

Design and Architecture of a Public Satisfaction Detection Camera Based on Facial Emotional Analysis

Martin Clinton Tosima Manullang^{1*}, Ahmad Luky Ramdani¹ and Namuri Migotuwio²

¹ Teknik Informatika, Institut Teknologi Sumatera, Lampung Selatan, Indonesia

² Desain Komunikasi Visual, Institut Teknologi Sumatera, Lampung Selatan, Indonesia

* E-mail: martin.manullang@if.itera.ac.id

Abstract. One way to measure customer satisfaction is to involve the Internet of Things and Artificial Intelligence in the form of mood recognition services provided by Amazon Web Service and the services available to it, such as S3 (Simple Storage Service, a storage service from AWS), Relational Database Services, Rekognition services (An Image Recognition Services), Application Protocol Interface, Lambda Services. The source of input acquisition in this study uses the ESP-EYE device, the ESP32 microcontroller which has a camera. The device is connected to Amazon web service via the internet. Data is stored in a storage service called S3. S3 is used as a storage bucket, where Lambda will be the center of the process that pre-processes the face image data to crop the edge. After the cropping process, lambda will call the Rekognition service to start the Rekognition services process. The relationship of each data is stored in RDS using MariaDB. The results are, that the internet of Customer Satisfaction Systems can be built with the involvement by several products of Amazon Web Services. There is also considerable efficiency after the face image file has been processed by lambda to crop. The output from the system is recorded in the database and sent back to ESP-EYE for display in the serial monitor. The development of this system can be forwarded to an information system that allows each column in the database to be presented in the form of an information system. This system can help to measure customer satisfaction and recapitulate the results without involving the surveyor team so that it will save operational costs and time efficiency.

1. Introduction

Measurement of customer satisfaction is one of the tasks and assessment indicators in the success of a business. According to Peterson and Wilson, a business that lasts a long time is a business that satisfies customers. Thus, the primary purpose of public services, both profit-based and non-profit, is to satisfy customers [1]. That is why, large companies, especially those engaged in services, budgeted a large enough cost in measuring customer satisfaction. Not only that, dissatisfied customers must be treated properly to prevent the development of dissatisfaction.

In general, measuring customer satisfaction requires direct interaction with customers. Measurements can be made either through interviews or questionnaires. For some non-face-to-face interactions, for example, in the process of customer complaint services over the telephone, recording interactions between customer service officer and customers become the measurements. Then using several methods, the assessor makes observations to determine the level of customer satisfaction [2].



Measurement of customer satisfaction with conventional models triggers to several problems and errors. Interview and questionnaire methods require considerable human resources to conduct surveys. Also, other problems that must be prepared is survey support equipment such as questionnaire papers, stationery, until the time needed to ask customers to fill out or conduct interviews. In Addition, the data from interviews and questionnaires cannot produce results instantly. It takes a recapitulation of the mathematical calculations to process the data into outcome information that is useful and meaningful for the sustainability of the business/service [3].

In today's modern era, artificial intelligence has much involvement in the business world and public services. Artificial intelligence provides solutions in business decision making to stock management in warehouses. Through this research, artificial intelligence is involved in aspects of customer satisfaction assessment. There is some method to acquire and interpret emotional expression from humans, such as speech, eye movement, and human-computer interaction. Artificial intelligence is trained to be able to identify unique data obtained through the camera for further processing automatically to produce useful information output. Artificial intelligence, in its relation to face recognition, creates an analytical computer that can identify a person's face and process information in the face [4].

To create a centralized processing, business lines often use cloud computing systems that store and process data from a variety of sources, one of which can be hardware. Cloud computing and the internet of all (Internet of Things) allow data from each hardware, processed by the same processing centre, and stored in the same storage media. This provides practical solutions in database infrastructure and remote processing technology [5]. Many cloud computing solutions including Microsoft Azure, Amazon Web Service, Google Cloud, IBM Bluemix and more provide a multi-purpose image recognition service that provides mood and facial recognition [6]. Amazon Web Service is used as a cloud service in this study due to performance evaluation results and compatibility with the hardware used, ESP-EYE, which is based on ESP32-made ESPressif boards [7]. The design of this system can certainly be scaled in very large sizes. Assuming a system is applied in measuring customer satisfaction in supermarkets that have 30 cashiers, it takes 30 ESP cameras that can work and are synchronized with each other to obtain a similar database between all cameras.

2. System Overview

The overall system design involves at least three elements, namely software, hardware, and networking. All three are needed to be integrated with each other. This is what is referred to as system architecture. This section will discuss some system objects that are used in the development of this system.

2.1. ESP-EYE

ESP-EYE is a control device which is a combination of the ESP32 microcontroller and a camera as an input device. This tool allows image processing and sending images via Bluetooth and WiFi networks. ESP32 works well in encoding images and transmitting them through the internet network [8]. ESP-EYE is an ESP32-based developing module, equipped with an external 2-megapixel camera and a digital microphone, 8 MB of Pseudo Random Access Memory and 4 MB of flash. These things make the board preferred the option for facial recognition, face identification and natural language processing implementations. In addition, the device can also support Wi-Fi image transmitting and testing via a Micro Universal Serial Bus connector, which allows innovative Artificial Intelligence solutions to be created [9]. The figure of ESP32 can be seen from the images listed below.

2.2. Amazon Web Service

Amazon Web Service (In general, it is known and abbreviated as AWS) is one of the most commonly used cloud computing services by companies around the world [10]. Amazon Web Service has an excellent ability in terms of virtualization processing through computing devices and services [11]. Cloud computing is a paradigm for the quick deployment and launches with limited maintenance activity and interference between the service provider for the delivery of all-round, accessible and on-demand network access to a shared pool of computer resources, optimized (e.g. networks, databases,



Figure 1. ESP-EYE Board [9].

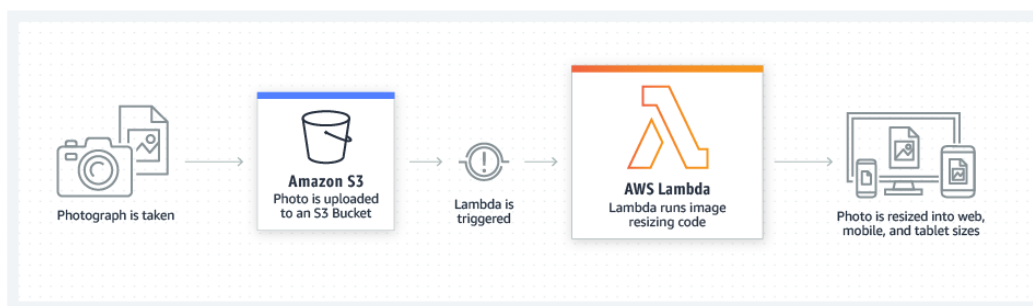


Figure 2. Implementation of Lambda Function in Resizing Photos [17]

storage systems, apps, and services) [12]. Some of the services from AWS (Amazon Web Service) used in developing this system will be explained in the next subsection.

2.2.1. Lambda

AWS Lambda is a computing service that allows software execution without servers. AWS Lambda will only run the software manually as necessary from a few requests every day to several thousand requests per second. User can run the software for nearly every program form or back with AWS Lambda. AWS Lambda is one of microservice things in cloud computing [13]. In general, the process of working lambda in converting images can be seen in Figure 2.

Microservice architectures offer a solution to effectively calculate computer resources and to address a large number of other challenges in monolithic architectures. Microservice architectures, however, pose other obstacles, such as making efforts to deploy and scale and run that microservice in Cloud infrastructures. Services such as AWS Lambda enable the deployment of microservice architectures without the need to handle databases to address this concern. It makes it easier for systems to be distributed and streamlined and helps to reduce maintenance and operational costs. The process, therefore, decreases operational costs [14].

2.2.2. S3 Storage

Amazon S3 is a web-based service provided by Amazon Web Service S3 (Simple Storage Service) is a storage service that allows data which has been acquired, stored for further processing or as an archive. Amazon S3 provides the products of AWS. S3 Cloud storage systems provide an easy and quick way to store and transfer items of various kinds. Nonetheless, the above software framework masks information of the system's deployment and output data, thus maintaining smooth work [15].

2.2.3. RDS Database

It is easy to create, run and scale relational databases into the cloud with Amazon's Relational Database Service (Amazon RDS). In automating lengthy administrative tasks, such as software

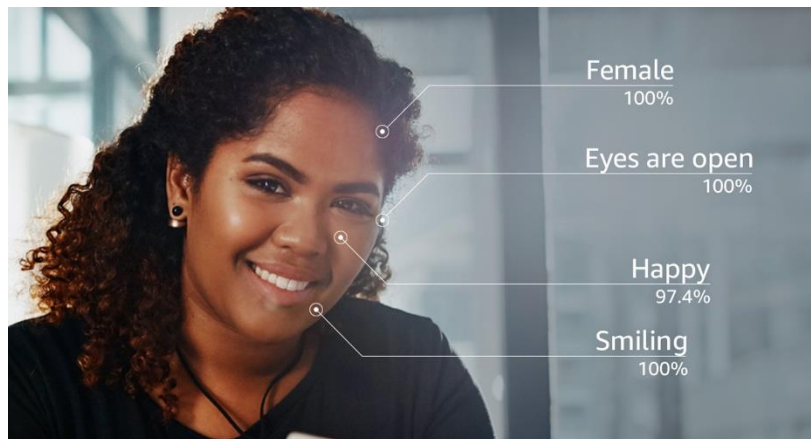


Figure 3. Facial Analysis Using Amazon Rekognition services [18].

delivery, database management, patcher, and backup. Amazon RDS offers cost-effective, resized power. Amazon RDS provides user with the ability to concentrate on requirements in order to ensure accelerated performance, responsibility, security and reliability. Several types of databases are supported by RDS such as Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server.

2.2.4. AWS Rekognition services

It is easier to include image and video analysis in an application by Amazon Rekognition services. The Rekognition services API also includes images or videos, and the software recognizes artifacts, people, language, graphics and behaviours and detects inappropriate information. In the images and videos, Amazon Rekognition services provides very detailed facial analysis as well as face Rekognition services. For a wide range of client authentication, counting and public security applications, user can identify, evaluate and contrast heads. Examples of face identification that can be done by recognition services on AWS can be seen in Figure 3.

Amazon Rekognition services is built on the validated, widely scalable computing software that computer vision scientists create to evaluate billions of images and videos every day and does not involve the use of professionals in machine learning [6]. AWS Rekognition services can provide xml output responses that are easily used for further development.

2.3. API

API stands for Application Programming Interface which allows developers to seamlessly combine two pieces of a device or with different applications. The API consists of different elements, such as functions, protocols, and other resources that allow developers to create apps. The aim of using the API is to speed up the development phase by independently offering functionality so developers don't need to create similar apps. The application of the API will be intensely felt if the desired features are very complex; of course, it takes time to create something similar to it. For example integration with payment gateway. There are various types of API systems that can be used, including operating systems, libraries, and the web [16].

2.4. ESP-WHO

Detection, Recognition and Photo Usage are at the core of the framework in ESP-WHO.

- Image Utility offers APIs that process fundamental images.
- Identification uses photos as data and offers the facial location when the object is in. It is implemented with MTMN (Multi-Technology Network Management) model referring to both MTCNN (Multi-task Cascaded Convolutional Networks) and MobileNets.

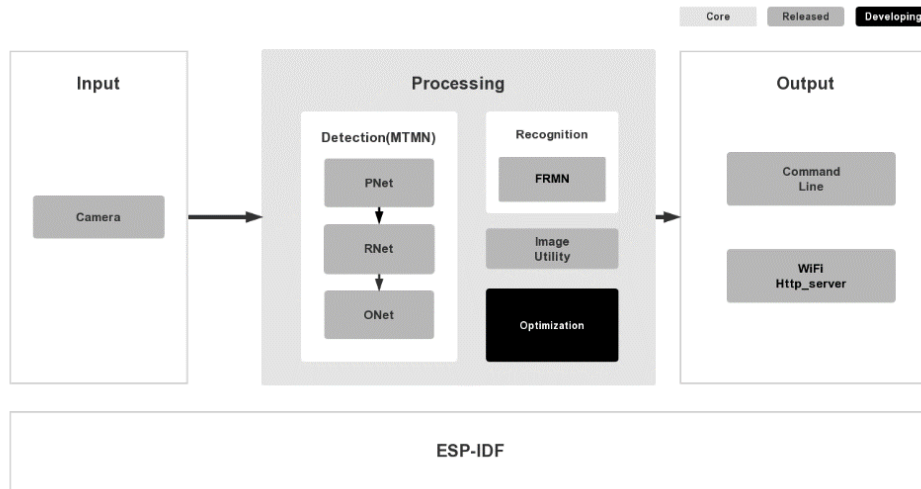


Figure 4. ESP-WHO Schematic Diagram [19]

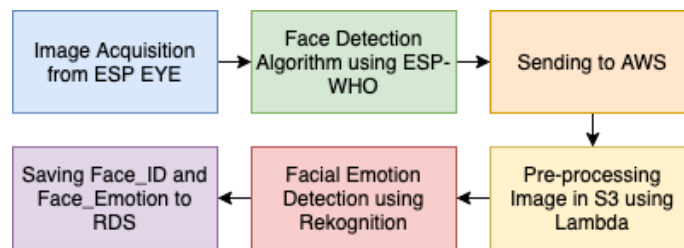


Figure 5. System Block Diagram

- Identification is to recognise the individual person, and the detection results are required. It is introduced with the software MobileFace.
- Optimisation is mainly intended to increase decision consistency and speed up the entire cycle. But it could also alter the network configuration, upgrade the coefficients, refactor the application etc.

Overall, the WHO ESP framework can be seen in Figure 4.

3. System Design

Of the several components that have been outlined, the design is explained in this section. In general, the system works as the block diagram in the figure 5.

Face images are obtained from a camera connected to the ESP32 control board. By using a face recognition algorithm (ESP-WHO), the camera will capture an image if a face is detected. For reasons of efficiency in internet and memory resources, cameras and control boards do not transmit continuous images (video) continuously to the cloud computing system. ESP-WHO is implemented as a code into ESP-EYE using the Arduino Integrated Development Environment. All kinds of ESP microcontroller variants allow it to be programmed using Python and Arduino Integrated Development Environment. Some configurations, such as Service set identifier and password from the wireless fidelity access point are also embedded in the form of code that is uploaded to the ESP board.

After the image is sent to AWS via a wireless fidelity internet network with the API, the image will be saved in S3. Lambda pre-process according to the script that has been set on Lambda. Pre-

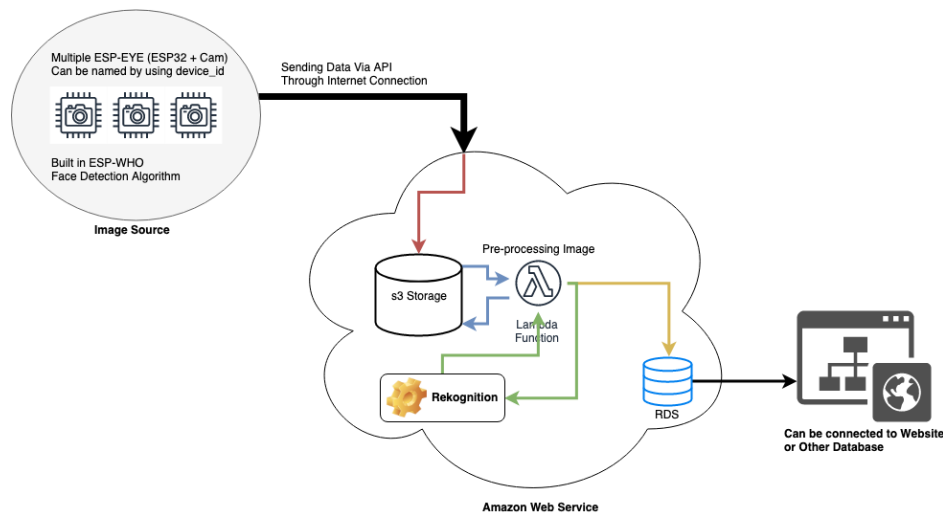


Figure 6. Schematic Diagram.

processing the image is performed by cropping the face to save storage media, computing time, and of course the cost of cloud services. The processing can be seen in the following line of code:

```
for face in faces['FaceDetails']:
    left = CurrImageWidth * face['BoundingBox']['Left']
    width = CurrImageWidth * face['BoundingBox']['Width']
    top = CurrImageHeight * face['BoundingBox']['Top']
    height = CurrImageHeight * face['BoundingBox']['Height']
    right = left+width
    bottom = top+height
    cface = CurrImage.crop( (left, top, right, bottom))
```

The next step of lambda processing is checking whether the detected face has been recorded in the system. If so, the system will use the same face_id.

```
faceBytes = io.BytesIO()
cface.save(faceBytes, format=CurrImage.format)

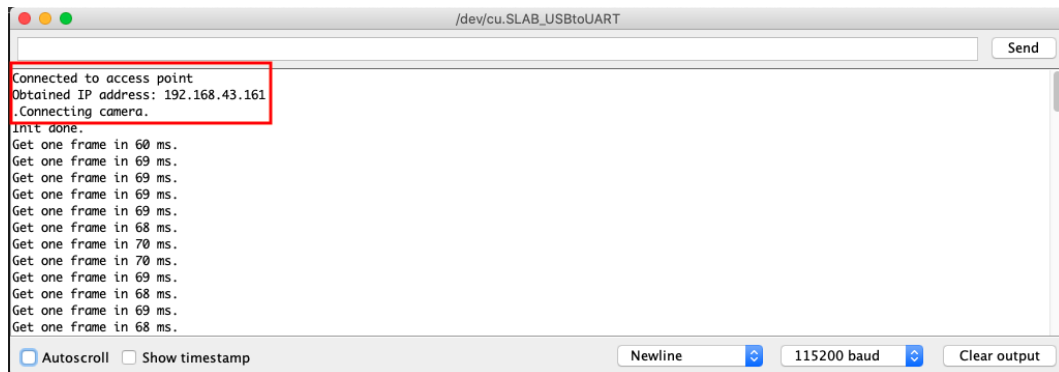
faceExists = Rekognition services
.search_faces_by_image(CollectionId=FACES_COLLECTION,
                      Image={'Bytes': faceBytes.getvalue()},
                      FaceMatchThreshold=MATCH_THRESHOLD,
                      MaxFaces=FACES_MAX)

if len(faceExists['FaceMatches']) == 0 :
    dto = datetime.now();
    eid = dto.strftime("%y%m%d-%H%M%S-%f")
    file_name = BucketFolder + eid + ".jpg"

saved = s3.put_object(Bucket=BUCKET_NAME, Key=file_name, Body=faceBytes.getvalue())

if saved['ResponseMetadata']['HTTPStatusCode'] == 200:
    record = Rekognition services .index_faces(CollectionId=FACES_COLLECTION,
    Image={'S3Object':{'Bucket':BUCKET_NAME, 'Name':file_name}},
    ExternalImageId=eid,
    MaxFaces=5,
    QualityFilter="AUTO",
    DetectionAttributes=['ALL'])

faceId = record['FaceRecords'][0]['Face']['FaceId']
```



```

/dev/cu.SLAB_USBtoUART
Send
Connected to access point
Obtained IP address: 192.168.43.161
Connecting camera.
Init done.
Get one frame in 60 ms.
Get one frame in 69 ms.
Get one frame in 69 ms.
Get one frame in 69 ms.
Get one frame in 69 ms.
Get one frame in 68 ms.
Get one frame in 70 ms.
Get one frame in 70 ms.
Get one frame in 69 ms.
Get one frame in 68 ms.
Get one frame in 69 ms.
Get one frame in 68 ms.
Autoscroll Show timestamp Newline 115200 baud Clear output

```

Figure 7. Serial Monitor Status - ESPEYE connected to access point.



```

/dev/cu.SLAB_USBtoUART
Get one frame in 45 ms.
Get one frame in 47 ms.
Get one frame in 46 ms.
Get one frame in 67 ms.
Get one frame in 42 ms.
Get one frame in 43 ms.
Face Detected. Uploading to AWS, please wait...
{"status": "success", "faces": [{"FaceId": "6b25db69-f87e-4fe9-9467-3b8cfa304a16", "mood": "CALM"}]}
Get one frame in 35 ms.
Get one frame in 65 ms.
Get one frame in 68 ms.
Get one frame in 68 ms.
Get one frame in 67 ms.
Get one frame in 69 ms.
Get one frame in 69 ms.
Get one frame in 68 ms.
Autoscroll Show timestamp Newline

```

Figure 8. Serial Monitor showing Face Detection status and Feedback from System.

Face image that has been processed is then ready to be detected using Rekognition services. Rekognition services then is called to measure the mood and sentiment of the face. The results of the analysis will be recorded on the RDS and also sent back to ESP-EYE, which can be displayed via a serial monitor. A complete schematic diagram of this system can be seen on Figure 6.

4. Results and Discussion

From the above design, the system is tested to be able to record the face_id and emotion obtained and store both in a database. In testing, the database is emptied, and all entries in the storage bucket are emptied. ESP EYE is connected to an access point (Wireless Tethering) and successfully connected to the internet. Screenshot of the serial monitor can be seen in Figure 7

After that, ESP-EYE is directed to the face to be tested. Within 3 seconds (depending on network conditions), server feedback is received explaining face_id and mood analysis. Each frame takes about less than 100miliseconds to be captured and sent through the internet by using POST API. Detection status can be obtained from figure below 8.

To see the processes that occur behind the server, observations are made on the cloud side. The first step is to open the bucket on S3 to see the face image that has been processed by lambda (cropping edges). In Figure 9 you can see the display of the s3 bucket that can be opened via the console on the Amazon web service. To show the results of Lambda Services processing, the data is downloaded and displayed as Figure 10.

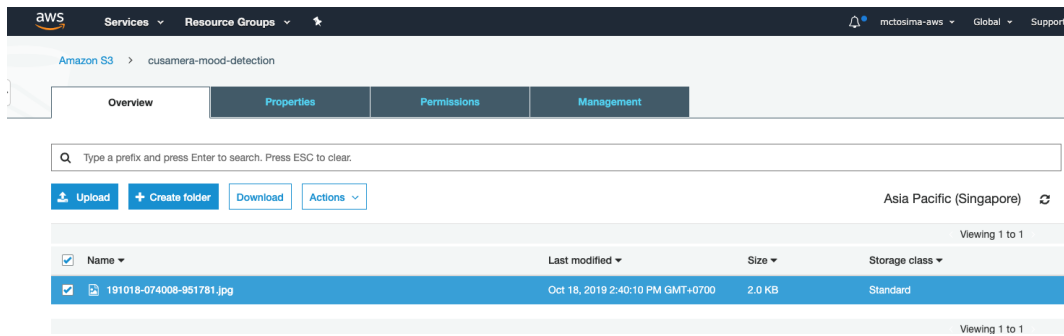


Figure 9. There is an entry on the S3 bucket that contains photos.



Figure 10. A Cropped Face Image.

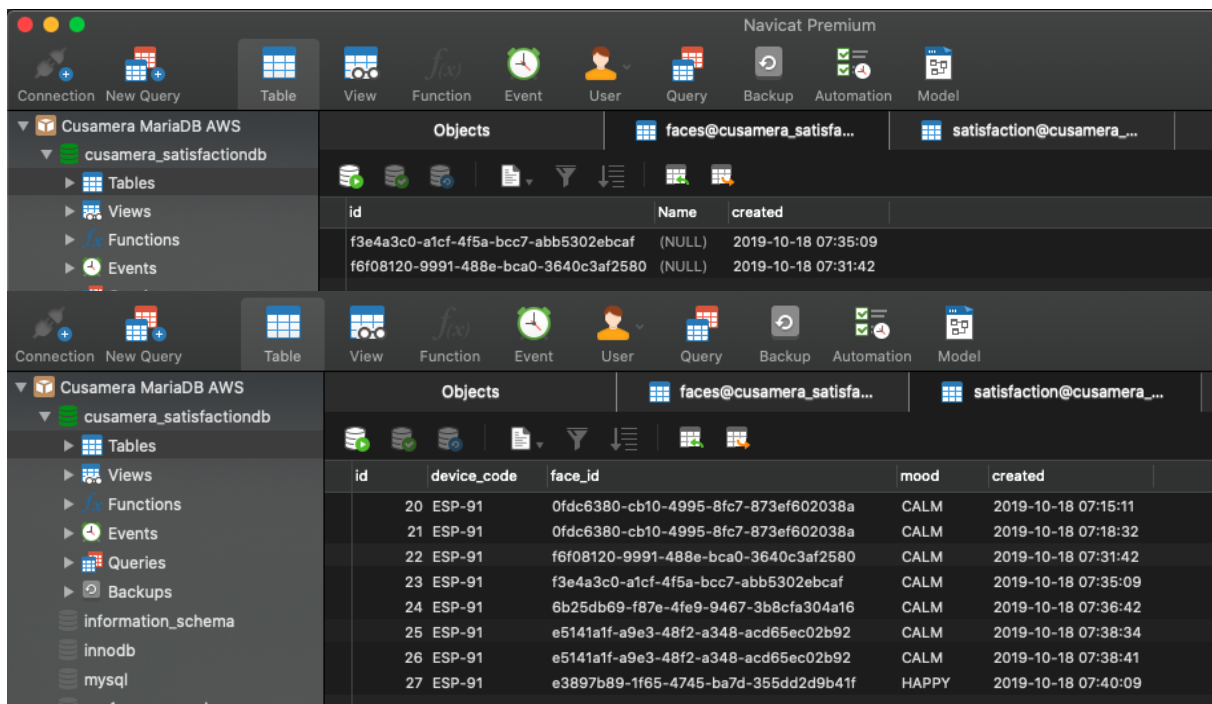


Figure 11. Faces Table and Satisfaction Table in RDS Database.

As for the display on the database, Navicat Premium is used to inspect the database so that it is not bothered with the Syntax Query Language command_line view that is normally accessed using the terminal/command prompt. Figure 11 shows Navicat Premium windows that show the database.

Faces table contains a collection of data relations between recorded faces. In its implementation, the Real_Name column can be added as an interpretation of the face so that users can utilize this system for more convenient purposes, such as customer Rekognition services, which allows bank

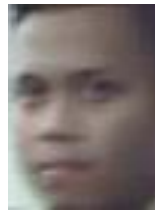


Figure 12. A Processed Image from S3 Buckets.



Figure 13. Another Serial Monitor Status from Different Test Object.



Figure 14. Test Object That Replicate a Happy Face.

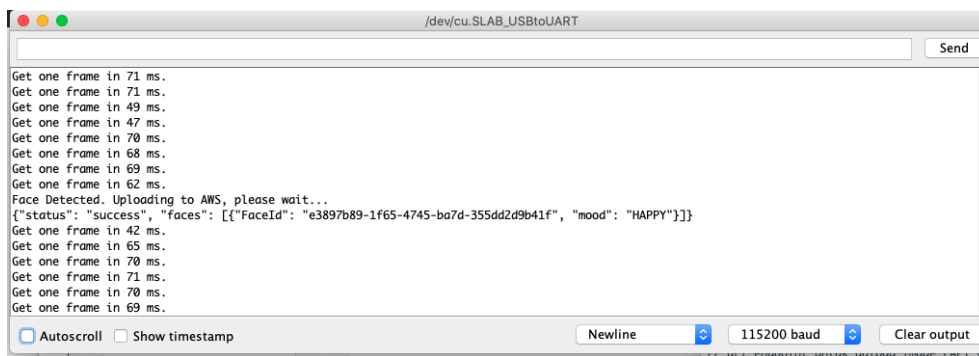


Figure 15. A Result of Happy Face Detection.

tellers/tellers to know the customer's name and data even before the customer introduces himself. There is also a Satisfaction table, which contains the history of detection that occurred, a mood column that is the result of facial emotion recognition. There is also a timestamp as a detection time indicator. The same person, is detected and recorded with the same face_id. The system is also tested on other people, who are expected to issue different face_id such as Figure 13. The compressed picture can be seen on Figure 12.

Can be seen in the picture, that when the system is confronted with different people, then the system will bring up different face_id. So everyone will have their face_id in database recording. The

system was also tested against different face mood. A test object tries to replicate the mood of a happy person to produce such results in Figure 14 and 15.

5. Conclusion

Internet of Things which is very closely related to processing in the cloud can be implemented to develop customer satisfaction measurement systems whereby involving several services on Amazon Web Service such as Lambda, S3, RDS, and Rekognition services. In terms of response speed, the system is able to work in real-time and produces latencies below 100 milliseconds which allows uninterrupted waiting time from the capturing image process to the results obtained. The process of cropping edges produces s3 storage efficiency so that costs are more efficient and the computing process becomes faster. This design can be implemented in many fields, from retail businesses to visitor registration systems. Development can be continued by considering input in the form of sound, and further classification in terms of gender, age, and other biometric aspects.

Acknowledgement

This research and project was funded by the 2019 Institut Teknologi Sumatera (ITERA) SMART Research Grant and authors would like to thank to Institut Teknologi Sumatera for providing the research grant (No.B/320/IT9.C1/PT.01.03/2019) through "Hibah Penelitian ITERA SMART 2019"

References

- [1] Peterson R A and Wilson W R 1992 Measuring customer satisfaction: Fact and artifact *J. Acad. Mark. Sci.* **20** 61–71
- [2] Ngo V M 2015 Measuring Customer Satisfaction: a Literature Review *Proc. 7th Int. Sci. Conf. Financ. Perform. Firms Sci.* 1637
- [3] Pizam A, Shapoval V and Ellis T 2016 Customer satisfaction and its measurement in hospitality enterprises: a revisit and update *Int. J. Contemp. Hosp. Manag.* **28** 2–35
- [4] Seng K P and Ang L M 2018 Video analytics for customer emotion and satisfaction at contact centers *IEEE Trans. Human-Machine Syst.* **48** 266–78
- [5] Gubbi J, Buyya R, Marusic S and Palaniswami M 2013 Internet of Things (IoT): A vision, architectural elements, and future directions *Futur. Gener. Comput. Syst.* **29** 1645–60
- [6] Al-Omar O M and Huang S 2018 A comparative study on detection accuracy of cloud-based emotion recognition services *ACM Int. Conf. Proceeding Ser.* 142–8
- [7] Sadooghi I, Hernandez Martin J, Li T, Brandstatter K, Maheshwari K, Pitta De Lacerda Ruivo T P, Garzoglio G, Timm S, Zhao Y and Raicu I 2017 Understanding the Performance and Potential of Cloud Computing for Scientific Applications *IEEE Trans. Cloud Comput.* **5** 358–71
- [8] Rai P and Rehman M 2019 ESP32 based smart surveillance system *2019 2nd Int. Conf. Comput. Math. Eng. Technol. iCoMET 2019* 1–3
- [9] Espressif ESP-EYE Getting Started Guide
- [10] Höfer C N and Karagiannis G 2011 Cloud computing services: Taxonomy and comparison *J. Internet Serv. Appl.* **2** 81–94
- [11] Jackson K R, Ramakrishnan L, Muriki K, Canon S, Cholia S, Shalf J, Wasserman H J and Wright N J 2010 Performance analysis of high performance computing applications on the Amazon Web Services cloud *Proc. - 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010* 159–68
- [12] Armbrust M, Fox A, Griffith R, Joseph A D, Katz R H, Konwinski A, Lee G, Patterson D A, Rabkin A, Stoica I and Zaharia M 2009 Above the Clouds: A Berkeley View of Cloud Computing *EECS Dep. Univ. California, Berkeley* 1–25
- [13] Malawski M, Gajek A, Zima A, Balis B and Figliola K 2017 Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions *Futur. Gener. Comput. Syst.*
- [14] Villamizar M, Garces O, Ochoa L, Castro H, Salamanca L, Verano M, Casallas R, Gil S, Valencia C, Zambrano A and Lang M 2016 Infrastructure Cost Comparison of Running Web

- Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures
Proc. - 2016 16th IEEE/ACM Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2016 179–82
- [15] Persico V, Montieri A and Pescape A 2016 On the Network Performance of Amazon S3 Cloud-Storage Service *Proc. - 2016 5th IEEE Int. Conf. Cloud Networking, CloudNet 2016* 113–8
- [16] Perez J L and Carrera D 2015 Performance characterization of the servioticy API: An IoT-as-a-service data management platform *Proc. - 2015 IEEE 1st Int. Conf. Big Data Comput. Serv. Appl. BigDataService 2015* 62–71
- [17] Amazon Web Service AWS Lambda
- [18] Amazon Web Service Amazon Rekognition
- [19] Espressif ESP-WHO

Reproduced with permission of copyright owner. Further reproduction prohibited without permission.