Voice Controlled Smart Mirror with Multifactor Authentication

Adokiye Charles Njaka, Na Li, and Lin Li
Department of Computer Science
Prairie View A&M University
Prairie View, Texas 77446, USA
Email: anjaka@student.pvamu.edu, {nali, lilin}@pvamu.edu

***Abstract*—Internet of Things, Smart Home, and Smart Cities have become hot topics in recent years. The development of these areas grows spirally due to the emergence of advanced smart devices. Smart mirrors are a new addition to the smart home family that has been getting a lot of attention nowadays by both commercial manufacturers and academia. This paper describes how a Raspberry Pi device can be used to enhance such mirrors with intelligence and security. The goal is to develop a cost effective intelligent mirror system that not only works as a regular mirror, but also be able to display various kinds of information, such as weather, time and location, current events, and users. The mirror can provide multimedia services while ensuring high end security across the entire system.**

***Index Terms*—smart mirror, Raspberry Pi, artificial intelligence, voice recognition, face recognition, Internet of Things, Alexa voice service**

I. Introduction

Smart Home has become an emerging topic in recent years in both academia and industry. Except for the well-developed commercial Internet of Things (IoT) products, such as smart door locks, smart plugs, and smart thermostats, smart home has attracted a massive number of people to work on self-motivated projects using low-cost hardware, such as Raspberry Pi [1]. The known Raspberry Pi based smart home projects cover a variety of topics, from Raspberry Pi powered security cameras to automated pool controllers [2] [3]. Due to its low cost, such projects are very popular in education, where students can get engaged in hands-on projects.

In this paper, the authors developed a smart mirror system. As a new addition to the smart home family, smart mirrors gained increasing attention recently. There have been some interesting projects published on the Internet [4]. For instance, Philips HomeLab [5] created an intelligent personal care system that implemented an Interactive Mirror [6] to provide customized services to users. The system is capable of displaying TV feeds, monitor the latest weather, and so on. The mirror has an LCD display combined with a mirrored surface and a processor to provide the intended services. Another project developed by Sam Ewen and Alpay Kasal at Lit Studios [7] is a touch and gesture functional mirror. The main method of interacting with the mirror is by touching the screen. Five students at Chalmers University in Sweden designed the HUD Mirror [8]. They used a two-way mirror to allow the LEDs mounted behind to illuminate the information. The system can display information such as time, weather, temperature, and a toothbrush timer.

The smart mirror described in this paper differs from the aforementioned projects mainly in that it offers unique features to improve the system security through biometric authentication, a voice assistant with added Alexa skills and multimedia capabilities. The biometric authentication is implemented with multi-factors, facial and voice, to ensure a higher security level than that provided by only voice which has been developed in Amazon Alex and Google Home. Plus, re-authentication is enforced if the user has not been interacting with the mirror for a while. Therefore, it can defend the system against several traditional security threats, such as session hijacking or man-in-middle attack. Additionally, the voice assistant and the multimedia display implemented in the mirror make users enjoy the intelligence of the smart home.

The roadmap of this paper is given as follows: Section II describes the overview of the system. Section III discusses the details of the system design and implementation. Section IV presents the experimental study, and a conclusion is given in Section V.

II. System Overview

The smart mirror system is designed to be a user-friendly solution that offers simplified and personalized services in the form of a mirror. Hence, the system was structured to give the user a complete control over all services and functions through voice interaction. Figure 1 depicts a schematic overview of the system, which can be divided into three distinct parts:

- Physical structure
- Smart mirror software
- Cloud services

The *physical structure* of the system consists of a one-way acrylic mirror placed over a display monitor, a Raspberry Pi device, and a microphone. The setup allows the reflection of light rays from one surface of the mirror while allowing light from the display to pass through. So a user can see his refection in addition to the contents shown on the display. The setup is finally enclosed with a wooden frame to give the natural mirror feel to the system.

978-1-5386-5959-5/18/$31.00 2018 IEEE

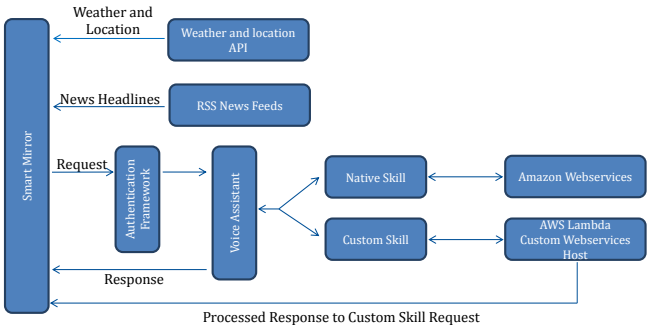

Fig. 1. System overview

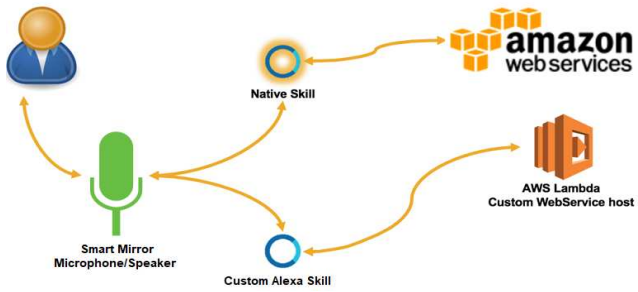

Fig. 2. System voice command architecture

The *smart mirror software* was developed using Python programming language. It consists of several modules which handle individual functions of the system, including weather information, current city, time and date information, news feeds and multimedia display. The mirror displays white characters over a black background for increased visibility from the other side of the mirror.

The system is supported by two *cloud services*. Amazon voice service (AVS - Alexa), Amazon web services (AWS), and Google custom search engine. As shown in Figure 2, Amazon voice service is used to provide the system's voice assistant capabilities in addition to the added Alexa skills. Amazon web services is used to create a queuing system and lambda functions to process special vocal instructions from the voice assistant. Finally, Google custom search is used in creating a customized search engine to provide multimedia content based on voice requests by the user.

III. System Design and Implementation

This section details how the system is designed and developed. First, two key components of the system, authentication framework and voice-based command, are presented. Then the entire smart mirror system is described. Figure 3 shows the design of the system.

A. Authentication Framework

The authentication framework of the system is designed to enable multifactor biometric authentication, namely, *facial recognition* and *voice recognition*.

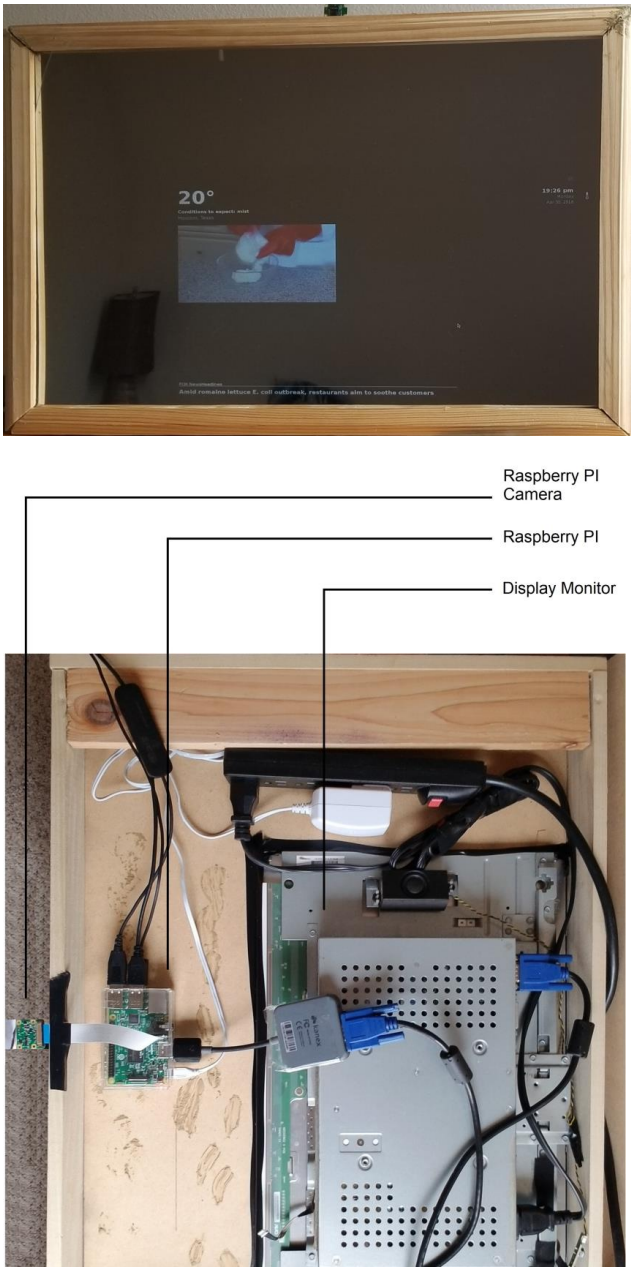

Fig. 3. Proposed system design

1) Facial Recognition: Facial recognition technology adopts an image matching approach to identify individuals. It can identify people from video sources or digital images. While there are several facial recognition algorithms, they generally work by comparing selected facial features from the given images with faces within a database. The technology has led to applications in consumer and enterprise use cases. This project implemented facial recognition using OpenCV library [9]. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning library. It supports face detection, face training, and face prediction. Face detection is the process of identifying faces

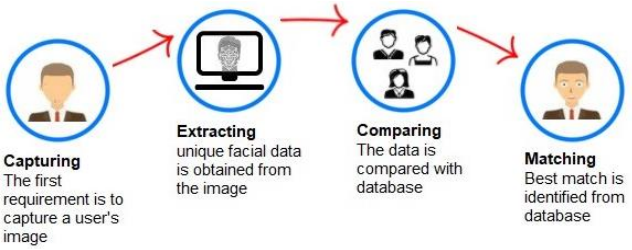

Fig. 4. Face recognition and prediction

in a picture. Face training is the process of studying several pictures of the same person to learn the features specific to the individual. This process creates a database of the known users. Face prediction is the process of identifying a known or pre-registered individual in a new image. With the support of OpenCV, two modules, namely *user registration* and *user authentication*, were developed in the authentication framework. Figure 4 shows the general procedure of face recognition and registration.

- User Registration - face detection and training

This module was implemented by taking images with a Raspberry camera. The images are passed into an OpenCV cascade classifier to identify human faces. A cascading classifier is a model that is pre-trained with several hundred face images, called positive examples, and negative examples - arbitrary images of the same size. A sample code snippet is shown as follows:

```
//Import opencv Library
import cv2
//Loading required cascading classifiers
face = cv2.CascadeClassifier('frontalface.xml')
//Load image and convert to grayscale mode
img = cv2.imread('image.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
//Face detection in converted image
faces = face.detectMultiScale(gray, 1.3, 5)
//Save image containing faces to disc
cv2.imwrite('image'+ str(Id))
```

If a face is identified in an image frame, the frame is saved. This step is repeated until 20 pictures of the user are saved and entitled with a unique integer and the image count (ranging 1 - 20). A unique integer is generated for each user in the system. Then the training process starts. All the saved images are loaded into an array, named faces, and passed into a recognizer. The recognizer is created as an instance of the OpenCV Local Binary Patterns Histograms class. Local Binary Patterns (LBP) describes a method for extracting useful features from face images. Basically, this method summarizes the local structure in an image by comparing each pixel with its neighbors.

```
//Create a LBH face recognizer object
recognizer = cv2.createLBPHFaceRecognizer()
//Train images and extract face features
//Each user has a unique integer Id
recognizer.train(faces, np.array(Id)))
//Save extracted features to face database
recognizer.save('model.yml')
```

As shown in the above snippet, this operation extracts the unique face features from all the pictures and saves them to a model file, which is known as the face database.

- User Authentication - detection and prediction

When a registered user wants to log into the system, he needs to be authenticated. The authentication procedure includes both face detection and face prediction. The authentication mechanism uses the Raspberry camera to obtain live images of the user. Like the previous step, a recognizer object is created. The live images are passed to the recognizer and the prediction method is called. This method compares the image with the model file created earlier and returns the unique id of the user (from the model file) whose facial features are closest to the current image. The prediction model also returns an integer value called the confidence which represents how close the image is to the identified user. In this paper, *a value of zero represents a perfect match.*

2) Voice Recognition: Voice recognition is a method to identify a person based on his voice characteristics (voice biometrics). Voice biometrics holds a significant advantage over other biometrics as it can recover from a data leak. If facial data were stolen, the biometric security would be damaged as we cannot easily change our face images. However, voice biometrics employs text-dependent characteristics and permits the creation of new voiceprints. Furthermore, in specific situations where fingerprints or facial scans cannot be used, such as where there is temporary damage to fingerprints or where there is limited lighting, voice biometrics possess will add advantages. One major concern plaguing the adoption of voice authentication technology is the possibility of an attacker recording a users voice or the possibility of voiceprint data being compromised [10]. The voice recognition API used in the project, is automatically configured to identify and reject all authentication attempts that are too similar to one or more previous voiceprints. This is based on the assertion that no two voiceprints can ever be exactly the same. Another main advantage of voice biometric is that the Equal Error Rates are much lower than finger, iris, and face, as shown in Figure 5. Equal error rate refers to an algorithm that describes the point at which the false rejection rate of a biometric system equals the false acceptance rate. A low Equal Error Rate indicates a more accurate biometric system.

The voice recognition function was implemented using a third-party API of VoiceIt [11]. The VoiceIt system uses three basic steps for the recognition of a speaker: capturing, digitalizing, and processing the sound waves using a complex mathematical method. The voice recognition consists of two modules: *user enrollment* and *user authentication*. Along with facial recognition, it forms a two-factor authentication.

- User Enrollment - Voice Registration

The first step of the enrollment process is user creation. The username of the current user is obtained from the database and passed to the "createUser" method of the VoiceIt API to create the user on the VoiceIt platform. To enroll a user, the system turns on the microphone and prompts the user to

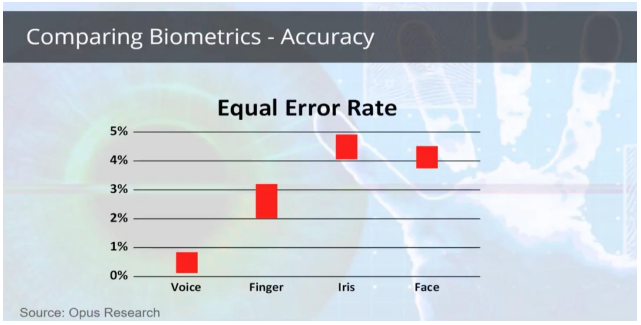

Fig. 5. Equal error rate chart [11]

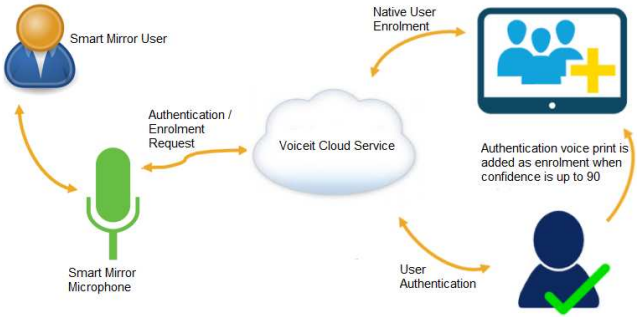

Fig. 6. System learning described

say the selected pass phrase, e.g., "remember to wash your hands before eating". Once the pass phrase is detected using the Google speech-to-text" API, it is saved as a wave file and represents the user's voiceprint. The voiceprint is then passed to the "createEnrollment" method of the VoiceIt API to create an enrollment for the user. Five enrollments are required from each user to ensure high accuracy.

- User Authentication

To authenticate a user using the voice recognition module, the microphone is turned on and the user is prompted to say the pass phrase saved during registration. The pass phrase may not be sensitive, but how the user says it is (i.e., the voiceprint). Again, like in the enrollment stage, the user's voiceprint is recorded as a wave file and passed as a parameter to the "authenticateUser method of the VoiceIt API. A response is sent in the form of a JSON file indicating whether the authentication is successful or not. A response code of "SUC" indicates success and a response code of "ATF indicates authentication failure, which is determined by a confidence value that ranges from 0 to 100. A value of 100 indicates a perfect match. The system has a learning capability wherein it enrolls new voice prints for the user using authentications that have a confidence of 90% and above. In this way, the system keeps learning little changes that may occur over time in the voice pattern of the user, as shown in Figure 6.

B. Voice Based Command

The system supports interaction between user and system. Users can control all functions using voice commands, including registration, authentication, searching, and signing out. The

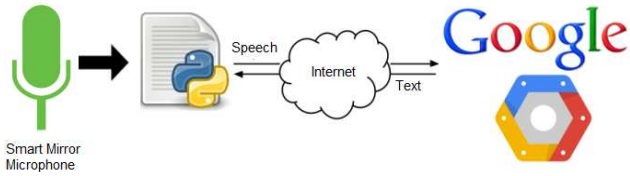

Fig. 7. Google speech to text

implementation features three primary technologies: Google's "Speech-to-Text [12], Google's "Custom Search Engine [13], and Amazon's "Alexa Voice Service (AVS)" [14].

1) Google Speech-to-Text Conversion: As shown in Figure 7, it enables developers to convert audio to text by applying powerful neural network models in an easy way. The API recognizes 120 languages and variants to support a global user base. It can be used to enable voice command-and-control, and more by processing real-time streaming or pre-recorded audio.

Speech-to-Text is implemented in "wake word" detection, "pass phrase" detection and in the registration module to accept the username. As explained, voice instructions are obtained by using the microphone to create recording of the users utterances. Each voice recording is passed to the Speech-to-Text API to obtain the text transcription for the audio file.

- Wake Word Detection

Like other voice-controlled systems, this project implements the use of a wake word. A wake word is a word or phrase that is uttered by the user to notify the system that he is ready to give an instruction. The system, upon detection of the wake word, begins to wait for instructions. The smart mirror uses the phrase "smart mirror" as the wake work. Once the smart mirror detects the wake word, it listens for the following instruction, such as login, logout etc.

- Pass phrase Detection

Pass phrase detection refers to accepting and validating a voiceprint from the user. This happens before user enrollment and user authentication.

- Registration Module

The registration module works a bit different than the two cases described above. This module gives the user a prompt and waits for a response. The module first asks the user to select and say his username. It then listens to the username before requesting a text transcription from the Speech-to-Text API. The user is prompted to confirm if the text transcription is accurate following which the system waits for a response, either "yes" or "no". Again, a text transcription of the users response is obtained. If it is "yes", the system proceeds to complete the registration. If it is "no", the system returns to the first step to collect username.

2) Google Custom Search Engine: it is a free platform that enables developers to perform customized searches based on Google Search. This service also allows users to drill down to only those actual webpages that are relevant. With an active Google account, users can create a free custom search engine by visiting the Google custom search page

https://cse.google.com/cse to obtain an API key and custom search engine ID. A Python snippet is shown below to demonstrate how to query a custom search engine.

```
query = 'python programming language';
sv = build('customsearch', 'v1',
        developerKey = key);
result = sv.cse().list(q = query, cx = cse_id,
        searchType = 'image', num = 5,
        fileType = 'png',').execute();
```

The above code snippet creates an array called "result" which contains the URLs to five images of PNG type that match the description of Python programming language. This project used a custom search engine to obtain multimedia contents based on user voice search queries, which is discussed in detail in the next section

3) Amazon Alexa Voice Service (AVS): AVS is a platform that Amazon has provided to allow developers integrate "Alexa" into their projects. With an AVS enabled product, users can stream media, get updates on weather, traffic, and news, as well as ask general knowledge questions. Alexa (the brain behind the AVS system) is triggered when a user says the wake word, "Alexa". Once the device detects the wake word, it begins to listen for a request. The request is then relayed to cloud based voice service. It is thus a requirement to have a functioning Internet connection. The cloud-based voice service interprets the request and sends a command back to the device or in this case, the smart mirror system.

Additional/custom skills - Skills are commands that Alexa can process. They are essentially applications but tailored specifically for AVS. Alexa provides a wide range of custom skills that are readily available to users of Alexa enabled devices. Perhaps the most impressive capability of AVS is that it allows the creation of custom skills that can be used to improve the richness of user experience. The number of existing custom skills runs into the tens of thousands. Figures 2 and 8 show the details of the Alexa custom skill architecture and the skill creation checklist, respectively.

This project created a new Alexa skill called "Smart mirror". This name represents what the uses say to indicate that they want to interact with the skill. The skill is created by visiting https://developer.amazon.com/alexa-skills-kit and selecting to create a new Alexa skill and listing the intents for the new skill. Intents are the various reasons/intentions a user could have for invoking the skill. Once the intents have been specified, the users need to create an interaction model. An interaction model represents how users interact with the new skill. The model contains various statements a user might say to interact with each of the intents in the new skill.

Once the model has been built, the Alexa device (e.g., the smart mirror) connected to the amazon developer account would be able to access the skill. A skill named "smart mirror" for instance may contain an intent called "getTime" and a sample utterance like "tell me the time". If a user says "Alexa, ask smart mirror to tell me the time", the Alexa Voice Service would recognize that the user is invoking the smart mirror skill.

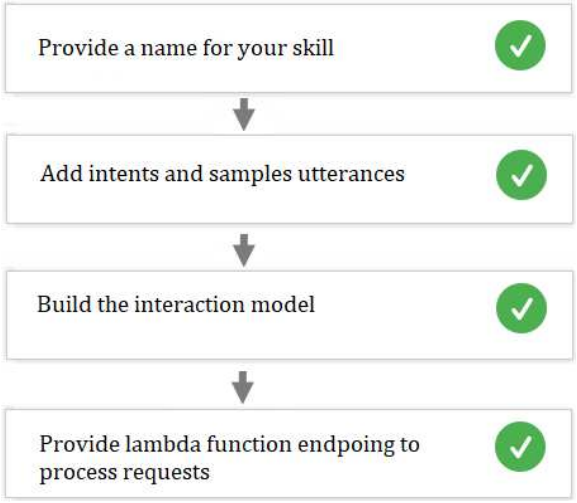

Fig. 8. Skill creation checklist [15]

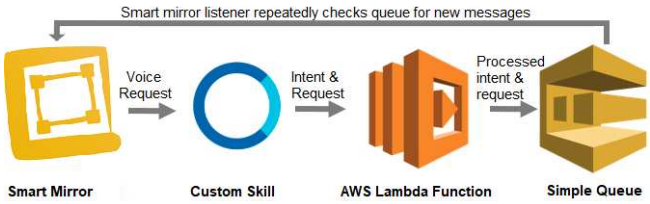

Fig. 9. Custom skill and lambda function structure

It would also recognize the users intention for requesting the skill - "getTime".

The new skill needs to be connected to a lambda function that will handle the skill requests. This can be done by specifying the endpoint of an Amazon Web Services (AWS) lambda function. The following section highlights the steps to create such a function. AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of the Amazon Web Services. It is a compute service that runs code in response to events and automatically manages the compute resources required by that code.

The authors followed the steps below to create a simple lambda function [16]. First, a free account to access the AWS console was created. Next, they created a new function from scratch and specified a programming language of choice. A variety of programming languages are supported. In this project, the authors used Python. In the "Add triggers" panel, the "Alexa Skills Kit" should be selected to indicate that the lambda function will be triggered by an Alexa skill and that the Alexa Skill will pass a JSON request containing the details of the intent and request to the lambda function. The AWS lambda console contains an embedded IDE (Integrated Development Environment) which is used to define the lambda function. The lambda function used in this project parses the JSON file from the Alexa Skill and pushes the intent and request details into a simple queue data structure that the Smart mirror can access, as shown in Figure 9.

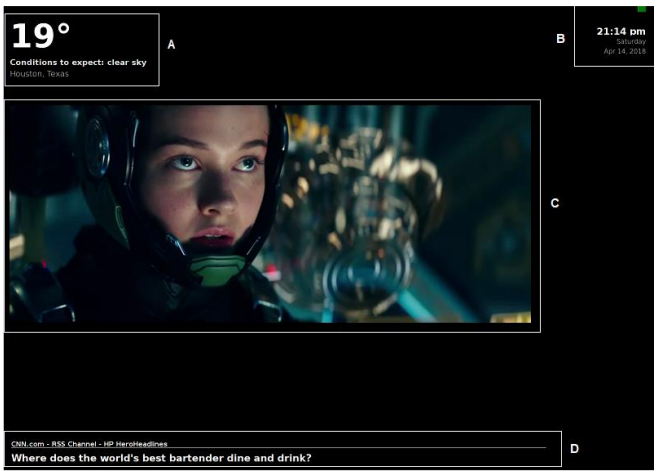

Fig. 10. Smart Mirror Software Display

C. Smart Mirror Software

The smart mirror software is developed using python programming language and PyQt 4 [17]. PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. The smart mirror has a display that is made up of four distinct sections as boxed in Figure 10.

1) Section A: This section displays weather and location information. The data is obtained from two APIs, namely freegeoip [18] and openweathermap [19]. Below is a code snippet showing how to extract data using the APIs.

```
//Call API to fetch city and region name
url = 'http://freegeoip.net/json/' + ip;
r = requests.get(url);
js = r.json();
//Parse response to obtain city and region
response = (str(js['city']) + ',' +
        js['region_name']);

//Call API to fetch temperature and weather conditions
//Data retrieval using developer ID (APPID)
r =  requests.get('http://api.openweathermap.
        org/data/2.5/weather?q = ' +
        str(js['city']) + '&APPID=b2a86......')
result = r.json();
//Parse response to get temperature in Kelvin
temperature = (str(int(math.ceil((result
    ['main']['temp']) - 273.15))) + u'\u00b0');
//Parse response to get weather conditions
condition = result['weather'][0]['description']
```

2) Section B: This section shows how time and date information is retrieved from the computer system. A code snippet is as follows:

```
//Assign current system date and time
now = datetime.datetime.now();
//format value to get current time & date
time = (now.strftime('%-H:%M %p'));
date = (now.strftime('%b %d, %Y'));
my_date = date.today();
day = (calendar.day_name[my_date.weekday()]);
```

This section also contains an indicator showing the status of the voice assistant. Green means that the voice assistant is enabled while red means that it is disabled. The voice assistant is automatically disabled after five minutes of inactivity. Once this happens, the user needs to be authentication again to continue using the system.

3) Section C: This section handles the display of multimedia content (e.g., videos, pictures, and audio files). This section works with a listener that constantly checks the AWS simple queue for new requests. For each new request, the listener obtains multimedia files matching the request using the Google custom search engine described before. Section C in Figure 10 shows a video being played. This is in response to the following request "Alexa, ask the smart mirror to play pacific rim."

4) Section D: This section shows news headlines pulled from 16 news stations. A news feed is displayed every five seconds. The code extract below shows how this section was implemented.

```
//Declare a counter to track progress
count = 0;
//Load RSS feed URLs in list
l =['http://rss.cnn.com/rss/cnn_topstories.rss',
    'http://www.tmz.com/rss.xml'];
//Use feedparser to load feeds
content = [];
f = feedparser
while count < len(l):
    try:
        temp = f.parse(l[count])
        if len(temp['entries']) > 0:
            content.append(temp)
        count = count + 1
    except:
        count = count + 1
        pass
//Get title/name of first RSS station
title = (content[0]'feed'][title_detail']
        ['value']);
//Get the first feed for the first station
feed = (content[0]['entries'][0]
        ['title_detail']['value']);
```

IV. Experimental Study

A series of tests were conducted to determine the appropriate values for some important system variables, including frame rate, camera resolution, face recognition confidence threshold, and voice recognition confidence threshold. Frame rate here refers to the number of frames that are extracted from the smart mirror webcam and displayed per second. The camera resolution refers to the amount of detail the camera can capture. This is further described by the height and width of pictures taken by the camera. Face recognition confidence refers to the distance between a face being identified and the closest match in the known face database. Voice recognition confidence refers to the possibility of a voiceprint being identified as a registered user's voiceprint record. It should be noted that due to the different definitions, the measure of the voice recognition confidence is the reverse of that of the

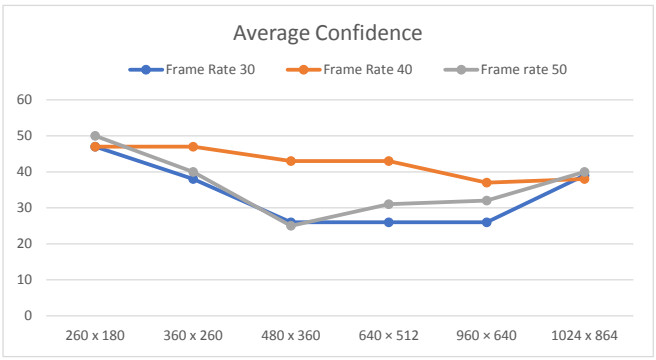

Fig. 11. Confidence comparison based on difference frame rates

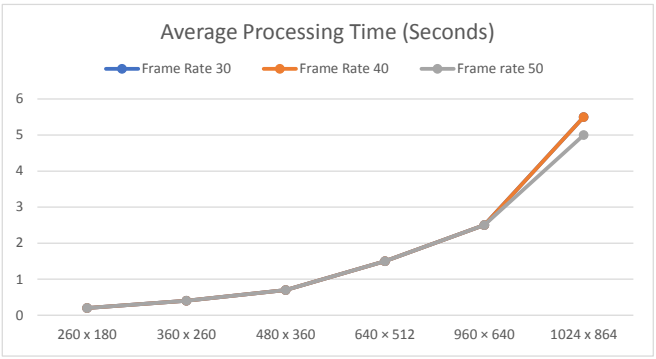

Fig. 12. Processing time comparison based on different frame rates

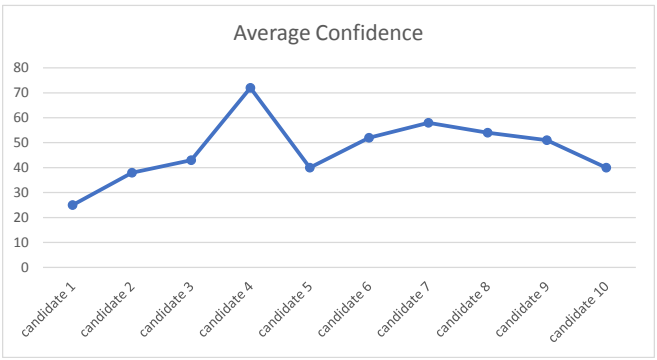

Fig. 13. Average confidence levels of 10 candidates on face recognition

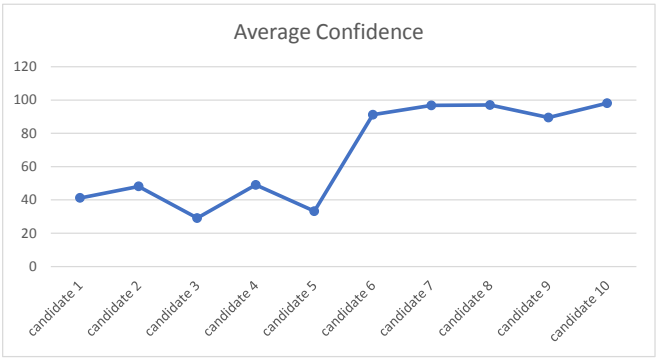

Fig. 14. Average confidence levels of 10 candidates on voice recognition

face recognition, i.e., a confidence of zero indicates a perfect match in face recognition, while a confidence of 100 indicates a perfect match in voice recognition.

A. *Tuning frame rate and camera resolution*

To find appropriate frame rate and resolution values, the authors enrolled a new user and measured the average confidence achieved for various frame rates and resolution values during authentication. Figure 11 shows a comparison of the confidence based on different frame rates, and Figure 12 depicts a comparison of the processing time based on different frame rates. Please note that the data of the frame rate at 30 are exactly same as that of the frame rate at 40. Thus, the two lines overlapped. The results indicate that higher resolution values significantly increase the time taken by the system to process each frame. Despite relatively accurate results, the processing time rules out the selection of higher resolution values. An optimal value, as can be observed across all graphs, would be to select a frame rate of 50 and a resolution of 480 x 360. This is the combination that obtains the lowest average confidence value, 25, while keeping frame processing time low.

B. *Finding appropriate confidence value for face recognition*

After determining the appropriate frame rate and camera resolution values, the system was further tested to find a confidence threshold for successful authentications. To do this, the authors used a room with a constant lighting and asked 10 candidates to attempt logging into the system. Among the candidates, candidate 1 was preregistered on the system to serve as the control setup. Thus, a lower confidence value for candidate 1 was expected during authentication.

As shown in Figure 13, candidate 1 achieved the lowest average confidence of 25. Candidate 2 had the second lowest average confidence of 38. This comes in at a safe distance from candidate 1 and alleviates fears of false acceptance. Finally, a confidence value of 28 was selected to serve as the required threshold for successful facial authentication of the system.

C. *Confidence measurement and testing for voice recognition*

Similarly, the smart mirror was tested to validate its voice recognition authentication. To do this, the face recognition was turned off and ten different candidates, including five unregistered users and five registered users, were asked to attempt logging into the system. All of the candidates knew the pass phrase and each of them attempted five times. The confidence value of each candidate was calculated based on the average confidence prediction of the voice recognition.

Figure 14 shows the system's performance on the ten candidates. As expected, all five registered candidates (candidates 6 to 10) have much higher confidence value than that of the unregistered users (candidates 1 to 5). After repeating the experiments among a number of candidates, a confidence value of 83 was selected to serve as the required threshold for successful voice authentication of the system.

Based on the confidence threshold, a set of experiments were conducted to verify the voice authentication accuracy

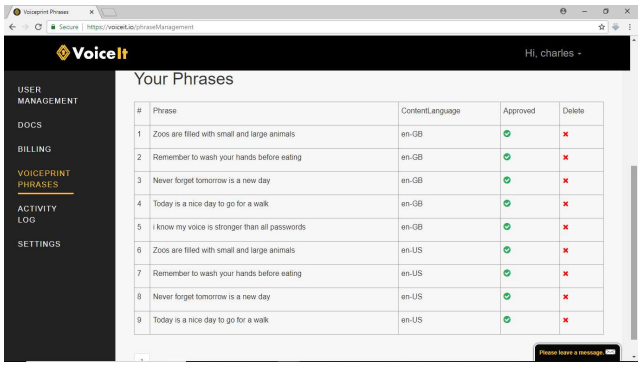

Fig. 15. Pass phrases in the VoiceIt account

TABLE I
COMPARISON OF SUCCESSFUL ATTEMPTS

Pass Phrase	RU	UU
Remember to wash your hands before eating	9	0
Never forget tomorrow is a new day	10	0
Today is a nice day to go for a walk	10	0
Zoos are filled with small and large animals	10	0

under different pass phrases. As described in Section III-A2, the VoiceIt API is used for authentication, where a registered user speaks his pass phrase to log into the system. In order to use the VoiceIt API, the authors created an account on voiceit.io. After logging into the account, one can create his own pass phrases, but the pass phrases have to be checked and approved by VoiceIt before application. The pass phrases displayed in Figure 15 are the default ones provided by VoiceIt. Table I compares the successful attempts between a registered user (RU) and an unregistered user (UU) under four different pass phrases. For each pass phrase, both users made 10 attempts to log into the system. The result shows that the voice authentication has high accuracy level. The unregistered user was never authenticated to log in. As a comparison, except for one time on the first pass phrase, the system always succeeded to recognize the registered user and authorize him to log in. Similar results were also found through the experiments conducted on other users.

V. CONCLUSION AND FUTURE WORK

This paper presented a smart mirror system that has integrated several impressive features with a user-friendly architecture. A reliable and easy to use design was implemented following a service-oriented approach. In today's world of interconnected devices, security cannot be ignored. Considering this, this system was equipped with a strong authentication framework to ensure end to end security of the whole system. This feature sets this project aside from other similar works. The future offers endless possibilities for advancement of the prototype. The most notable is the possibility for the user to take the smart mirror display around the entire home. This can be achieved by connecting the smart mirror to a smart projector [20] that is able to project the smart mirror display on various surfaces (walls, table tops etc.) around the home. With an added camera and a microphone, the user would be able access the full features of the smart mirror from other parts of the home without compromising security.

ACKNOWLEDGMENT

This research is supported in part by the National Science Foundation under grant no. 1712496. Any opinions, findings, and conclusions expressed in this paper are those of the authors, and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank Sheikh Tareq Ahmed for helping with the experiments and the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

REFERENCES

[1] R. P. I. Foundation, "Teach, learn, and make with raspberry pi," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/
[2] A. Orr, "Use python to build a raspberry pi-powered home security camera," 2018. [Online]. Available: https://medium.com/@andyorr/use-python-to-build-a-raspberry-pi-powered-home-security-camera-for\-50-84ab7e344e2d
[3] "Web enabled pool control - raspberry pi," 2016. [Online]. Available: http://www.instructables.com/id/Web-Enabled-Pool-Control-Raspberry-Pi/
[4] M. Patkar, "6 best raspberry pi smart mirror projects we've seen so far," 2016. [Online]. Available: https://www.makeuseof.com/tag/6-best-raspberry-pi-smart-mirror-projects-weve-seen-far/
[5] P. Research, "Meet philips research - research." [Online]. Available: http://www.research.philips.com/technologies/misc/homelab/index.html
[6] T. Lashina, "Intelligent bathroom," 2014. [Online]. Available: https://www.researchgate.net/publication/228881021_Intelligent_bathroom
[7] A. Kasal and S. Ewen, "A project of the interactive mirror with artsy visuals in lit studios," *International Journal of Engineering Technology, Management and Applied Sciences*, vol. 5, no. 5, pp. 1863–1869, May 2017.
[8] A. Bostrom and F. Ramstrom, "Head up display for enhanced user experience," *Chalmers University of Technology in Sweden*, 2014. [Online]. Available: http://publications.lib.chalmers.se/records/fulltext/223949/223949.pdf
[9] "OpenCV library." [Online]. Available: https://opencv.org/
[10] "VoiceIt Technologies FAQ." [Online]. Available: https://voiceit.io/faq
[11] "VoiceIt Technologies LLC." [Online]. Available: https://voiceit-files.s3.amazonaws.com/OpusResearch-Biometrics-Equal-Error-Rate.jpg
[12] "Cloud Speech-to-Text - Speech Recognition." [Online]. Available: https://cloud.google.com/speech-to-text/
[13] "Google Custom Search." [Online]. Available: https://developers.google.com/custom-search
[14] "Alexa Voice Service - Integrate Alexa Directly into Your Connected Products." [Online]. Available: https://developer.amazon.com/alexa-voice-service
[15] D. Isbitski, "Announcing new alexa skill builder (beta), a tool for creating skills," *Amazon Developer Services*, April 2017. [Online]. Available: https://developer.amazon.com/blogs/alexa/post/02d828b6-3144-46ea-9b4c-5ed2cbfadb9c/announcing-new-alexa-skill-builder-beta-a-tool-for-creating-skills
[16] "Aws lambda serverless compute - amazon web services," *Amazon Web Services*. [Online]. Available: https://console.aws.amazon.com/lambda
[17] "What is PyQt?" [Online]. Available: https://riverbankcomputing.com/software/pyqt/intro
[18] "IP geolocation web server." [Online]. Available: https://github.com/fiorix/freegeoip
[19] "Weather API." [Online]. Available: https://openweathermap.org/
[20] C. Lim, J. Choi, J.-I. Park, and H. Park, "Interactive augmented reality system using projector-camera system and smart phone," in *IEEE International Symposium on Consumer Electronics (ISCE)*, 2015, pp. 1–2.