**CHAPTER 4**

**SYSTEM REQUIREMENTS SPECIFICATION**

4.1 OVERVIEW

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.

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.

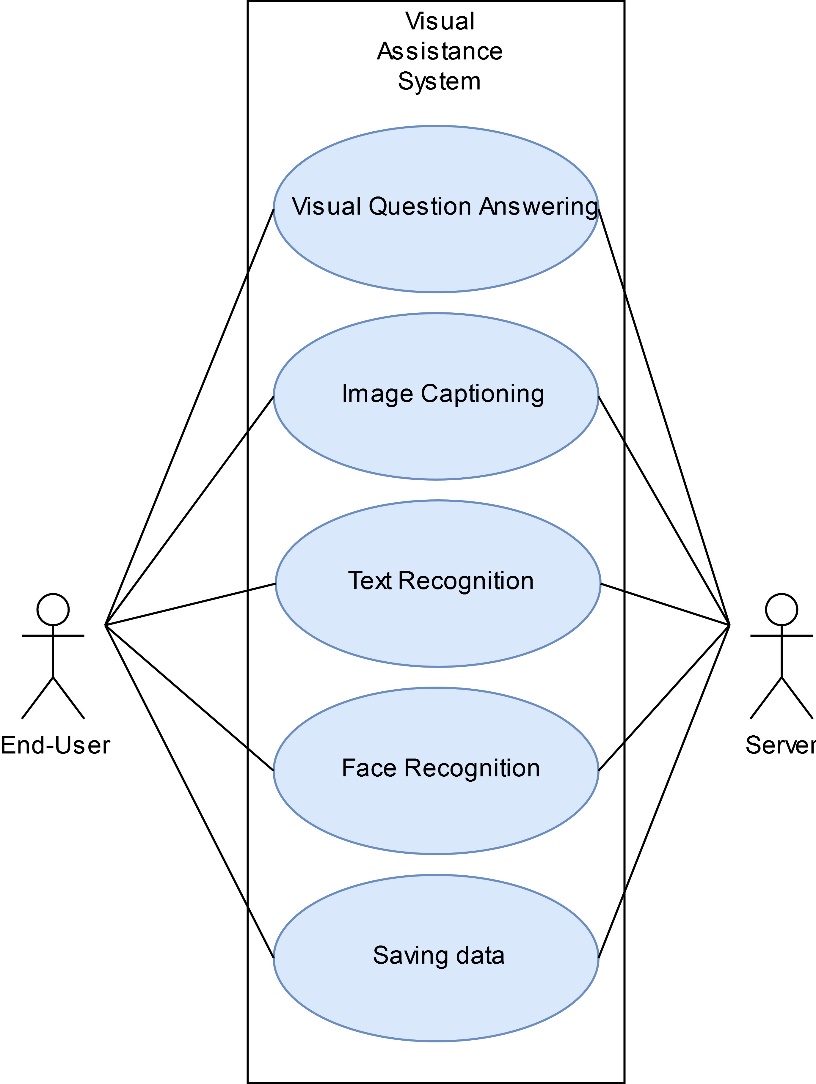


Figure 5.1: Use Case Diagram

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, 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.

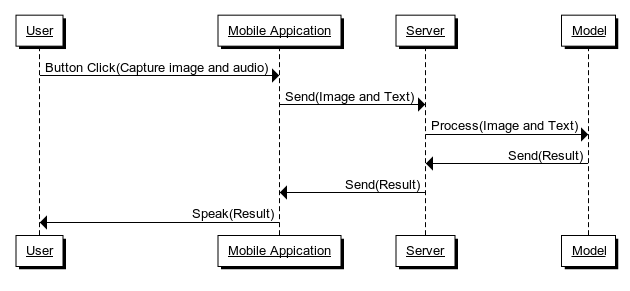


Figure 5.2: Sequence Diagram

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 conversion 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.

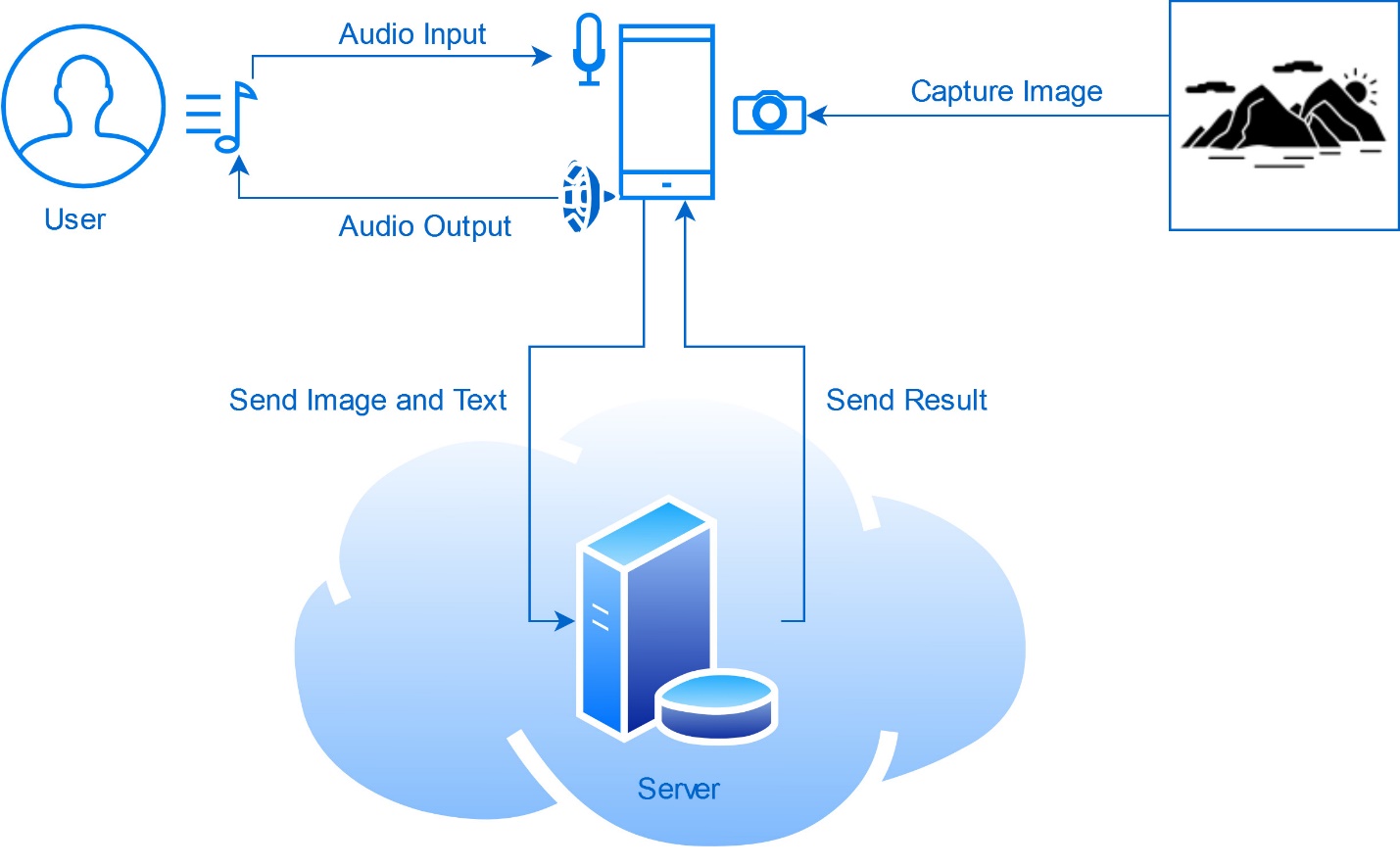


Figure 5.3: Architecture Diagram

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.

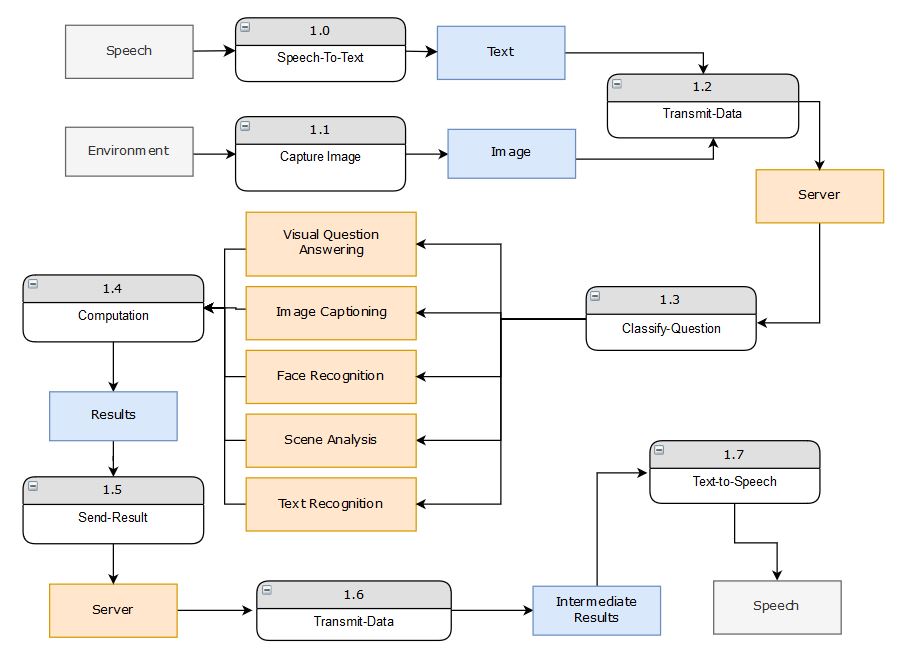


Figure 5.4: Data Flow Diagram

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**

6.1 OVERVIEW

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 6.1 shows the working of Google API to capture the audio signal into text.

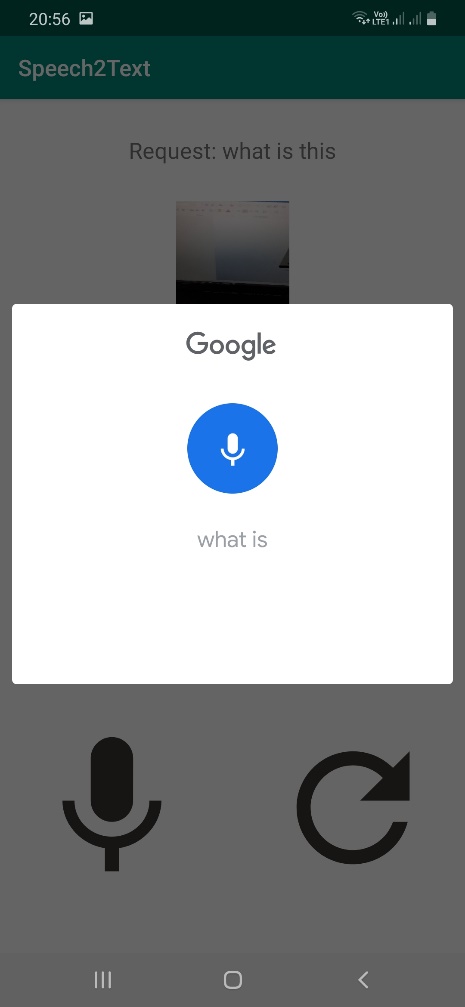


Figure 6.1: Data Flow Diagram

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.

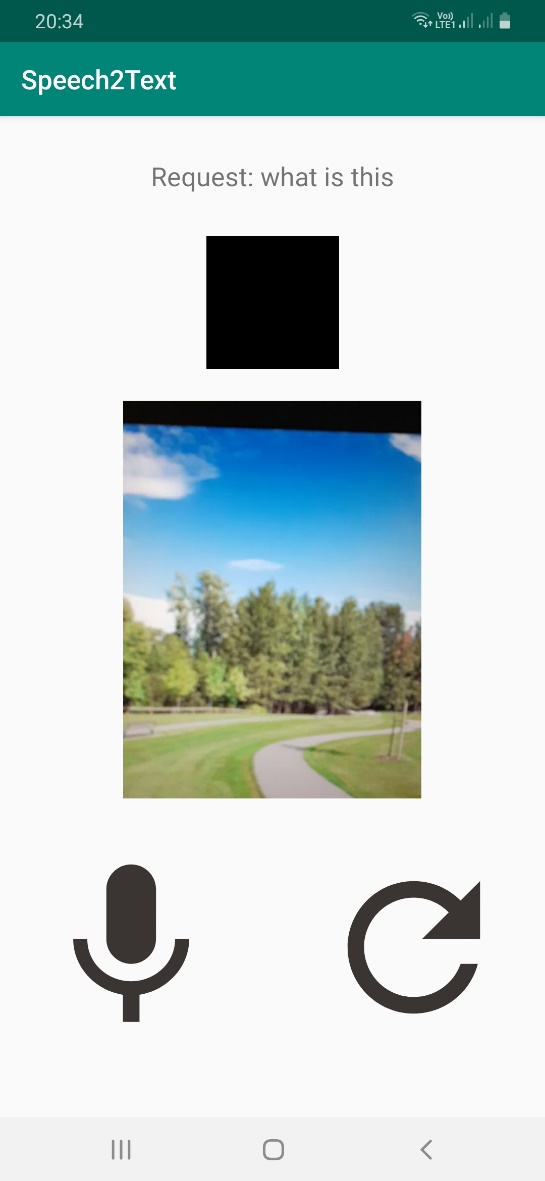


Figure 6.2: Application Screen

Figure 6.2 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 6.2 below.

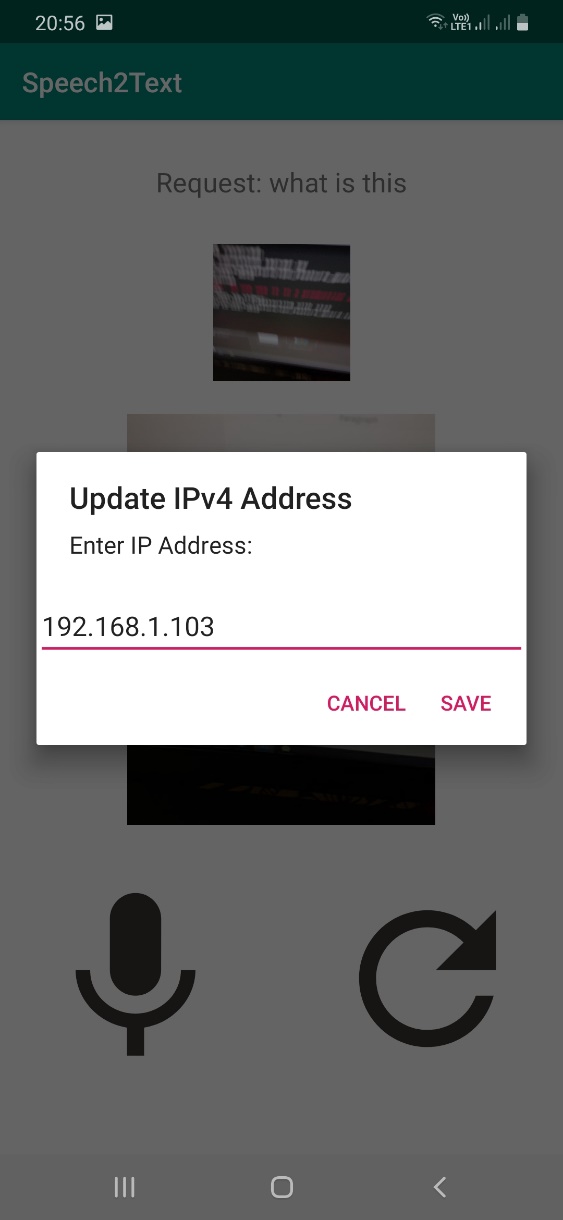
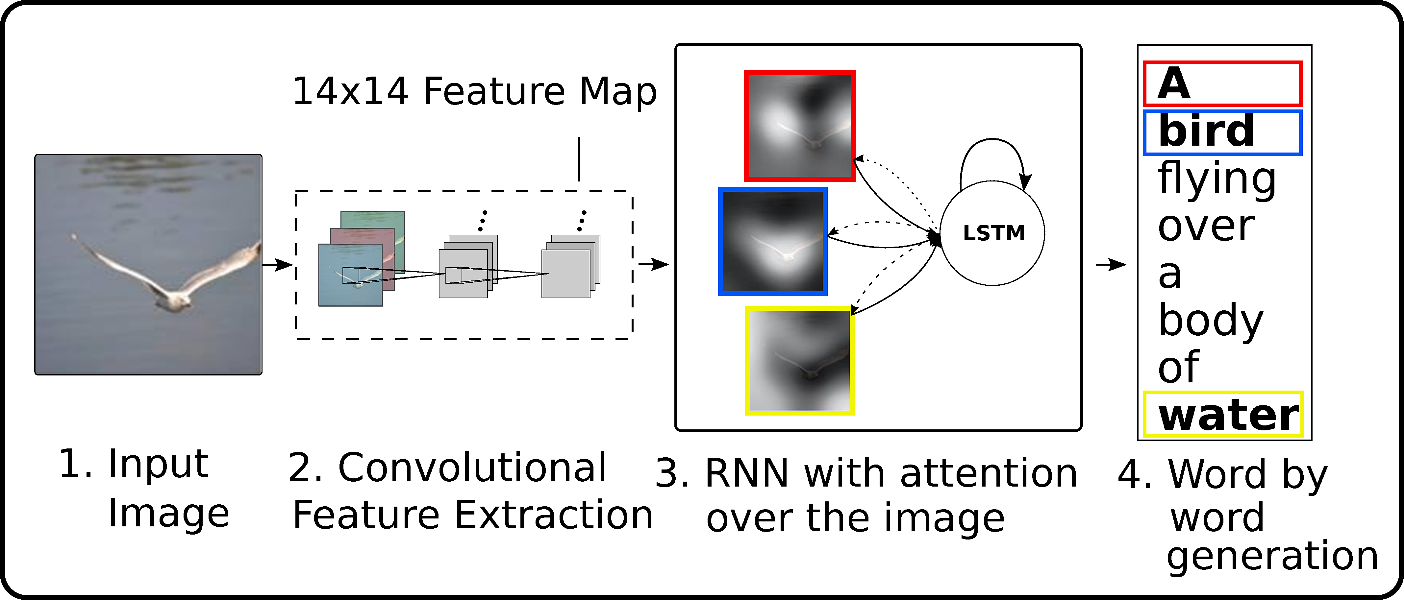


Figure 6.3: IP Address Connectivity

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 6.4 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 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 6.4: Image Captioning Architecture Diagram

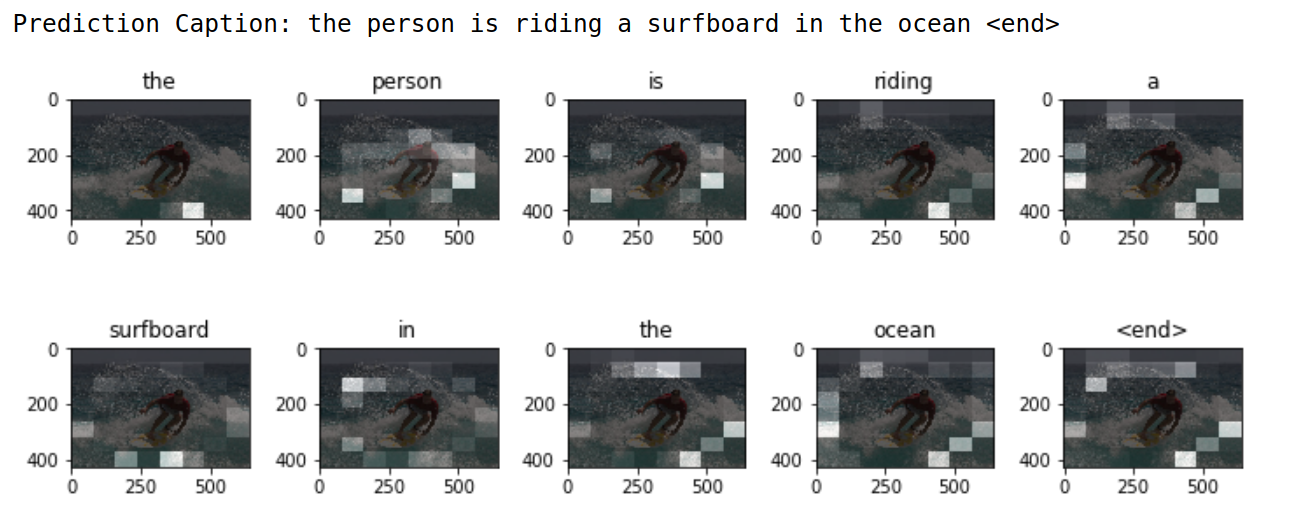
 Figure 6.5: Image Captioning using Attention

Figure 6.5 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 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 6.6 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.

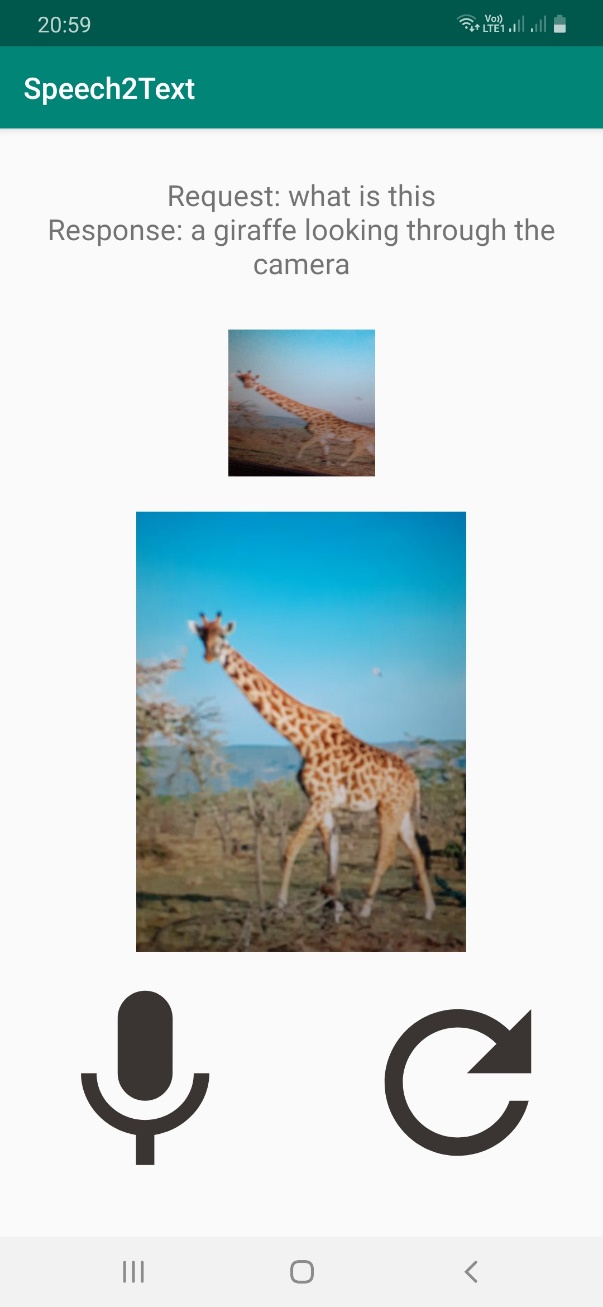


Figure 6.6: Image Captioning output in the application