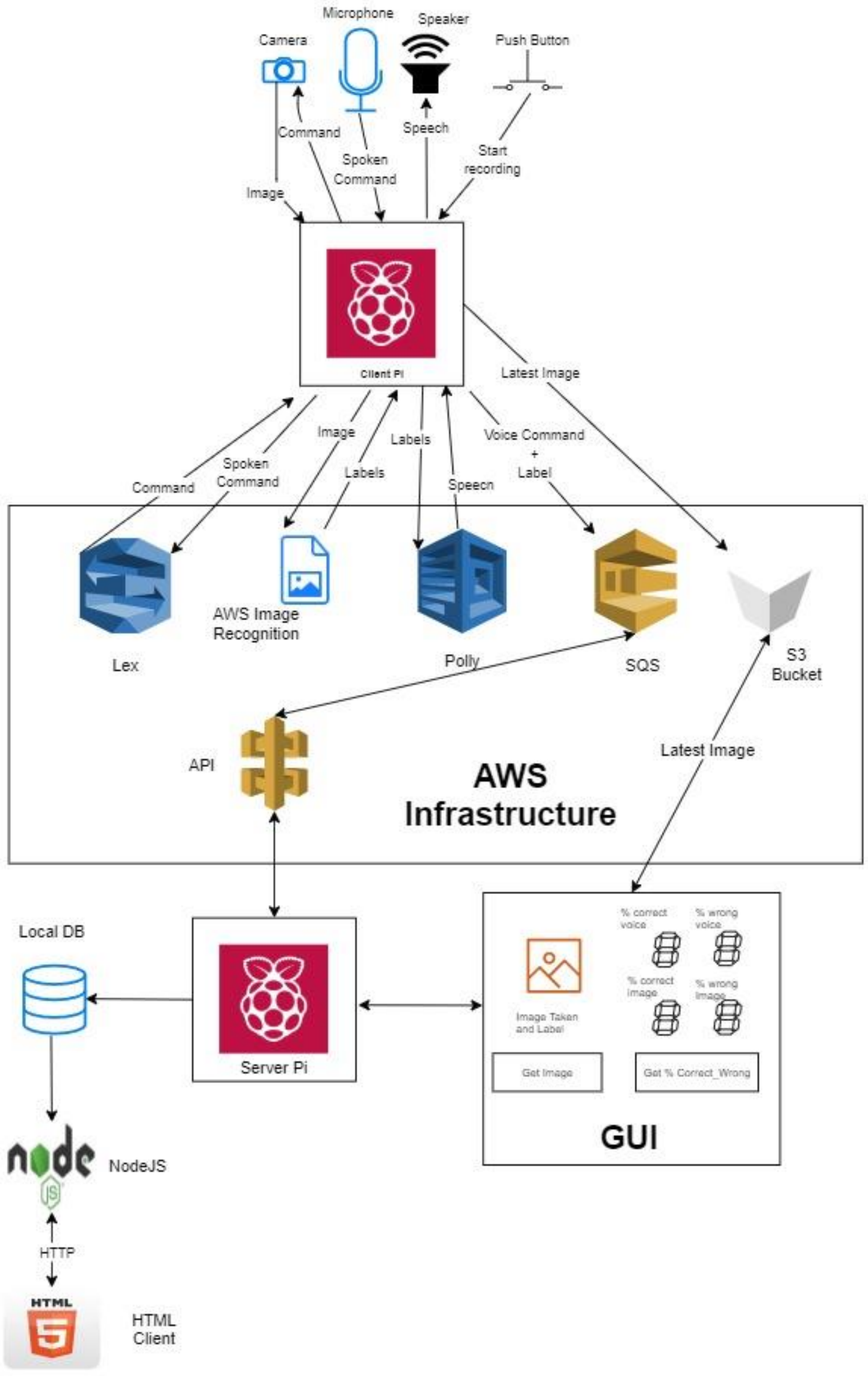
Magic Wand

Sanju Prakash Kannioth

Srinath Shanmuganadhan

12/11/2019

Architecture Diagram:



Working:

Client:

When the user presses the push button on the client pi side, the microphone is turned on and the user can speak in the voice command. This voice is then sent to AWS Lex through the client pi for speech to text conversion. Once the conversion is done, the text is returned to the client pi. If the voice command spoken by the user is “Identifio”, the camera on the client pi is turned on and an image is captured. This image is sent to AWS Rekognition for image recognition. Once the recognition is complete the labels are sent back to the client pi. After the labels have been received by the client pi the label with the highest confidence factor is chosen and is sent to Amazon Polly for text to speech conversion. Once this conversion is completed the speech file is sent to the client pi. This speech file is then played on the client pi using the speaker connected. Once the speech file has been played, the user can push the button again to indicate if the image recognition is correct using the commands “Correcto” or “Wrongo”. The label and voice command is then sent to the AWS SQS queue. The image captured is also sent to an AWS S3 bucket for storage and retrieval from the server.

Server:

The server consists of a GUI made using PyQt5 and a HTML client. The PyQt GUI is used to query the AWS SQS queue for data. The percentage of correct and wrong image label and voice command data are retrieved from this SQS queue. This data is then stored in the Local DB which is a MySQL database. There is also a NodeJS script that is running as a WebSocket server that can be used to query the database. The HTML client has a button to request for all the data in the local DB. This data is then displayed in a tabular format on the HTML Client. The communication between the HTML client and NodeJS server is over HTTP WebSockets.

Project Deviation Statement:

- Initially the plan was to send the AWS Rekognition labels directly to the SQS Queue. This has been changed by routing the image labels to the client pi and then the client pi sending the image to the SQS Queue.
- Initially the plan was to send the AWS Lex output directly to the AWS SQS queue. This has been changed by routing it to the client pi and the client pi sending it to the SQS queue.
- Storing the captured image for the server pi to retrieve it was not thought of in the last 2 updates. This storage is achieved by using AWS S3 bucket. The images stored in the bucket can be retrieved by the server by using AWS S3 API.
- The on and off of the microphone was also not planned out in the last 2 updates. A push button has been interfaced to signaling the ON of the microphone and the microphone will stay active for 3 seconds.

Third Party code used statement:

Most of the code used in this project has been taken from API Documentation examples. All other code is done using stand libraries.

The AWS Python codes were referenced from:
https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

Push button code: https://www.electronicshub.org/raspberry-pi-push-button-interface/

PyQt5 Image Display tutorial: https://www.youtube.com/watch?v=D0iCHFXHb_g

Microphone: http://jonamiki.com/2019/07/04/sound-on-raspberry-pi-separate-speaker-and-microphone/

Project Observation Statement:

Better than expected:

- Integration of all the AWS code.
- Usage of AWS with python was easier than using it with JavaScript.

Worse than expected:

- Designing the communication between each element in the system.
- NodeJS component was left until the end of the project and took time to complete.
- Setting up the microphone was harder than expected.

Differently than expected:

- Initially our idea was to directly send the labels and voice commands directly to the SQS queue. But we had to change this to work as explained above.