# THE MAGIC WAND

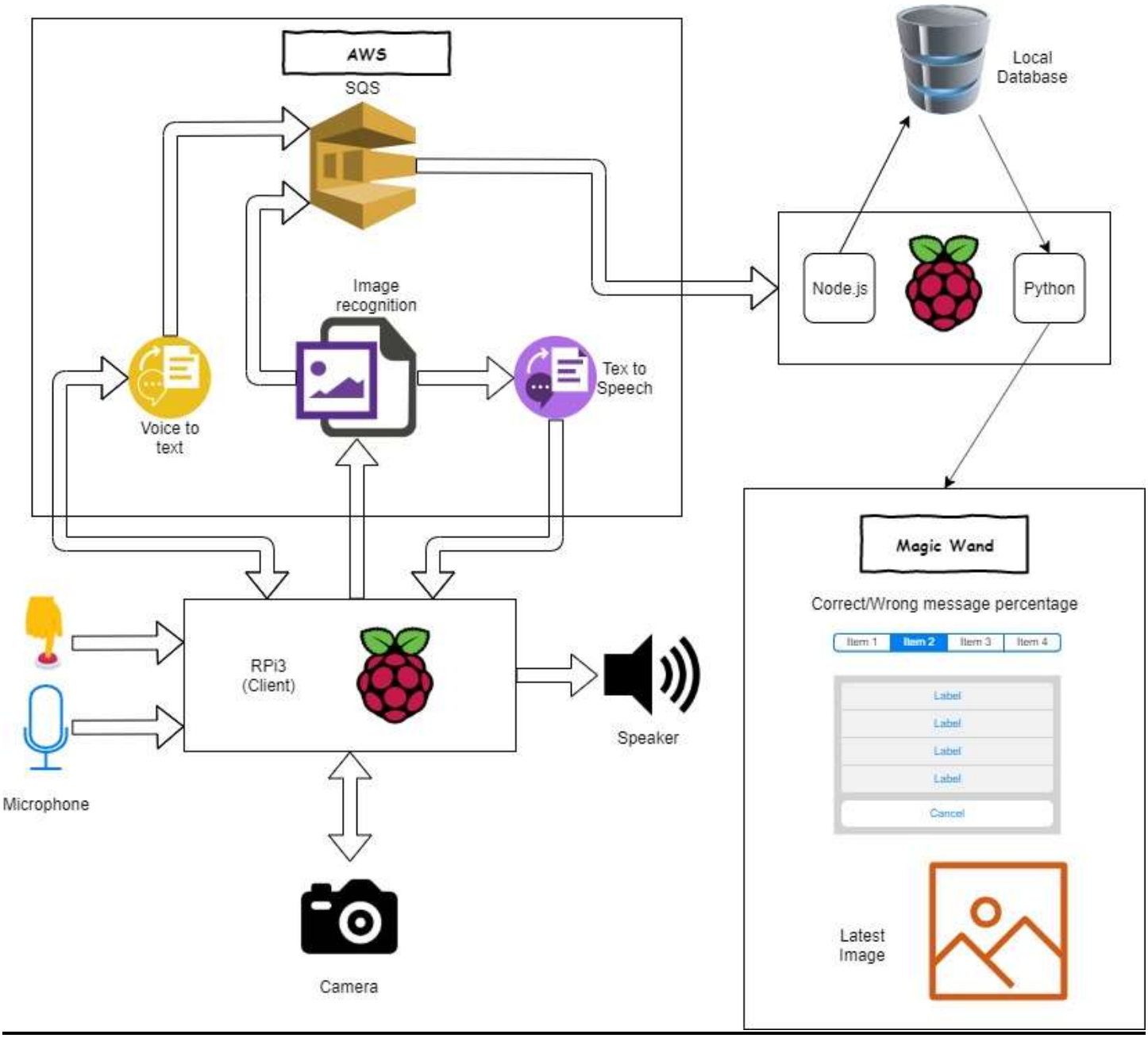
# SUPER PROJECT- EMBEDDED INTERFACE DESIGN

**Group Members**:

- Puneet Bansal
- Amogh Shrikhande

**Date: 12/11/2019**

## FINAL ARCHITECTURE DIAGRAM AND STATEMENT:



- The user pushes the button to start the functioning of the device. The user says “identify” which goes to Amazon Lex service. This service returns the text which then gets stored in my SQS queue.
- Then, the camera is enabled which clicks the picture and sends it to S3 Bucket. The image is sent to Image Recognition service through s3 bucket. The image label is identified and sent back to pi.
- The image label is sent to amazon poly service which converts text to speech and this speech is then played via speaker.
- The user then speaks whether the label played by the speaker is the correct recognition of the image or not.
- The command, response and label is sent to AWS SQS in 3 separate queues.
- On the server side, Node JS is used to take the data from SQS and store it in MySQL from where the data is fetched in the python app and displayed on the graphical user interface.

## PROJECT DEVIATION STATEMENT:

1) **GPIO Pushbutton integration:**
    a. We integrated a push button interfaced to pin 11 of the Raspberry Pi. The code on client-side Pi is supposed to run continuously in a loop in order to mimic “Amazon Alexa”. However, if we just include a while loop to keep on taking the inputs, we would read a lot of unwanted inputs. In order to trigger the microphone only at the point where we want to record “identify” we integrated the push button.
2) **Node JS Integration:**
    a. In the initial phases of project, we developed the project completely on python due to our familiarity and comfort with the language and syntax. After we got a proof of concept of our entire loop working, we then integrated Node JS into our system. Node JS in our project is used to fetch data from the Amazon SQS services and feed it to the local MySQL database. The python application then, fetches data from MY-SQL database.

## THIRD PARTY CODE USED STATEMENT:

- MySQL link: https://pimylifeup.com/raspberry-pi-mysql/
- Python MYSql link : https://www.w3schools.com/python/python_mysql_create_db.asp
- Python: https://www.geeksforgeeks.org/python-programming-language/
- NodeJs talking to mysql: https://www.w3schools.com/nodejs/nodejs_mysql.asp
- SQS AWS: https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/SQS.html#receiveMessage-property
- AWS login: https://aws.amazon.com/education/awseducate/
- Playing audio files in python: https://raspberrypi.stackexchange.com/questions/7088/playing-audio-files-with-python
- Configuring amazon lex using boto3: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/lex-runtime.html#LexRuntimeService.Client.post_content
- Setting up amazon lex bot: https://www.youtube.com/watch?v=Gy0C9g16DW0
- Setting up aws text to speech (polly): https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/polly.html#Polly.Client.syn thesize_speech
- Interfacing camera with Rpi3: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera
- Interfacing usb microphone with Rpi3: https://raspberrytips.com/add-microphone-raspberry-pi/

**PROJECT OBSERVATION STATEMENT:**

1) **Node JS continuous loop problem:**
    a. We are using node.js to fetch data values from SQS and feeding it to the local database. The problem that went worse than expected was to figure out how to poll the SQS to fetch latest values. Initially, we used an infinite while (1) loop for the same, but this caused the code to go in an unknown blocked state.
    More thoughts on the same, made us realize that we need to have some mechanism to trigger my-sql query mechanism every-time rather than running it continuously. So, we implemented a timer that expires every 4 second and triggers my-sql query mechanism.
2) **Comfort with AWS API's**
    a. Due to our persistent use of Amazon AWS services from the past few projects, we didn't face much problem integrating AWS API's with both NodeJS and Python. This was something we thought would take quite a lot of time, since when we planned the time-frame we were not that comfortable with AWS.
3) **Embedding Image with PyQt5**
    a. As per the project requirement we had to display the latest image captured, on the PyQt5 GUI. As we had not done this earlier, we thought that this would be time consuming. But this task was easily completed without any difficulty, thanks to the reference that we got