText.get
```

```
Text().toString().trim()))
        {

            String
        merge=source.toString().toLowerCase().trim
        ()+"*"+checksoutce.toLowerCase().toString
        ().trim();
            int
        getseat=Integer.parseInt(Tseat.get(merge).to
        String().trim());
            count++;
            if(count<=getseat)
            {
                String
                booked=source+"to"+txtText.ge
                tText().toString().trim()+"Ticke
                t Booked for passanger"+name;
                ttobj.speak(booked,
                TextToSpeech.QUEUE_FLUS
                H, null)name="";
                try
                {
                    Thread.sleep(1000);
                    speech();
                    status="start";
```

```
                    }
                    catch (InterruptedException e)
                    {
                        e.printStackTrace();
                    }
                }
                else
                {
                    ttobj.speak("Sorry Seat Not
                    Available",
                    TextToSpeech.QUEUE_FLUS
                    H, null);
                }
            }
            else
            {
                ttobj.speak("Invalid Destination",
                TextToSpeech.QUEUE_FLUSH,
                null);
            }
        }
}
```

APPENDIX-B

SCREEN SHOTS

WELCOME PAGE

Fig b.1: Welcome Page

When the user enters into the application, the first page that appears is the welcome page.

ADMIN PAGE

Fig b.2: Admin Page

The above page is meant for the admin to append or clear the database.

BEFORE DATA ENTRY

Fig b.3: Before Data Entry

When the user clicks on the “settings” button, the above page would be displayed where the user has to enter the required data.

AFTER DATA ENTRY

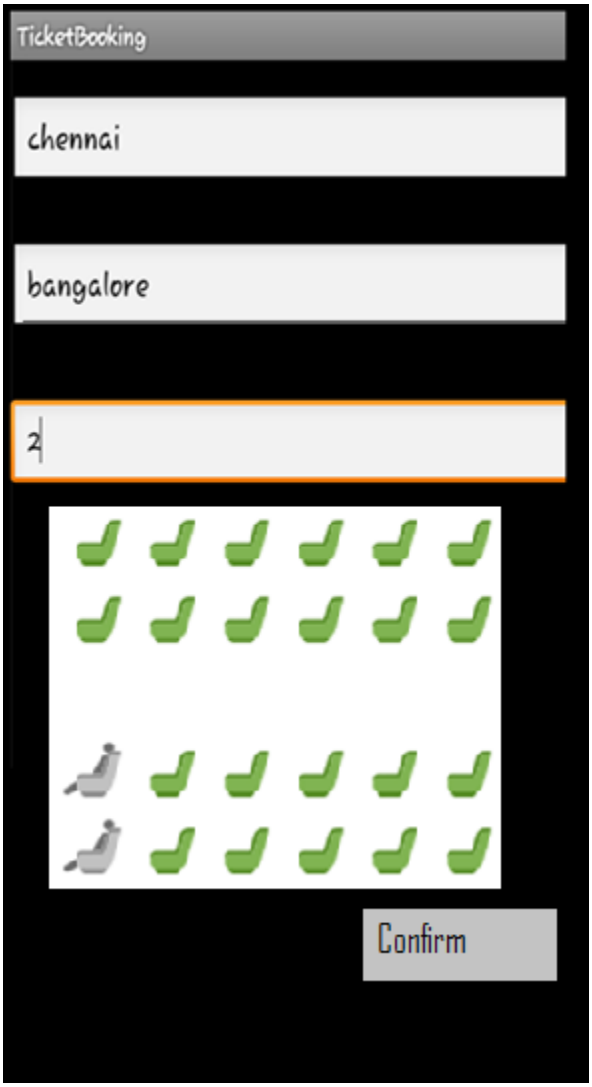

Fig b.4: After Data Entry

The user has to enter details like source, destination and the number of tickets to check the availability.

SPEECH RECOGNIZER

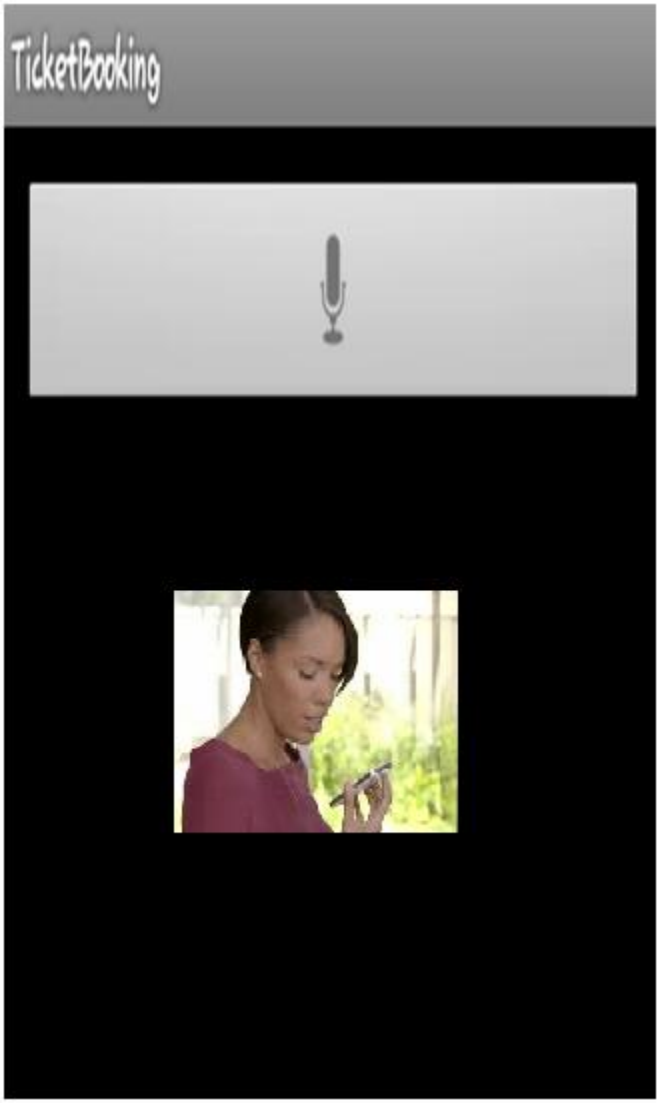

Fig b.5: Speech Recognizer

The mic icon is available here which is used to recognize the speech delivered by the user.

SPEECH RECOGNITION

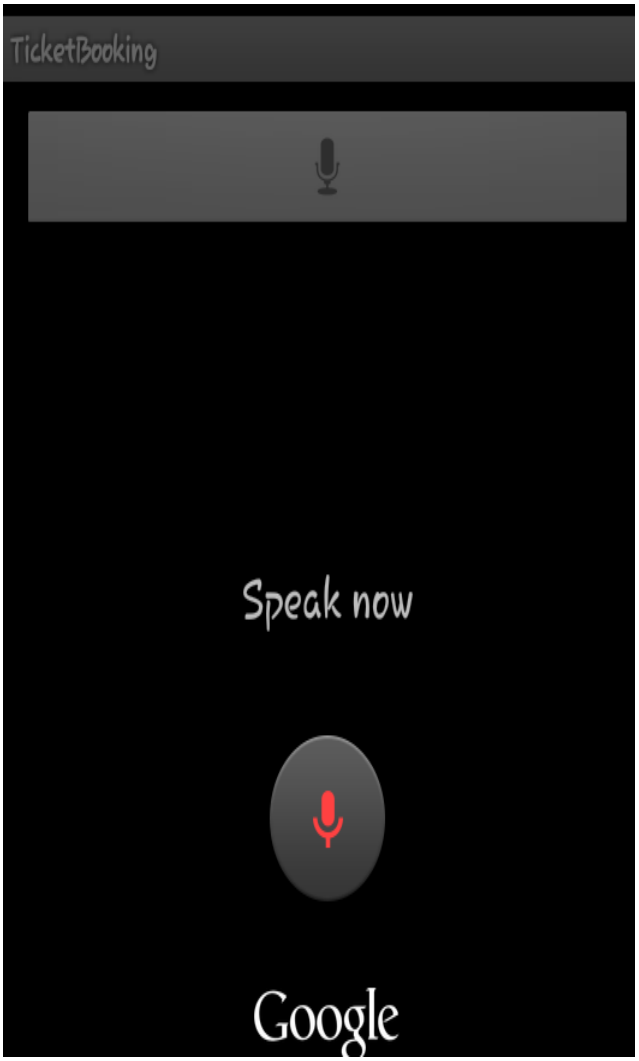

Fig b.6: Speech Recognition

The user has to click on the mic icon which allows the user to talk by giving "speak now" as the caution.

OUTPUT DISPLAY

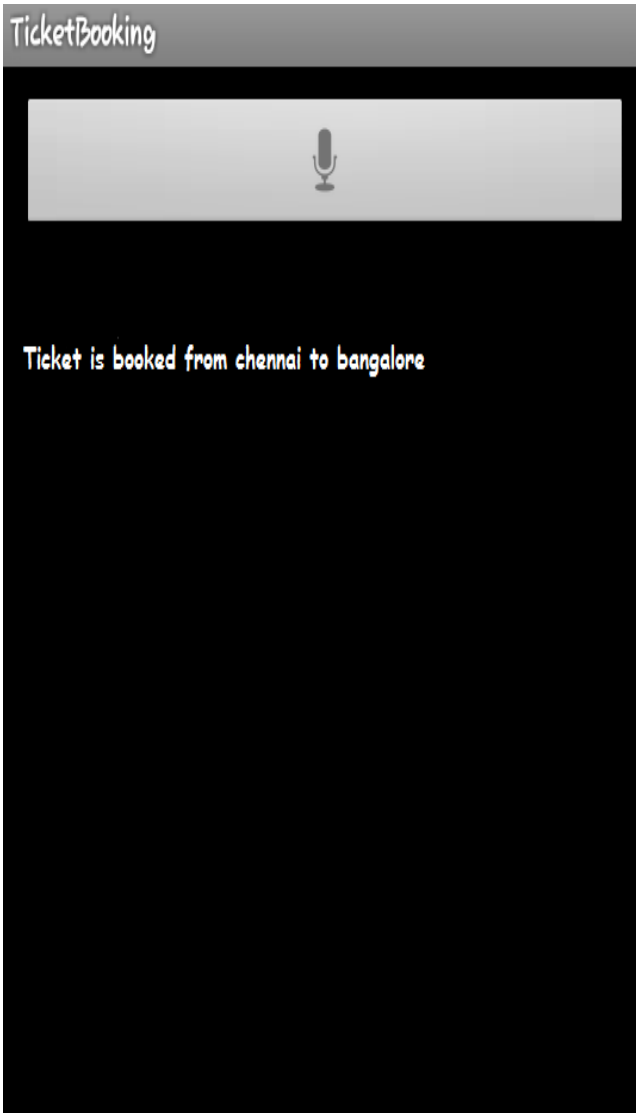

Fig b.7 Output Display

Once the process is completed, the above page shows the user that the ticket is booked successfully.

http://yourlisten.com/masterzz1394/voice-004

The above link provides the output of our project in the form of voice.

ERROR MESSAGE

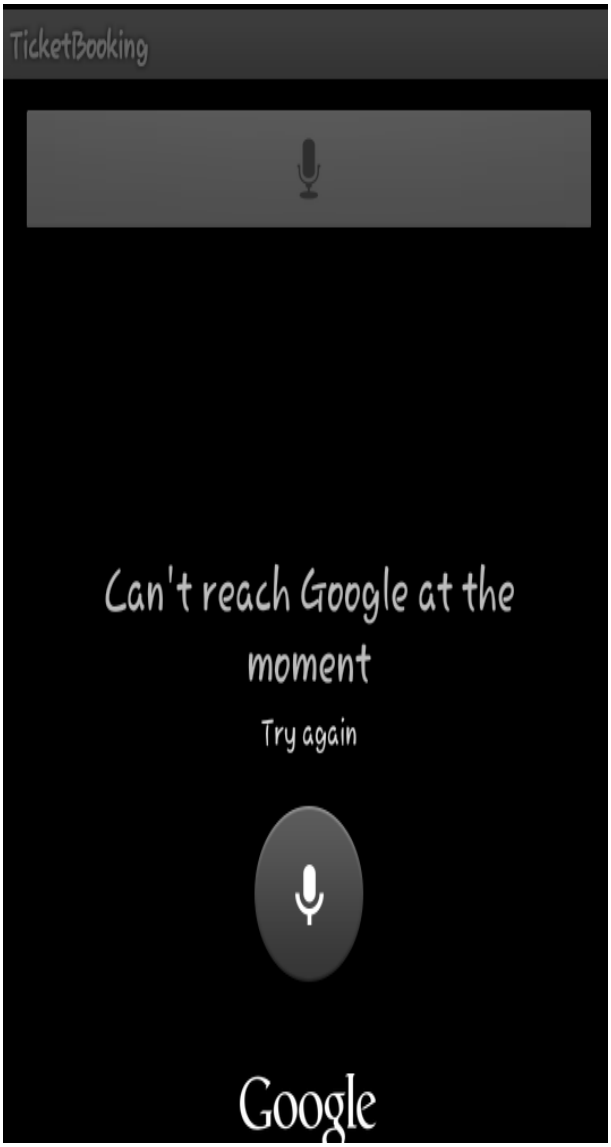

Fig b.8: Error Message

If the speech given by the user is not recognized or if there is any fault with the internet connection the above error message would be displayed.

REFERENCES

[1] Machine Learning: An Introduction by *Tom Mitchell*. Publisher: McGraw-Hill Higher Education; International edition (1 October 1997).

[2] Reinforcement Learning: An Introduction by *Richard S. Sutton & Andrew G. Barto*. A Bradford Book. The MIT Press, Cambridge, Massachusetts, London, England.

[3] Android Application Development by *Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike.* Copyright © 2009 Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike. Printed in the United States of America.Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[4] Pro Android 2 by *Sayed Y. Hashimi, Satya Komatineni, Dave MacLean.* Copyright © 2010 by Sayed Y. Hashimi, Satya Komatineni, and Dave MacLean. Printed and bound in the United States of America.

[5] Professional Android Application Development by *Reto Meier*. Published by Wiley Publishing, Inc. Copyright © 2009 by Wiley Publishing, Inc., Indianapolis, Indiana. Manufactured in the United States of America.