Project Phase-1 Report

on

A Visual Assistant Chatbot for Visually Impaired

Submitted to

NMAM INSTITUTE OF TECHNOLOGY, NITTE
(An Autonomous Institution under VTU, Belagavi)

In partial fulfillment of the requirements for the award of the

Degree of Bachelor of Engineering
in
Computer Science and Engineering

by

Pramukha R N	**4NM16CS101**
Rahul D Shetty	**4NM16CS111**
Shetty Yashas Shashidhar	**4NM16CS137**

Under the guidance of

Dr. Venugopala P S

Associate Professor

Dept. of CSE, NMAMIT, NITTE

NITTE EDUCATION TRUST | **N.M.A.M. INSTITUTE OF TECHNOLOGY**
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

CERTIFICATE

*Certified that the project work entitled **A Visual Assistant Chatbot for Visually Impaired** is a bonafide work carried out by Pramukha R N (4NM16CS101), Rahul D Shetty (4NM16CS111) and Shetty Yashas Shashidhar (4NM16CS137) in partial fulfillment for the award of Degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2019-20. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project Phase- 1 prescribed for the said Degree.*

Name & Signature of Guide   **Signature of HOD**   **Signature of the Principal**

 Dr. Venugopala P S

 Associate Professor

<u>**External Viva**</u>

   Name of the Examiners   **Signature with Date**

1. _____ _____

2. _____ _____

ACKNOWLEDGEMENT

The satisfaction that accompanies the completion of any task would be incomplete without the mention of all the people, without whom this endeavour would have been a difficult one to achieve. Their constant blessings, encouragement, guidance, and suggestions have been a constant source of inspiration

First and foremost, our gratitude to our project guide, **Dr. Venugopala P S** for his constant guidance throughout the course of this project Phase-1 and for the valuable suggestions.

We also take this opportunity to express a deep sense of gratitude to the project coordinators for their valuable guidance and support.

We acknowledge the support and valuable inputs given by **Dr. Uday Kumar Reddy** the Head of the Department, Computer Science, and Engineering, NMAMIT, Nitte.

Our sincere thanks to our beloved principal, **Dr. Niranjan N Chiplunkar** for permitting us to carry out this project at our college and providing us with all the needed facilities.

Finally, thanks to staff members of the Department of Computer Science and Engineering and our friends for their honest opinions and suggestions throughout the course of our project Phase-1.

Pramukha R N (4NM16CS101)
Rahul D Shetty (4NM16CS111)
Shetty Yashas Shashidhar (4NM16CS137)

ABSTRACT

A key challenge faced by people who are partially or completely blind is the perception and navigation of the environment that they are not accustomed to. Traveling to an unfamiliar place or merely walking down a crowded street or can be a challenge. As a consequence, many people with impaired vision travel with a friendly person or a family member while navigating an alien environment. This proxy person helps them in their navigation, describes the external environment to them and thus helping them in their external cognition. We propose a chatbot assistant who plays the role of this person. Our "Visual assistant chatbot" acts as their artificial eye describing and summarizing the external environment in real-time. Working with the chatbot does not require any specialized or technical knowledge. The user does not even need to know to operate a phone. The user just has to press the home button which is present at the bottom-center of every android phone and access the functionality of the application using just their voice.

Visually impaired people undergo many hardships regarding the perception of the external environment. The power of computer vision and natural language processing, along with cloud computing can be leveraged to alleviate their burden.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The "Visual assistant chatbot" is a simple and interactive application that is very easy to use and also doesn't require any external or extra equipment, other than a proper internet connection. It is voice-activated and also helps the user in finding out what all is included in the view range of the user. It also responds to various queries of the user. It uses Machine Learning algorithms to identify the various objects that are detected in the image that is captured in the user's smartphone. It also uses Google SDKs for Text-to-Speech and Speech-to-Text conversion. As the application is voice-command-based, it is very easy for the visually impaired users to identify and get alerted about the coming danger in their path.

This chatbot can change the lifestyle of the visually impaired people in a positive way. It can help the users to identify various surrounding objects and also to find specific objects in the viewing range of the camera. This chatbot also answers the various queries a user may ask about any image present in the smartphone, i.e. it uses a Visual Question Answering model. It also gives a summary of the contents of an input image. All these activities are done by the chatbot application without using any external equipment, just a smartphone with internet connection is sufficient for its proper working.

CHAPTER 2

LITERATURE SURVEY

For nearly a decade, manual visual question answering systems[1], [2] have been employed where a person uses a mobile application to ask a question and obtains answers through human volunteers or crowdsourcing workers. They were used for tasks like locating an object in a complex scene [3], helping the user in grocery shopping[1]. Such systems while valuable depends on manual human labor. An automated system for such a task would have great benefits like enhanced latency, cost, privacy. In 2018, Gurari et al. [4] described unique challenges facing the visual question answering systems in assisting blind people. They also created a real-world visual question answering dataset which was obtained from blind people. While constructing the dataset, prior works used dataset gathered from the web [5]–[8]. This was very tedious and required a lot of resources. The dataset was not domain-specific. Works like [9]–[11] used artificially created datasets. The artificially created dataset might inherently contain human bias which makes it drastically different from the real-world examples. The dataset taken from the web might not mimic the images taken by the blind in real-time. Hence the current VizWiz dataset solves these problems where the blind themselves have taken the pictures and asked a question. This mapping is saved and used in training.

The second class of the model that we use is for automatic image captioning. It deals with image understanding and language description of the image. Now, we review some image captioning techniques used in the current literature. Farhadi et al. [12] used a triplet of scene elements to fill the template slots for generating image captions. Kulkarni et al. [13] used a conditional random field (CRF) to infer objects, attributes, and prepositions before filling in the gaps. In our model, the template-based method can be used with the help of the answer generated by the VQA model. Such models are relatively faster. However, template-based captioning follows a pre-defined template and has fixed length captions. In later iterations, parsing-based methods[14]–[18] can be used which are more powerful than simple template-based methods. Captioning can also be done using pre-defined retrieval-based approaches. Here, a set of captions is stored in advance. While captioning, visually similar images are found the captions are retrieved from the training set.

These captions are called candidate captions and the captions for query images are retrieved from this caption pool[19]–[22]. Retrieval-based methods can be desirable as it is efficient and it can be assumed that a visually impaired person only encounters objects and scenes from a fixed set of possibilities. Such methods help in reducing communication latency and also help in lowering the cost of the model. More advanced techniques that generate novel captions can be used [23]–[25]. They analyze the visual content of the image and generate captions using a language model.

A helpful and convenient feature is an analysis of facial expressions or mood detection. Such a system can be very helpful for a visually impaired person as they are deprived of these capabilities. They generally rely on instinct and sound cues to make decisions in such cases. An image-based automated system for the detection of action units or basic and non-basic emotions can be implemented with the whole face as the region of interest. Rigid registration [26]–[29] can be used by detecting facial landmarks and using a distance metric like Euclidean distance or any affine transform that maps an input face to a prototypical face. Other non-rigid registration approaches [30] enable local registration and can suppress errors due to facial inactivity. Hence, these systems are more robust. Point registration systems [31]–[33] are needed for shape representations. They involve localization of fiducial points and localization accuracy is crucial for such systems. Systems that depend on localizing the facial features require a good quality image. Such a privilege cannot be guaranteed by our system. Hence, we depend on a rigid registration methodology for facial expression identification and mood detection.

Visual-based scene change detection is also implemented. This helps provide additional real-time assistance to the visually impaired. Such a system is also helpful in implementing additional features like warning indicators and caution system. The scene change system we are implementing only depends on visual data. No auxiliary textual or audio data is required. Certain rule-based approaches [34] use techniques like rhythm-based heuristics. Rule-based approaches only work for certain situations and have low performance. Kender and Yeo [35] introduced one of the first video segmentation algorithms. They calculated shot-to-shot coherence based on color similarity. Other works [36], [37] also used such shot similarity measures to identify scene change. Hanjalic et al. [38] used block-based similarity to measure in LUV color space. Kwon et al. [39] used motion-based features and

improved overlapping link methods. Zhao et al. [40] and Cheng and Xu [41] considered temporal distance between the two shots. Wang et al. [42] introduced an overlapping link method that uses forward and backward search. Such scene changes detection methodologies work well for our implementation as there is limited image data available.

CHAPTER 3

PROBLEM DEFINITION

One of the major challenges faced by visually impaired people is the perception of the external environment. Traveling to an unfamiliar place or even places they are accustomed to can be a challenge. This an unheard challenge every visually impaired person faces. In the era of artificial intelligence, an integrated model can be built to address this issue. This is the motive of this project. Visually impaired people need an everyday solution to address the challenges they face. Many such people rely on an auxiliary person for their navigation. This person assists the visually impaired by giving audio cues and by physically guiding them. Most people cannot afford to have a family or friendly person with them, all the time for assistance. The goal of this project is to replace this proxy person. A visual assistant chatbot is proposed which can be fully controlled using only voice. The chatbot assists the user in their daily activities. A Visual Question Answering model is implemented which is specifically trained on distorted images to replicate the behavior of a visually impaired person. The VQA model takes any question and outputs an answer with probabilities. The final answer is predicted using a language model. Additionally, an image captioning model is implemented for image captioning and summarization purposes. This model helps in summarizing the image. A facial expression and mood detection model is implemented to compensate for the lack of visual capabilities of the visually impaired. Normally, they rely on audio cues for these purposes, but with this model, those capabilities can be replicated. A visual scene change detection model is implemented. This model helps in delivering important temporal information and also warning or caution signals. A combination of all these models works seamlessly and in a well-integrated way to achieve the goal.

CHAPTER 4

SYSTEM REQUIREMENTS SPECIFICATION

4.1 OVERVIEW

The system currently under design has to take care of be suitable for a visually impaired person to use. The app should not have any small and intricate text or buttons. The buttons must be large and vibrant and most of the functionality has to be voice-controlled. The final goal is to make all of the functionality to be voice-controlled. The speech and text interconversion accuracy must be high and the model must be compatible with the accent of the user. As the application is controlled by a visually impaired person, there is a high chance that they might click the wrong button or get confused by the response. In such cases, a reset button has to be implemented. The app must work seamlessly, without any specialized knowledge.

4.2 PURPOSE

The purpose of this chapter is to present a detailed description of the Visual Assistant System. Here we explain the purpose and feature of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. We will describe the scope, objective and goal of the new system along with the non-functional components. The entire purpose of this is to build the system focusing on the end-user interaction and important functionality that needs to be provided. We provide different diagrams like use cases, sequence diagrams, and class models to better understand the different interaction with different subsystems. The major goal of the project is to develop an Android application that assists visually impaired people in their day to day activities. All the requests are sent from the application to the online server and results are sent back to the mobile.

In this chapter, we will discuss what are the various components or features provided by the final application and also the expected metrics needed to measure the performance. The implementation details on the application and server will be provided in the later chapters.

4.3 SCOPE

The project is focused on providing visually impaired people with assistance in their day to day tasks. The app aims at replacing the need for an external aid for the visually impaired by providing an audio interface for communication. The app allows the user to ask about the current scenery that is in front of the camera, ask questions related to that scene, identify friendly faces and their mood, try to extract any text from menu's or billboards, read barcodes and tell details about product and even read bills. Above all, we hope to provide a comfortable and reliable system that is available all the time for our users with minimum faults.

4.4 FUNCTIONAL REQUIREMENTS

Functional Requirements specify the important features or functional components in our application. These also describe the various elements the user can access or make use of in the system. Some of the major functionalities present in our system are stated below:

1. Capturing the audio from the microphone and converting it to text.
2. Capturing an image from the camera and sending this image and the spoken text to the server.
3. Performing the required task on the server-side and sending back the result to the application and conveying this result as audio to the user in a meaningful way.
4. Applying Visual Question Answering model on the server end to the given image and text.
5. Image captioning system to describe the current scenery.
6. Allowing users to register friendly faces on the mobile for later identification using Face Recognition.
7. Identifying text from the image and conveying its content to the user.
8. Identifying the sentiment of the person seen by the camera by using their facial features.
9. Assisting users with taking the picture by providing alignment directions.
10. A chatbot system to communicate with the user as a companion and provide audio controls for their inbuilt mobile applications.
11. A danger awareness system that describes any potential threat.

4.5 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements describe the needs that are not related to any functionality but are essential for the ease use of the application. As our application is focused to help the visually impaired, it is a must to design the system with no faults or at least be minimized. Some of the non-functional requirements identified in our system are stated below:

1. Availability: It is a must for the system to be running all the time and any crashes should be handled properly.

2. Reliability: As the users are completely dependent on our application for their tasks, the results that the system provides should be atomic in nature. Either provide a complete answer or should not.

3. Hardware Support: Our application is designed for android devices, so it should be designed in such a way that it is compatible with all the variants of this operating system with different hardware.

4. Security: The system shouldn't be exposed to any online threats or vulnerabilities as the information we are dealing with is life-critical.

5. Response Time: This factor can be measured as the duration it takes to make some computation along with the time for sending the request and receiving the result.

6. Usability: The user should be comfortable using the system and also the system should provide the user with unambiguous results.

4.6 OPERATING ENVIRONMENT

The operating environment is separated into two parts where one of them is for running the mobile application and the other one for the server. The following are the requirements for running the application on a mobile device:

- Android Smartphone for running the application.
- Camera Sensor to capture Images.
- Microphone to capture the audio.
- Audio output (Speaker/headphones) for conveying any information.
- Location sensor to access the Geo-positional data.
- Internet access for communicating with the server.

The hardware or software requirements for the server are as follows:

- Python Environment with the required libraries installed.
- Stable Internet connection and the system which has high networking capabilities.
- Storage of 20GB or more.
- Minimum of 8GB RAM.
- Intel i5 or equivalent server processor.

CHAPTER 5

SYSTEM DESIGN

5.1 USE CASE DIAGRAM

Use case diagrams represent the different scenarios where the user interacts with the system. Most of the interaction involves the user to make use of the functionalities that were mentioned in the previous chapter. Figure 5.1 shows the Use Case diagram involving five main user interactions. In this system, we have mainly two actions: Visually Impaired user and the server. The first 4 use cases are the main functionality that is needed for the end-user and the last component involves the interaction needed to store information like facial features of friends or family, any captured images or spoken texts history.
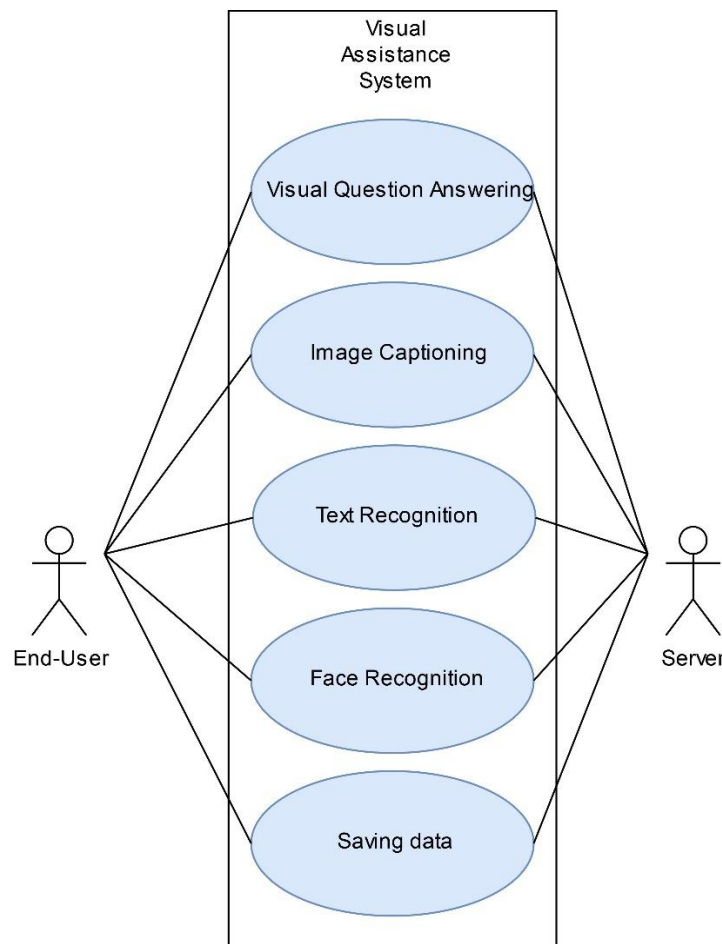

Figure 5.1: Use Case Diagram

5.2 SEQUENCE DIAGRAM

The sequence diagram provides a step by step process on how each of the components in the system interact with one another on the basis of some timestamp. In our project, we have four main entities: User, Mobile Application, Server and Model. The information is taken from the User when they click on the button and this is sent to the server where it will forward the data to the model for performing some computation. The computation can be Image Captioning, Scene Detection, Visual Question Answering, Text Recognition and so on. This decision is done based on the text. After performing the computations, the obtained result is sent back to the Mobile application which later conveys it to the user in terms of spoken audio. Figure 5.2 shows the sequence flow for the information from different components.
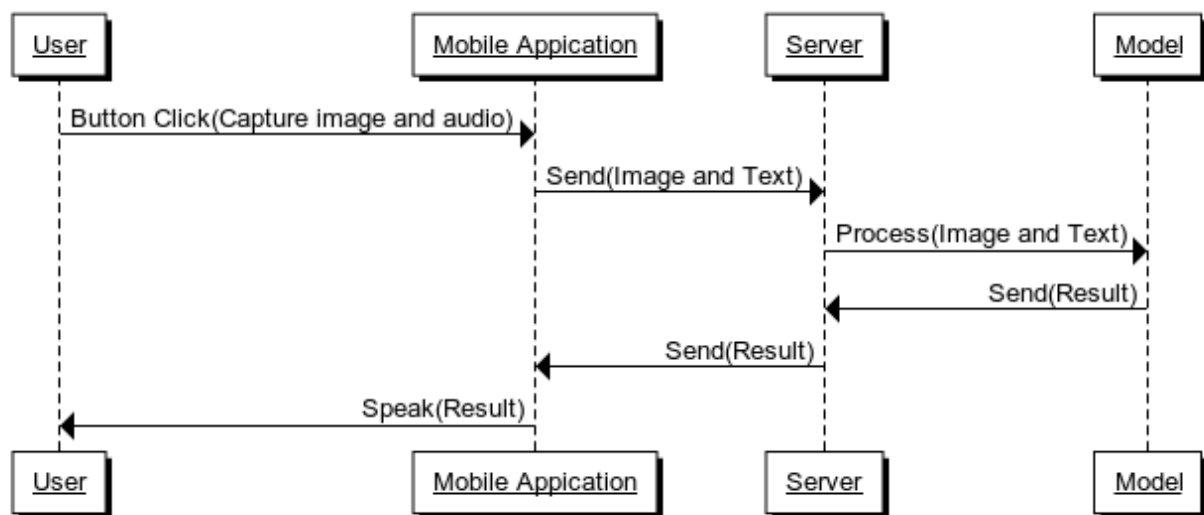

Figure 5.2: Sequence Diagram

The information is converted from one form to another as it moves along the different components. In our case, the initial audio input is converted into text by using an intermediate Speech2Text system from the Mobile Application. Similarly, the image data is converted into lower resolution so that it is faster to transmit in the network and even the further modules can easily perform processing at a faster rate without any conversion. The final result from the model is conveyed as audio output by using the Text2Speech module from the mobile device. It is to be noted that these intermediate conversions will provide a faster data transmission between the units as its easier to transmit a small image or text in the network when compared to large images or audio.

5.3 ARCHITECTURE DIAGRAM

Figure 5.3 shows the overall picture of the entire task performed by the Visual Assistance System. The mobile diagram represents the android application which reads in the current scenery from the camera sensor along with the audio input using the microphone. The audio input is converted into corresponding text and this information along with the pre-processed image is sent to the cloud server. All the heavy and major computations are performed on the server and the corresponding result is sent back to the mobile device as a text. Using the speakers or headphones the result is conveyed to the user in terms of speech.
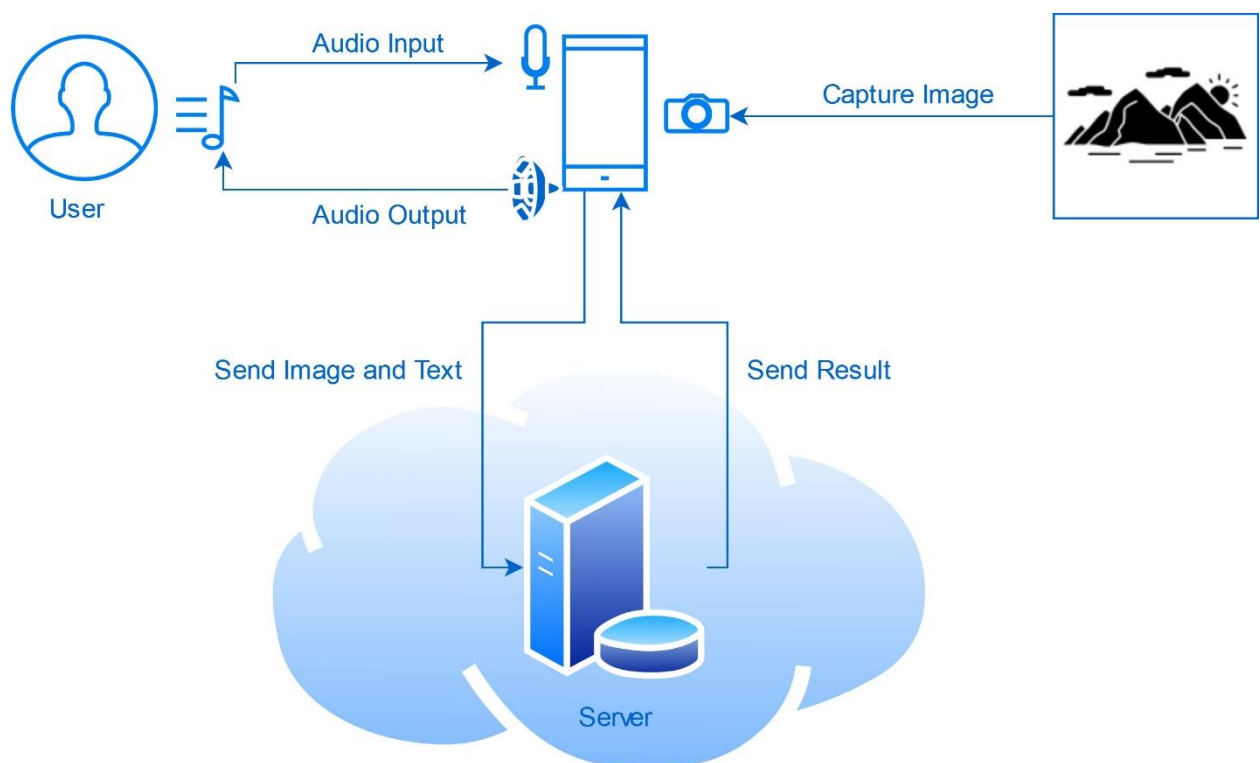

Figure 5.3: Architecture Diagram

5.4 DATA FLOW DIAGRAM

A Data Flow Diagram is a way of representing the flow of data of a process or a system. Here it gives the details about the inputs and outputs provided by each of the units present in the system. Figure 5.4 shows the Data Flow Diagram for our project.
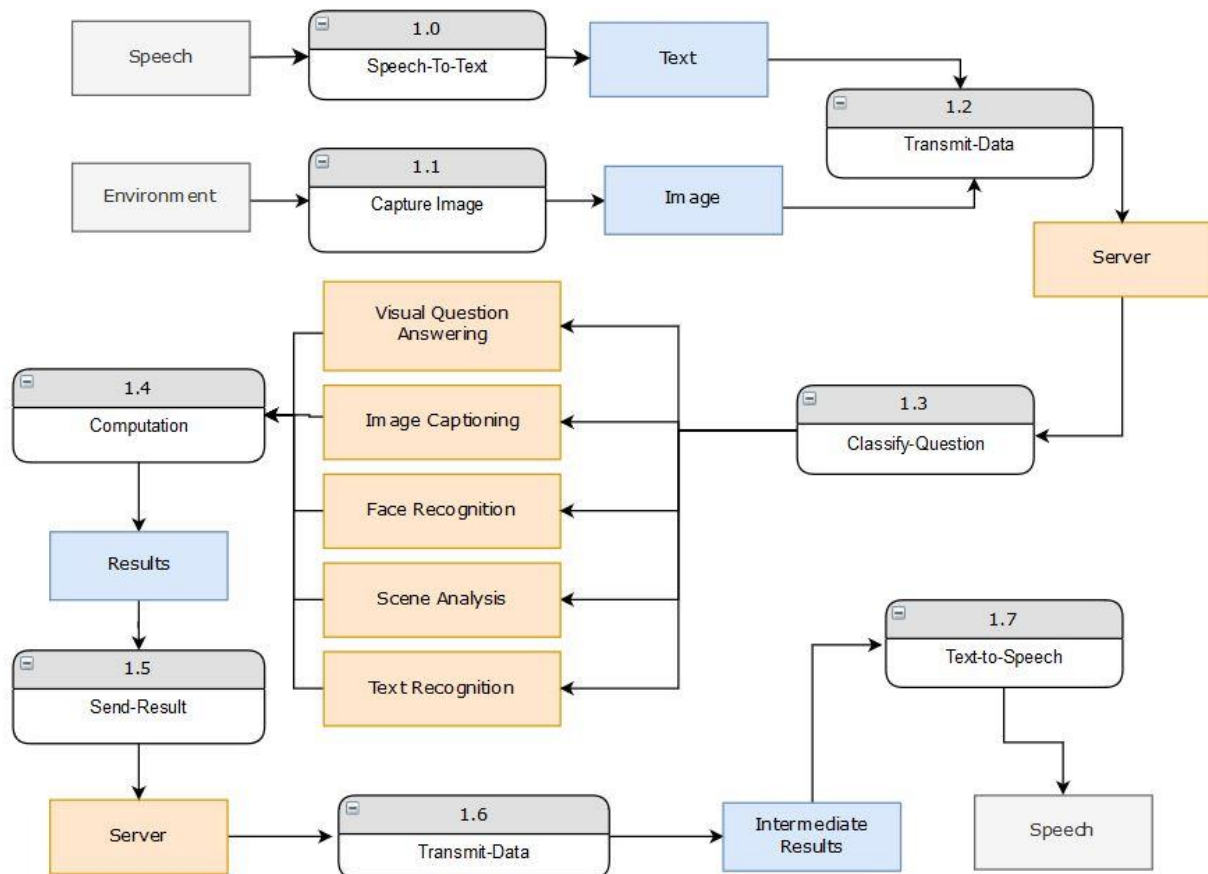
Figure 5.4: Data Flow Diagram

The indexed components in the above figure shows the processing units that takes some input and converts it to some output. The gray-colored components represent the raw data, blue colored components for the processed information. The yellow-colored components are used to denote a module.

CHAPTER 6

IMPLEMENTATION

In this chapter, we provide the implementation details for the mobile application and the server. The mobile application is designed and developed using Android Studio with Java as its programming language. The app provides a simple layout with two large buttons, a text view and two image views for now. One of the buttons is mainly used to perform the actual task and another one to refresh the feed and to set the address of the server. The mobile application makes use of Google's Speech-to-Text API which is present in the android devices to convert the spoken audio to text. For using this API we need to create Implicit Intent and start this for obtaining the results. Some parameters are passed based to set the language model. The code snippet shown below is used to call this API.

```java
// Calling the Google's Speech-to-Text API
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,Locale.getDefault());
try {
    startActivityForResult(intent, 100);
}
catch (ActivityNotFoundException a) {

}
```

Along with the above action, the current image as shown by the camera preview is captured. To perform this action, we have to implement a Picture Callback action that takes the picture when the *takePicture()* method is called. The following code snippet below shows how to perform the image capture.

```java
// captrue the image from camera
camera.takePicture(null,null,mPicture);

// callback method
private Camera.PictureCallback mPicture = new Camera.PictureCallback()
{
      @Override
      public void onPictureTaken(byte[] data, Camera camera) {
        // image is available as byte stream in data
        photo = BitmapFactory.decodeByteArray(data,0,data.length);
    }
};
```

The given image is a high-quality picture, sending this directly to the server could be cumbersome. To overcome this, we are reducing the quality of the image, by a certain factor. The resultant image was rotated and flipped about the vertical and horizontal axis. So we perform a Rotation by 90* in an Anti-Clockwise Direction to show the result to the user. For handing the flipping, we leave the implementation to the server end. The following code snippet shown below describes the above actions on a photo.

```java
// Reducing the Quality of Image
ByteArrayOutputStream stream = new ByteArrayOutputStream();
photo.compress(Bitmap.CompressFormat.JPEG,80,stream);

// Rotating the image by 90* Anti-Clockwise
byte[] byteArray = stream.toByteArray();
photo = BitmapFactory.decodeByteArray(byteArray,0,byteArray.length);
photo = Utils.rotateImage(photo,90);

// Getting the Bitmap object from the byte stream
Bitmap bmp = photo;
ByteArrayOutputStream new_stream = new ByteArrayOutputStream();
bmp.compress(Bitmap.CompressFormat.PNG, 100, stream);
img_stream = stream.toByteArray();

// Showing the results in the Image View
output.setImageBitmap(photo);
```

Any results obtained from the app are conveyed to the user through means of speech. Android has a built-in Speech Engine that is capable of generating different audio speech, just by specifying the text that we want to convey and setting up some parameters. The android package includes a *tts.TextToSpeech* class that performs this action. To use this class, we have written a helper class, that acts as a mediator between our application and this package. The code snippet given below shows some of the important aspects of this class.

```java
// Intialization with Parameters
tts = new TextToSpeech(context, new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if(status!=TextToSpeech.ERROR)
        {
        // Initializing Parameters
        tts.setLanguage(Locale.UK);
        tts.setSpeechRate(0.9f);
        }
    }
});
// Using the tts object to generate Speech
tts.speak("STRING",TextToSpeech.QUEUE_FLUSH,null);
```

The Speech-to-Text API generates different possible results that match the corresponding audio. These results can be accessed through the *onActivityResult()* method. Here a parameter *intent* contains the data that is sent back from an external application or activity. In our case, this is a list of text that match the audio. The string are arranged in the decreasing order of their match probabilities, so we consider the first item for our predictions. The above details are mentioned in the code below:

```java
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == RESULT_OK && requestCode == 100 && data != null){

            // Get all matched strings
        ArrayList<String> result =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

            // take the first item
            requestString = result.get(0);


    }
}
```

To transmit the data to the server, we make use of OkHTTP which is an HTTP client. We can pack the items in a Request object and pass it through this client to the server by specifying the IP address of that server. The server will process the data and send back the result. We can set a callback method to this action to convey the results to the user in terms of speech. The code given below describes the above scenario:

```java
// Pack the image inside the Request body
RequestBody postBodyImage = new
MultipartBody.Builder().setType(MultipartBody.FORM)

      .addFormDataPart("image", "android.jpg",
RequestBody.create(MediaType.parse("image/*jpg"), img_stream))
      .addFormDataPart("query", requestString)
                            .build();

// Instantiate OkHttp Client
OkHttpClient client = new OkHttpClient();

// Build a http request
Request request = new Request.Builder()
                .url(postUrl)
                .post(postBody)
                .build();
```

```java
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        // Cancel the post on failure.
        call.cancel();
        System.out.println("FAILED"+ e.getMessage());
    }
    @Override
    public void onResponse(Call call, final Response response) throws
IOException {
    // In order to access the TextView inside the UI thread, the code is
executed inside runOnUiThread()
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    try {
                        responseString = response.body().string();
                        // speak result
                        tts.speak(responseString);
                    } c
                    catch (IOException e) {
                        e.printStackTrace();
                        tts.speak("Failed to contact the server");
                    }
                }
            });
    }
});
```

The *onResponse()* method defines the action that is performed when the result is sent from the server-side. The result is stored in *responseString* object and then passed to the Text-to-Speech module for reading out the text.

The application on the server-side is written in Python with different modules like Flask, Tensorflow, scikit-image and so on. Out of these, Flask is the module that allows us to write the server hosting code. The below code snippet shows on using the Flask object. The user can visit the URL '*localhost:3999/*' to request the method *home()*.

```python
from flask import *
app = Flask(__name__)

@app.route('/')
def home():
    print("Req hello")
    return "Hello world"

if __name__ == "__main__":
    app.run(host= '0.0.0.0',debug = True,port=3999)
```

As we are sending images and text from the mobile application, we need to capture them at the server end. We can access these items from the Flask request object. These items are then passed to the predictor to make a decision. In the current implementation, we have integrated the Image Captioning model that takes the image and generates the summary. The overall process of this is shown in the code below:

```python
@app.route('/request',methods=["POST"])
def req():
    print("Requesting...")
    # Get image
    if 'image' in request.files:
        image_file = request.files['image']
    if 'query' in request.files:
        query = request.files['query']
    filename = werkzeug.utils.secure_filename(image_file.filename)
    image_file.save(filename)

    # Rotate and Flip image
    img = io.imread(filename)
    img = transform.rotate(img,angle=90)
    img = np.flipud(img)
    img = np.fliplr(img)
    io.imsave(filename,img)


    # Generate Caption
    res = predict(filename)
    return res
```

The image captioning model is designed by using a deep learning library, Tensorflow. The architecture of this neural network makes use of a pre-trained Convolutional Neural Network to extract image features. These features are passed to the Recurrent Neural Network which uses an Attention mechanism to come up with the caption for the given image.

The CNN part of the model acts as an Encoder while the RNN part as a Decoder. The entire network is trained on an annotation dataset MS-COCO. It contains thousands of images and some set of captions for each of them. RNN makes use of a vocabulary to formulate the words. A Tokenizer is used and its data is stored in persistent storage by using a pickle package. After the training process, we are saving the weights of encoder and decoder, so that we can reuse them to make predictions without having to train again. The implementation details for both the models are given in the next section.

```python
// Model code
class BahdanauAttention(tf.keras.Model):
    def __init__(self, units):
        super(BahdanauAttention, self).__init__()
        self.W1 = tf.keras.layers.Dense(units)
        self.W2 = tf.keras.layers.Dense(units)
        self.V = tf.keras.layers.Dense(1)

    def call(self, features, hidden):
        hidden_with_time_axis = tf.expand_dims(hidden, 1)
        score = tf.nn.tanh(self.W1(features) +
self.W2(hidden_with_time_axis))
        attention_weights = tf.nn.softmax(self.V(score), axis=1)
        context_vector = attention_weights * features
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, attention_weights

class CNN_Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(CNN_Encoder, self).__init__()
        self.fc = tf.keras.layers.Dense(embedding_dim)

    def call(self, x):
        x = self.fc(x)
        x = tf.nn.relu(x)
        return x

class RNN_Decoder(tf.keras.Model):
    def __init__(self, embedding_dim, units, vocab_size):
        super(RNN_Decoder, self).__init__()
        self.units = units
        self.embedding = tf.keras.layers.Embedding(vocab_size,
embedding_dim)
        self.gru = tf.keras.layers.GRU(self.units,
                                return_sequences=True,
                                return_state=True,

recurrent_initializer='glorot_uniform')
        self.fc1 = tf.keras.layers.Dense(self.units)
        self.fc2 = tf.keras.layers.Dense(vocab_size)
        self.attention = BahdanauAttention(self.units)

    def call(self, x, features, hidden):
        context_vector, attention_weights = self.attention(features,
hidden)
        x = self.embedding(x)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)
        output, state = self.gru(x)
        x = self.fc1(output)
        x = tf.reshape(x, (-1, x.shape[2]))
        x = self.fc2(x)
        return x, state, attention_weights

    def reset_state(self, batch_size):
        return tf.zeros((batch_size, self.units))
```

CHAPTER 7

RESULTS AND DISCUSSION

The project contains two main components: Server and Mobile Application. The mobile application is designed using Android Studio and the code base is written in Java. This app is responsible for capturing the audio, images from the user and convey the information to the server and later produce the result using Speech. To convert the audio speech into the text we make use of Google's Speech-To-Text API. This feature is available in all the Android devices and can convert audio to text with good accuracy. As the audio is being converted, we also capture the image from the camera sensor. This is needed for making the predictions that involve visual questions, image captioning and such tasks. Figure 7.1 shows the working of Google API to capture the audio signal into text.
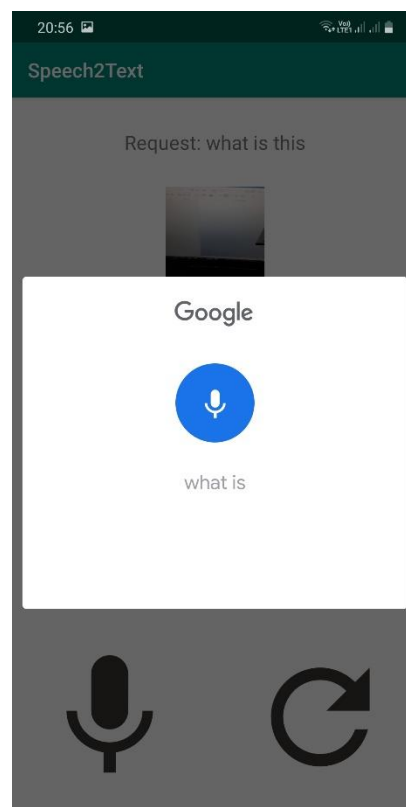

Figure 7.1: Speech-to-Text

Our server part is implemented in Python by using the Flask module. Flask is a micro-web server framework used to host dynamic webpages. Flask acts as a middleware between our model and the app. The mobile application makes use of OkHttp Http Client to make a server request. While making the request, the client

sends in additional parameters specifying the details about the spoken text and the current image. As of now, we have implemented the interface for image captioning, so the image is sent to this image captioning model and the obtained result is sent back as a response to the app.
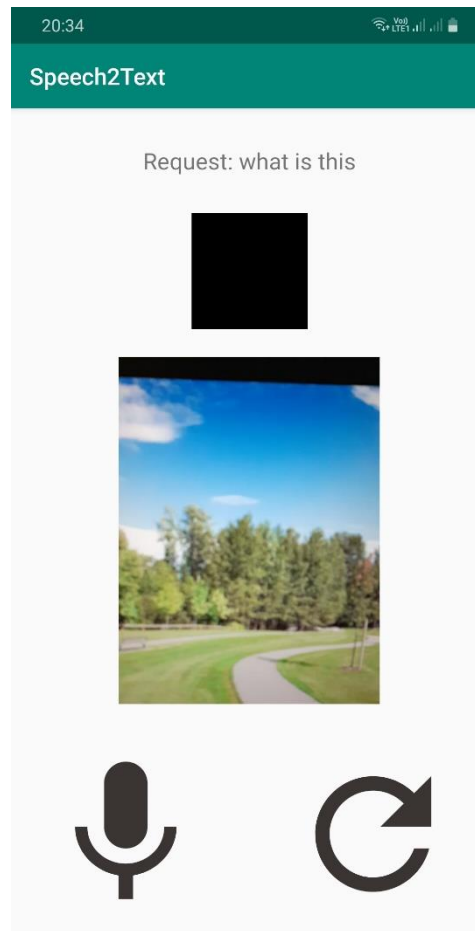

Figure 7.2: Application Screen

Figure 7.2 shows a sample demo of our application. On clicking the microphone icon, the user can speak to the phone and also the image that is seen as a preview in the camera view is captured. For the purpose of testing, we are hosting the server locally on the computer, so we need to specify the address for the connection request. On clicking the Refresh icon, the application opens a window to enter the host IP Address. This is illustrated in Figure 7.3 where a text field is available for the user to enter the address.
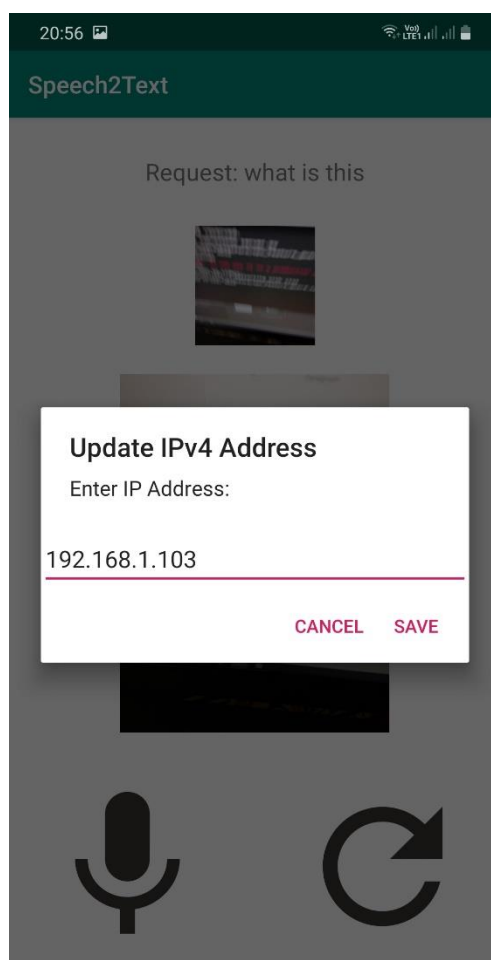
Figure 7.3: IP Address Connectivity

The application also refreshes the camera feed to refresh the resources which were allocated. After capturing the complete audio, the API provides multiple approximations and we are selecting the top 1 as our result. The spoken text, along with the processed image is sent to the URL specified at the above field by packing it all as an OkHTTP Request Object. The image is converted to bytes and are streamed on to the server. At the server-side, the first task is to decode this stream of bytes into the image and get the spoken text. Then we provide this image as an input to the Image Captioning model for generating the summary of the scene.

The solution to the image captioning problem is solved by creating a deep learning model that takes in the image and generated the caption. Here we used the Tensorflow Deep learning library to train and create the required model. The model architecture is shown in Figure 7.4 below. The input to our model is an image that is given to the Convolutional Neural Network for extracting features. These features are given to the Gated Recurrent Unit (GRU) for generating the words. In order to generate a better caption, we make use of the technique known as Attention. This

mechanism focuses on the required parts of the image individually to generate individual words. This technique is found to be most useful in different Deep learning problems.
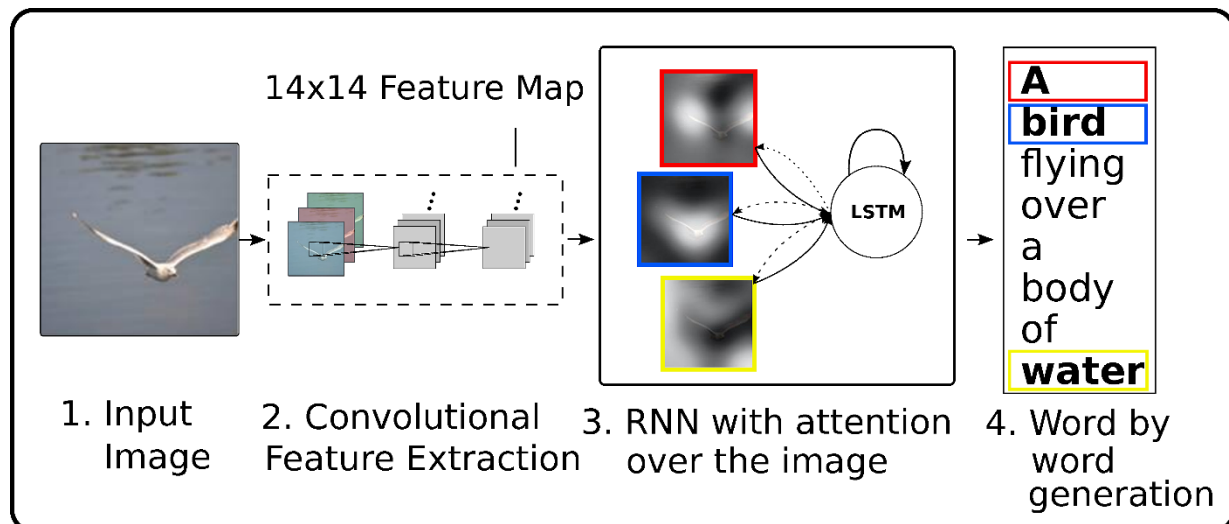

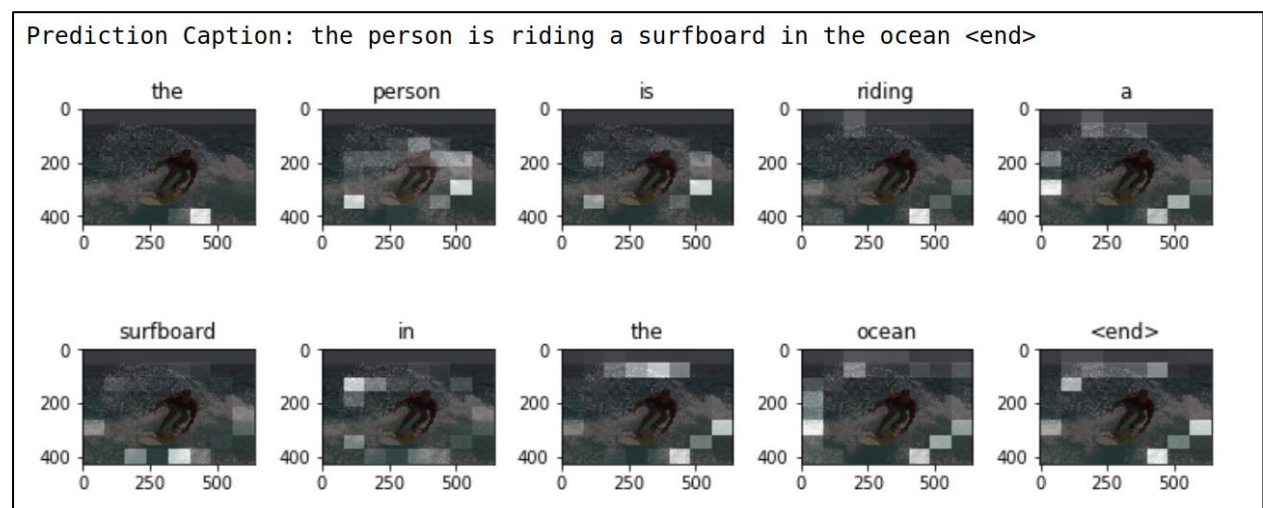Figure 7.4: Image Captioning Architecture Diagram

Figure 7.5: Image Captioning using Attention

Figure 7.5 shows a sample of how the neural network generated individual words by using the Attention Mechanism. Any neural network model needs to be trained on some dataset to produce any good result. In our case, we made use of the MS-COCO dataset, which is popular for this application as it contains annotations and image captions for a huge set of images.

The model is trained using Google Colab, which is an online Notebook Editor similar to Jupyter. The implementation of this is done in Python and Tensorflow. After training the model, we are saving the weights of CNN encoder and RNN decoder, vocabularies, and other required data so that we can reuse for making predictions.

Then we created a simple interface to communicate with this model so that we can integrate it with the server code. Figure 7.6 shows the output for the given image of the Giraffe which was sent to the server. The model provided with the response and spoken it to the user in terms of speech.
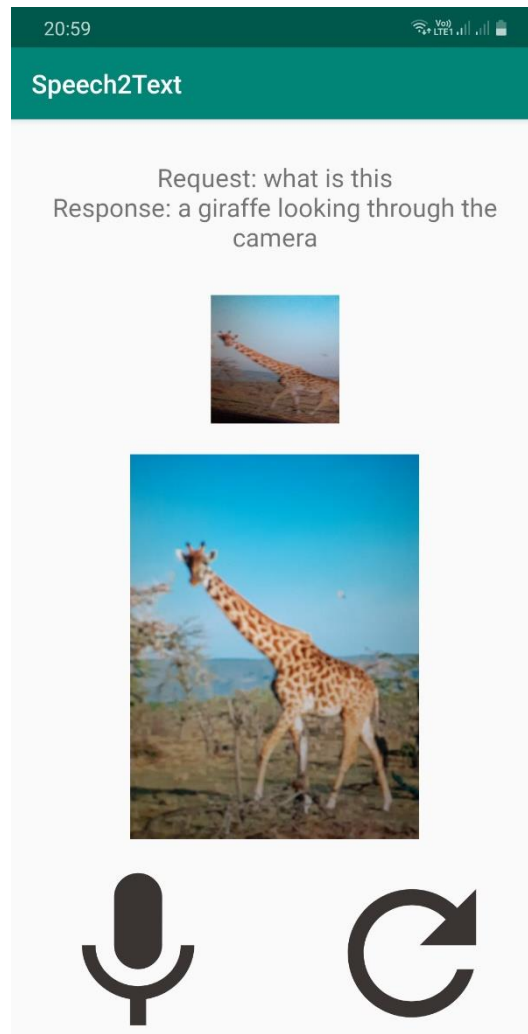

Figure 7.6: Image Captioning output in the application

The base app with Speech-to-Text and Text-to-Speech functionality has been implemented. Additionally, the image captioning feature has been implemented. An elegant graphical user interface with large icons has been provided. This improves visibility and ease of use for people with partial vision. The whole app can also be accessed using only speech as defined previously. All the processing takes place in a web app that runs remotely on a server. The whole process of image capturing, sending the image to server, processing and sending back appropriate results is implemented and works seamlessly. Visual Question Answering model has been implemented as tested as a standalone application and is yet to be integrated into

the main application. We further plan on integrating all the parts and build a suitable language model to frame sentences. The whole model is planned to be hosted remotely in the cloud.

CHAPTER 8

CONCLUSION AND FUTURE WORK

The basic functioning of the chatbot and application and the backend has been implemented. This chatbot can be accessed through voice for completely blind people and using simple keys for people with partial vision. Functionalities like image captioning, Text-to-Speech and Speech-to-Text have been implemented in the main app. We further plan on integrating other models and increasing the efficiency of the model.

The app is designed to be a useful utility application for the visually impaired. We also take care of ethical responsibilities as there is a transfer of personal data involved. The app does not store any data and further security measures are planned to be implemented at the user and server-side. We also plan to test the app in a real-world setting and practical features using the current models and add additional models if required.

As for future work, we plan to implement the following

- Integrating the Visual Question Answering model into the application.
- Implementing a Facial expression or mood detection module into the application. This sytem uses a global face detection system instead of a specialized local feature detection system.
- Implement a database for familiar faces like family members so that they can be recognized instantly and for better user experience.
- Implementing an scene change detection system. This system can be comparatively more challenging as we require a steady stream of video data to be sent to the the server through a continuous network.
- Using the above modules, implementing an danger warning system which gives warning to the user to alert them of any potential danger.
- Implementing an text recognition system which reads a block of text when requested by the user. This system requires more clear and steady image when compared to other systems.
- Determining a pipeline for the modules so that they work and interact with each other seamlessly, without any additional delays and minimum overhead.
- Testing the application in an real-world setting to determine its feasibility.

References

[1]   J. P. Bigham *et al.*, "VizWiz: Nearly real-time answers to visual questions," in *UIST 2010 - 23rd ACM Symposium on User Interface Software and Technology*, 2010.

[2]   W. S. Lasecki, P. Thiha, Y. Zhong, E. Brady, and J. P. Bigham, "Answering visual questions with conversational crowd assistants," in *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2013*, 2013.

[3]   J. P. Bigham, C. Jayant, A. Miller, B. White, and T. Yeh, "VizWiz::LocateIt - Enabling blind people to locate objects in their environment," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, 2010.

[4]   D. Gurari *et al.*, "VizWiz Grand Challenge: Answering Visual Questions from Blind People," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[5]   T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[6]   J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[7]   G. Patterson and J. Hays, "SUN attribute database: Discovering, annotating, and recognizing scene attributes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012.

[8]   O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, 2015.

[9]   S. Antol *et al.*, "VQA: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[10]   J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural module networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[11]   J. Johnson, L. Fei-Fei, B. Hariharan, C. L. Zitnick, L. Van Der Maaten, and R. Girshick, "CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[12]   A. Farhadi *et al.*, "Every picture tells a story: Generating sentences from images," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010.

[13]   G. Kulkarni *et al.*, "Baby talk: Understanding and generating simple image descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013.

[14]   M. Mitchell *et al.*, "Midge: Generating image descriptions from computer vision detections," in *EACL 2012 - 13th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings*, 2012.

[15]   P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi, "Collective generation of natural image descriptions," in *50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference*, 2012.

[16]   P. Kuznetsova, V. Ordonez, T. L. Berg, and Y. Choi, " Т ree Т alk : Composition and Compression of Trees for Image Descriptions ," *Trans. Assoc. Comput. Linguist.*, 2014.

[17]   A. Aker and R. Gaizauskas, "Generating image descriptions using dependency relational patterns," in *ACL 2010 - 48th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 2010.

[18]   D. Elliott and F. Keller, "Image description using visual dependency representations," in *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2013.

[19]   C. Sun, C. Gan, and R. Nevatia, "Automatic concept discovery from parallel text and visual corpora," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[20]   V. Ordonez, G. Kulkarni, and T. L. Berg, "Im2Text: Describing images using 1 million captioned photographs," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 2011.

[21]   M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," in *IJCAI International Joint*

Conference on Artificial Intelligence, 2015.

[22] Y. Gong, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik, "Improving image-sentence embeddings using large weakly annotated photo collections," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[23] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[24] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei, "Boosting Image Captioning with Attributes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[25] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *32nd International Conference on Machine Learning, ICML 2015*, 2015.

[26] B. Jiang, M. Valstar, B. Martinez, and M. Pantic, "A dynamic appearance descriptor approach to facial actions temporal modeling," *IEEE Trans. Cybern.*, 2014.

[27] G. C. Littlewort, M. S. Bartlett, and K. Lee, "Automatic coding of facial expressions displayed during posed and genuine pain," *Image Vis. Comput.*, 2009.

[28] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1998.

[29] G. Tzimropoulos, V. Argyriou, S. Zafeiriou, and T. Stathaki, "Robust FFT-based scale-invariant image registration with image gradients," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.

[30] S. Yang and B. Bhanu, "Facial expression recognition using emotion avatar image," in *2011 IEEE International Conference on Automatic Face and Gesture Recognition and Workshops, FG 2011*, 2011.

[31] M. Valstar, B. Martinez, X. Binefa, and M. Pantic, "Facial point detection using boosted regression and graph models," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[32] D. Vukadinovic and M. Pantic, "Fully automatic facial feature point detection using gabor feature based boosted classifiers," in *Conference Proceedings -*

IEEE International Conference on Systems, Man and Cybernetics, 2005.

[33] M. F. Valstar and M. Pantic, "Fully automatic recognition of the temporal phases of facial actions," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, 2012.

[34] B. Adams, C. Dorai, and S. Venkatesh, "Toward automatic extraction of expressive elements from motion pictures: Tempo," *IEEE Trans. Multimed.*, 2002.

[35] J. R. Kender and B. L. Yeo, "Video scene segmentation via continuous video coherence," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998.

[36] Z. Rasheed and M. Shah, "Scene detection in Hollywood movies and TV shows," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[37] Y. Rui, T. S. Huang, and S. Mehrotra, "Constructing table-of-content for videos," *Multimed. Syst.*, 1999.

[38] A. Hanjalic, R. L. Lagendijk, and J. Biemond, "Automated high-level movie segmentation for advanced video-retrieval systems," *IEEE Trans. Circuits Syst. Video Technol.*, 1999.

[39] Y. M. Kwon, C. J. Song, and I. J. Kim, "A new approach for high level video structuring," in *IEEE International Conference on Multi-Media and Expo*, 2000.

[40] L. Zhao, S. Q. Yang, and B. Feng, "Video scene detection using slide windows method based on temporal constrain shot similarity," in *Proceedings - IEEE International Conference on Multimedia and Expo*, 2001.

[41] W. Cheng and D. Xu, "A novel approach of generating video scene structure," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2003.

[42] X. Wang, S. Wang, H. Chen, and M. Gabbouj, "A shot clustering based algorithm for scene segmentation," in *Proceedings - CIS Workshops 2007, 2007 International Conference on Computational Intelligence and Security Workshops*, 2007.