Received: 19 September 2023, Revised: 18 October 2023, Accepted: 16 December 2023, Online: 30 December 2023

DOI: https://dx.doi.org/10.55708/js0212003

Smart Monitoring System for Housing Societies based on Deep Learning and IoT

Neha Koppikar*, Nidhi Koppikar

Department of Data Science MPSTME, SVKMs NMIMS University, Mumbai, India

*Corresponding author: Neha Koppikar & neha.koppikar@gmail.com

ABSTRACT: Since 2020, people have been getting their body temperatures checked at every public location, social distancing has become a norm, and it has become essential to know who has been in contact with whom. Therefore, we needed a system that helped us solve these challenges, especially in housing societies, as most of the general public stayed home more than ever. Therefore, it has become essential to safeguard housing societies. There has been a lot of research in building a security system, but there needs to be more research that targets housing societies as the end users. We have devised a possible solution, including a facial recognition system with body temperature sensing on a Raspberry Pi. The best part of our application is the automated data collection page on a web application, which makes collecting facial images more straightforward and faster. Code for this project can be found at: https://github.com/NehaKoppikar/Monitoring-System-for-Housing-Societies-using-Deep-Learning-and-IoT

KEYWORDS Face Recognition, Raspberry Pi, Edge Vision, Body Temperature, Sensors

1. Introduction

The security of housing societies is paramount to all of us. Currently, security services are provided with the help of watchmen or security guards. However, mistakes on their part, intentional or unintended, do happen, and the people held responsible for such anomalies are either our guardians (security guards/watchmen) or the housing society's cooperation. Technology can assist all the stakeholders in making security arrangements watertight to a greater extent. Anomalies due to misjudgment or carelessness considerably affect the residents of the housing society. There can be thefts and burglaries, and in recent times, this inadvertence can act as a catalyst for the spread of the COVID-19 pandemic into our homes. There should be some way to backtrack if anything goes wrong.

There are a lot of researchers working in this domain and have built a lot of systems to combat the COVID-19 situation. When writing this research paper, 211 papers appeared when searching for "COVID-19" at arxiv.org. There are face-recognition attendance systems [1], systems built on embedded devices for calculating body temperature [2], and so many other research papers have already been published on this topic or related topics that we can come across something new every week if not every day. As part of recognizing someone, bio-metrics has changed the attendance world. There was a time when the only way to acknowledge someone officially was their signature, and a lot has changed since then. Various security systems have been built, but no research has been found targeting housing societies.

We can now automatically store a person in the database without the person being actively involved. All that is required from the person is showing up, and the rest is taken care of. Even though it is possible, there is minimal digital initiatives taken to augment security guards in housing societies. This paper targets this gap.

Along with face recognition, checking a person's body temperature has become extremely important in the pandemic. That is why the three main parts of our solution involve face recognition, body temperature sensing, and regularly sending email notifications to a responsible point of contact.

What makes this paper unique is that even though there are many research papers with biometrics in a security system, there was no research that sends automated email reports to a responsible point of contact in a housing society.

The key contributions of this research are as follows:

- Face-Recognition web application built on streamlit
- Body temperature sensing integrated with face-recognition
- Face-recognition and body temperature sensing data stored automatically in a NoSQL database
- Stored data report periodically sent to a responsible point of contact
- Entire application deployed in Raspberry Pi

2. Literature Survey

To be able to build our solution, we referred a few research papers, which include:

1. MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19 [3]

   This paper is about a dataset built for face-recognition purposes on masked data. Many face-recognition systems are made in a way where the face-recognition algorithm gets triggered once a face is detected, and many times, the face should be entirely facing the input source (camera). A significant challenge faced by face-recognition systems was met because faces were not detected when covered in a mask. Rebuilding a design as per the requirement needs a vast dataset.

   The dataset on which this research paper is based helps rebuild face-recognition systems on an existing algorithm, FaceNet [4].

   This paper is better because it also addresses the issue of people needing to wear a mask properly. The dataset is labeled so the person is correctly masked or incorrectly masked. If incorrectly masked, it also addresses whether it is due to an uncovered chin, uncovered nose, or uncovered nose and mouth.

2. Monitoring System Heartbeat and Body Temperature Using Raspberry Pi.

   This paper concerns a wireless sensor network (WSN) system to monitor body temperature and heartbeat. The authors of this paper have built this system using Raspberry Pi and Arduino. They have also created a user-friendly website.

   We have taken inspiration from this paper to build the body temperature component of our system.

3. Impact of thermal throttling on long-term visual inference in a cpu-based edge device [5]

   This paper is a comparative study on various levels. It compares multiple Raspberry Pi devices, multiple versions of operating systems, and different CNN-based algorithms, with or without a fan. The results are tested with as many combinations as possible; fans improve efficiency irrespective of the variety they are tested on.

   We tried to use a fan in our approach as well. The issue we found was that we could use the camera or the fan, as the camera came in the way of the fan when it rotated, affecting the camera's connection with the Raspberry Pi board.

4. Smart Security for an Organization based on IoT [6]

   This paper concerns a security system to safeguard an organization from fire and intruders. The researchers have built an android application, which acts as an output source in case any anomaly gets detected.

5. Development of Face Recognition on Raspberry Pi for Security Enhancement of Smart Home System [7]

   This paper is about a face-recognition system that is built for houses. It magnetically locks or unlocks the door based on the output of the face-recognition model.

   This approach highly inspires our face-recognition system.

3. Research Design

The research methodology was designed as follows:

3.1. Define Research Problem

Our research problem is to build a security system to store when a person enters a housing society and their body temperature. We also wanted to ensure that the data is collected from the application deployed on an embedded device.

3.2. Literature Review

After defining the problem statement, we individually reviewed similar papers on Google Scholar.

3.3. Architecture Design

We designed the architecture for the application, as shown in figure 1.

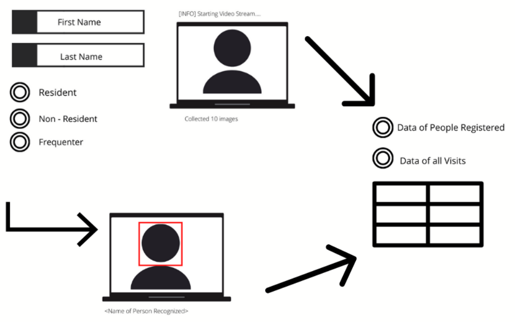


Figure 1: Architecture Design

For body temperature sensing, we designed the architecture as shown in figure 2.

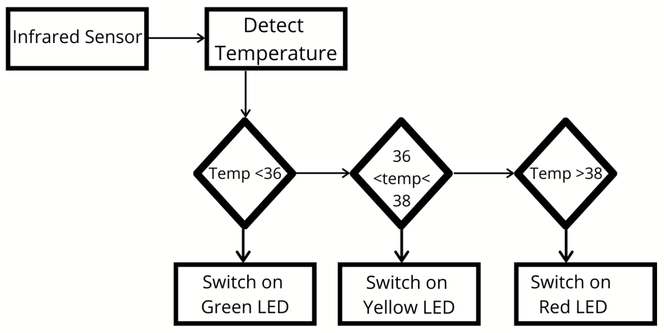


Figure 2: Body Temperature Architecture Design

3.4. Data Collection

Data was collected through the web application [8], as shown in figure 3.

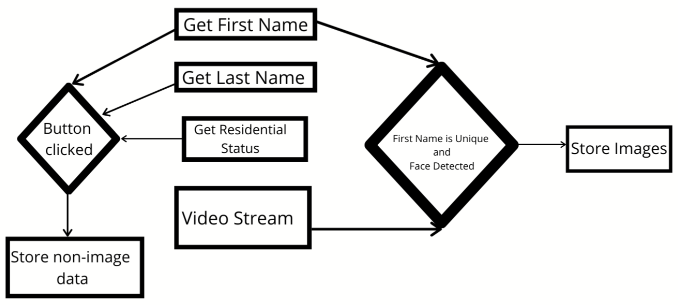


Figure 3: Data Collection

3.5. Building the application

We tried various frameworks and decided to build the application on Python Streamlit as it had fewer challenges and a faster turnaround time.

4. Proposed Work

4.1. Features (and our contributions)

4.1.1. Face-Recognition

We have used Adam Geitgey's Face Recognition library on the Streamlit framework. Using the same library, we also generated face embeddings.

This face-recognition algorithm is a series of several related problems:

- Identify a face in a video
- Focus on the face, check lighting conditions, and confirm if the person is correctly identified
- Pick unique features like eyes, nose, and so on
- Compare unique features with the training dataset

4.1.2. First-time Registration (Data Collection and storage)

We created a directory to store all the images. Every time the user starts writing their name, a directory gets automatically created inside the images directory, with the person's name being recorded into the system. This process is rapid, and even a second pause leads to the creation of the directory. Once the directory is created, ten images are automatically stored in the system. All the images are collected and stored in about 3 or 4 seconds.

Only one image is required to create the face encoding using Adam Geitgey's library [9]. Despite that, we are going ahead with this approach so that it gives us and any other person who wants to build upon this project the freedom to not only choose one-shot learning-based methods for face recognition but also try different approaches. The code for this is private in its complete form.

4.1.3. Body Temperature Sensing on Raspberry Pi

We have connected an infrared sensor with the Raspberry Pi. This sensor constantly detects the temperature; we can view it on Raspberry Pi's command line. The temperatures are in degrees centigrade. There are Python packages available to help achieve this. We have used smbus2, PyMLX90614 v.0.0.3, gpiozero, and RPi.GPIO.

4.1.4. Maintaining and Viewing the database

We have created a database using MongoDB to store the names, dates, and times of new records and anyone who visits the housing society [10]. The visitor could be a resident, a frequent visitor, or a nonresident is shown in figure 4.

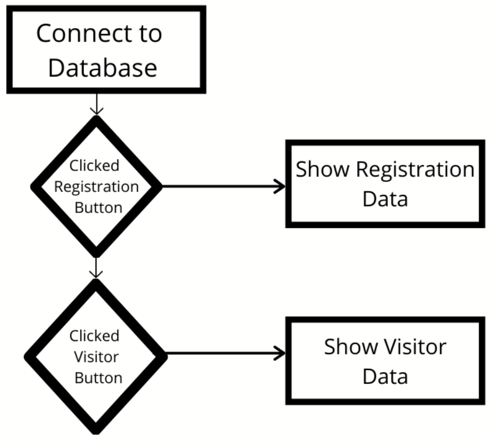


Figure 4: View Database

4.1.5. Sending Report via Email

We have used the SendGrid API, which is more secure than the SMTP client python package. The SendGrid API is also based on the Simple Mail Transfer Protocol (SMTP) principle. The difference is that the Python package required lowering the security of the receiver's GMail Account. In contrast, SendGrid authenticates and verifies the sender, and there is no action required from the receiver to be able to receive an email [11]. The Sendgrid mail flow is shown in figure 5.

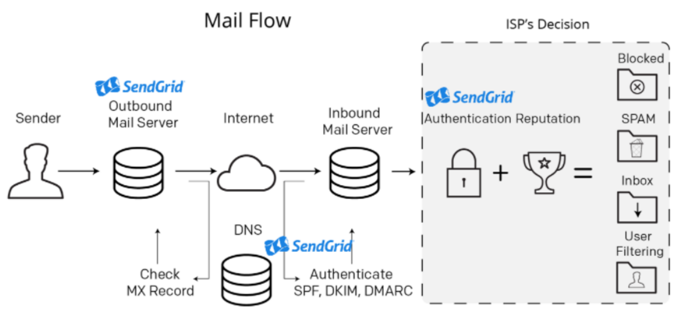
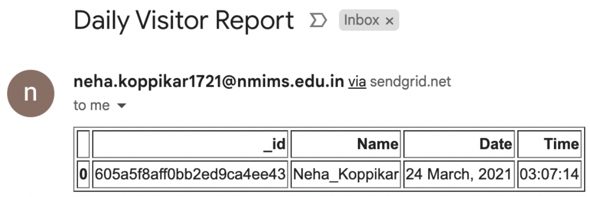


Figure 5: Sendgrid Mail Flow

The content sent via email is a dataframe generated from the MongoDB Database that is converted into the Hypertext

Markup Language.

The emails are then scheduled using Python's schedule package. The notification is shown in figure 6.


Daily Visitor Report

neha.koppikar1721@nmims.edu.in via sendgrid.net
to me

	_id	Name	Date	Time
0	605a5f8aff0bb2ed9ca4ee43	Neha_Koppikar	24 March, 2021	03:07:14

Figure 6: Email Notification

4.2. Materials

Software:

1. MongoDB (Database) [12]
2. Python (Programming Language) [13]
3. Streamlit (Web Framework) [14]
4. SendGrid API (Email) [15]
5. Remote Desktop Connection (Connecting Laptop with Raspberry PI)

Hardware:

1. Raspberry PI 4 Model B (4GB)
2. Raspberry PI 5MP Camera Board Module
3. LED Bulbs (Red, Yellow, Green)
4. 1 kilo ohm resistors
5. Breadboard
6. Infrared Temperature Sensor GY-906 MLX90614
7. Female to Female Wires (Infrared Sensor and Raspberry Pi Connections)
8. Male to Female wires (LEDs and resistors)
9. BIS 3 Amps Charger, USB C Cable
10. Ethernet Cable

4.3. Application walk trough

4.3.1. First-Time Registration

How it works:

This page of the application is responsible for the initial data collection. This is where the details the following details are collected:

1. First Name
2. Last Name
3. Images for facial recognition
4. Which category does the person fall under? (Resident, Non-Resident, Frequenter)

The resulting page is shown in the figure 7.

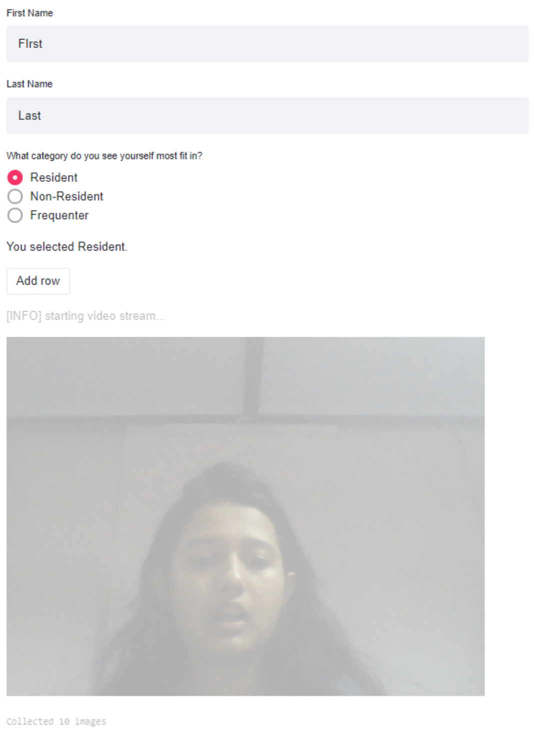

First Name
First
Last Name
Last
What category do you see yourself most fit in?
Resident [✓]
Non-Resident
Frequenter
You selected Resident.
Add row
[INFO] starting video stream...
Collected 10 images

Figure 7: Registration Page - Screenshot

When anyone starts typing in the "First Name" blank, the system is ready to make a directory based on the input collected from the registration page. The person organizing the records has to be quick in typing as even a second of a pause leads to the directory being created, and ten images automatically get clicked and stored in the new directory, and it gets printed on the screen that ten images have been collected.

Due to the pause issue, this page also has an "Add row" button under the category radio option so that the registration record of names and categories gets correctly stored in the database. This can also help in removing the directories that are not required.

Dependencies

Python packages used:

1. Streamlit (streamlit) [14]
2. OpenCV (cv2) [16]
3. Pandas (pandas) [17]
4. Time (time)
5. Datetime (datetime)
6. PyMongo (pymongo) [18]
7. OS (os)
8. Sendrgid API

4.3.2. Face-Recognition

How it works

The camera detects a person, compares the faces with the face embedding [9], and predicts to recognize the person if it is confident more than 60 percent [19].

As soon as the face is recognized, the name, date, and time get automatically stored in the database. This is the other data collection component of the application. The resulting page is shown in the figure 8

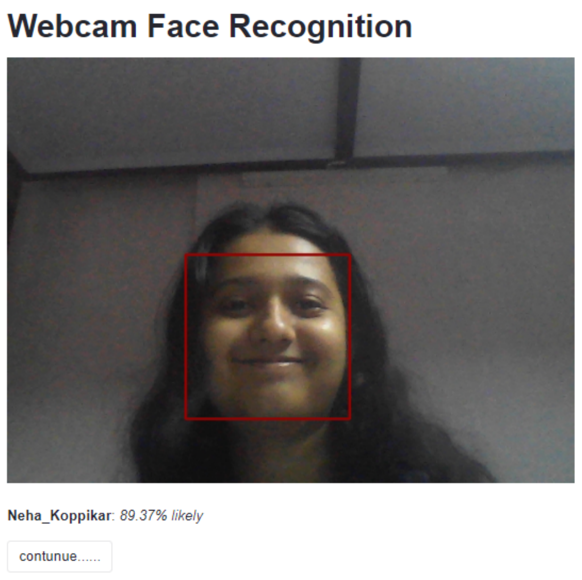


Figure 8: Face Recognition Page - Screenshot

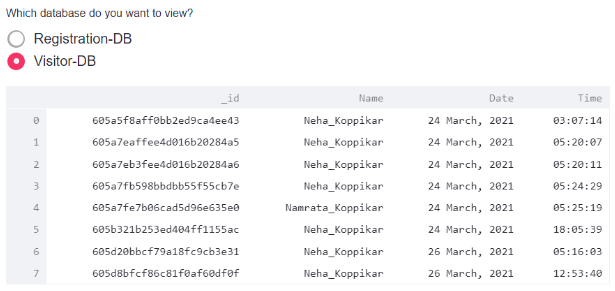
Dependencies
Python Packages used:

1. Streamlit (streamlit) [14]
2. OpenCV (cv2) [16]
3. Face Recognition (face-recognition) [20]
4. Dlib-ml: A Machine Learning Toolkit [21]
5. Pandas (pandas) [17]
6. Numpy (numpy) [22]
7. Pickle (pickle)
8. Datetime (datetime)
9. PyMongo (pymongo) [18]

4.3.3. Viewing Database

Why this feature: At times, the maintainer may want to see if the application is working correctly or not. This feature helps check that. The maintainer will remember the recent entries anyway as the maintainer is most probably a security who is habituated to keeping track of these things. The maintainer can view the database and compare it with their memory. Since the data cannot be edited from the application, the data is not compromised.

How it works:
All the data stored in the database is accessed as a data frame and displayed on the screen. The visitor data is shown in figure 9.


Which database do you want to view?

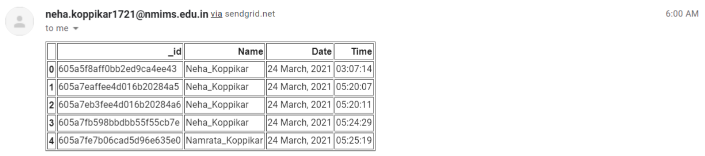
Registration-DB [✓ Visitor-DB]

	_id	Name	Date	Time
0	605a5f8aff0bb2ed9ca4ee43	Neha_Koppikar	24 March, 2021	03:07:14
1	605a7eaffee4d016b20284a5	Neha_Koppikar	24 March, 2021	05:20:07
2	605a7eb3fee4d016b20284a6	Neha_Koppikar	24 March, 2021	05:20:11
3	605a7fb598bbdbb55f55cb7e	Neha_Koppikar	24 March, 2021	05:24:29
4	605a7fe7b06cad5d96e635e0	Namrata_Koppikar	24 March, 2021	05:25:19
5	605b321b253ed404ff1155ac	Neha_Koppikar	24 March, 2021	18:05:39
6	605d20bbcf79a18fc9cb3e31	Neha_Koppikar	26 March, 2021	05:16:03
7	605d8bfcf86c81f0af60df0f	Neha_Koppikar	26 March, 2021	12:53:40

Figure 9: Visitor Database - Screenshot

4.3.4. Sending Emails

Why this feature:
While the application will be used mainly by the security guards, the society's secretary is also answerable in the case of an anomaly. That is why there is a requirement to keep a check on what is happening, and checking the system now and then can be inconvenient to both the maintainer of the application and the secretary. This feature helps solve this problem by emailing a report to the secretary. For this purpose, we have used the SendGrid API [15].
How it works:
The database stores all the face-recognition data as a data frame and this data frame is sent as a report to the receiver of the email, which is preferably the secretary of the society. The resulting email is shown figure 10.


neha.koppikar1721@nmims.edu.in via sendgrid.net
to me
6:00 AM

	_id	Name	Date	Time
0	605a5f8aff0bb2ed9ca4ee43	Neha_Koppikar	24 March, 2021	03:07:14
1	605a7eaffee4d016b20284a5	Neha_Koppikar	24 March, 2021	05:20:07
2	605a7eb3fee4d016b20284a6	Neha_Koppikar	24 March, 2021	05:20:11
3	605a7fb598bbdbb55f55cb7e	Neha_Koppikar	24 March, 2021	05:24:29
4	605a7fe7b06cad5d96e635e0	Namrata_Koppikar	24 March, 2021	05:25:19

Figure 10: Email - Screenshot

Dependencies
Python Packages used:

1. SendGrid (sendgrid) [15]
2. Schedule (schedule)
3. Datetime (datetime)
4. Pandas (pandas) [17]
5. PyMongo (pymongo) [18]

4.3.5. Raspberry Pi

Initial Set up:

1. Install Raspbian OS into SD card
2. Insert the SD card
3. Connect Raspberry Pi to PC [23]
4. Find the IP Address of the Raspberry Pi

5. Connect the laptop with the Raspberry Pi remotely [24]

   The above steps are generic.

Important Changes from the web application:

1. The camera is accessed from the laptop using the OpenCV Python package. The picamera is accessed from the Raspberry Pi using the imutils [25] python package.
2. There are dependency issues in installing MongoDB into Raspberry Pi locally. Alternatively, Ubuntu provides an unofficial Mongodb package.

Schematic Representation:
The Raspberry Pi Schematic Representation is shown in the figure 11.

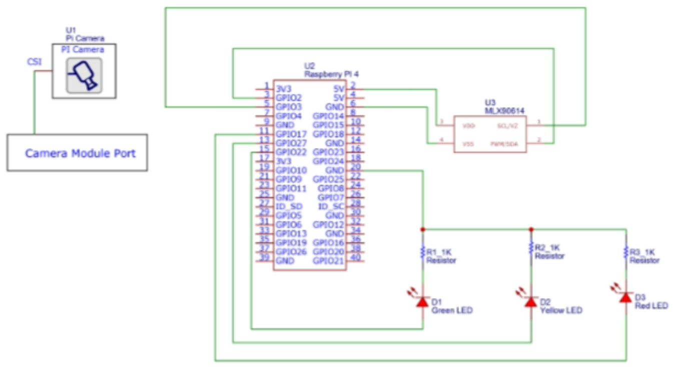


Figure 11: Raspberry Pi Schematic Representation

4.3.6. *Body Temperature Sensing*

How it works: In this project, we use a contactless Infrared (IR) Digital Temperature Sensor called MLX90614. This sensor makes use of IR rays to measure the temperature of a person or object without any physical contact with it. The communication between the sensor and Raspberry Pi 4 is established using the I2C protocol [26]. The severity of the sensed temperature is shown via an LED lighting system inspired by a traffic signal. The system is connected to the microcontroller with the help of wires. The working of the temperature sensing system can be summarized in the following points:

1. The person or object whose temperature is to be measured is brought in the range of the temperature sensor.
2. The infrared sensor senses the temperature and returns it to the Raspberry Pi 4.
3. With the help of the LED system, we can identify a person's temperature in three categories: Normal, Borderline, and High.
4. When the temperature sensed is below 37 degrees centigrade, the Green LED turns ON, indicating Normal temperature.
5. When the temperature sensed is between 37 degrees centigrade and 38 degrees centigrade, the Yellow LED turns ON, indicating Borderline temperature.
6. When the temperature sensed is above 38 degrees centigrade, the Red LED alone turns ON, indicating the temperature to be High.

Dependencies:

1. smbus2
2. PyMLX90614 0.0.3
3. gpiozero
4. RPi.GPIO

5. Alternatives considered

5.1. *Frameworks*

1. Django (Python-Based Web Framework) [27]
2. Flask (Python-Based Web Framework) [27]
3. React (JavaScript-Based Web Framework)

5.2. *Database*

1. MySQL
2. PostgreSQL

5.3. *Sending Emails*

Python's SMTP client can also be used. The primary issue is that it requires a few settings in our GMail accounts, which lowers the security level and increases the risk of hacking.

5.4. *Accessing Raspberry Pi*

We have used Remote Desktop Connection. VNC Viewer can also be used.

5.5. *Challenges Faced*

1. Remote Desktop Connection with the Raspberry is lost after about an hour.
2. Integrating the entire application seamlessly.
3. When we started, we were not well-versed with any framework, including Flask.
4. The issue that we faced with the flask was that we were not able to stream the input from the webcam for the registration page and the face recognition page at the same time. (Navigation bar used). That is why we switched to Streamlit.

5.6. *Limitations of Proposed System*

1. Cannot toggle between the database option when running the application file. To view the database, the file has to be run separately.
2. Streamlit Installation into Raspberry Pi was solved by downgrading the version [23].

6. Discussion

Our experiment shows that even though biometrics is essential in a security or monitoring system, what improves it even more is a email report notification system to be in place. This project showcases a smart monitoring system that stores images, fine-tunes a face-recognition model using one-shot learning, stores the images and personal information of a person, and sends email reports to a responsible point of contact daily. The web application also allows viewing the database that cannot be edited or changed, thereby acting as a security feature.

There are a few limitations to this research project, though. A better security design or mechanism can be built that not only stores and sends data but also alerts the responsible point of contact in case of an anomaly situation. This application cannot be scaled up or down quickly as it is built on-prem. Therefore, making it on a cloud computing platform could have been a better choice and scope for future research.

7. Limitation

A few limitations of the application are:

1. The hardware disconnects from the software sometimes.
2. Considering the sensitivity of the data we are using, a security system to ensure data privacy is not in place.

8. Future Scope

1. Adding the feature to detect mask positioning [6].
2. Make the application more user-friendly
3. The application has to be made spoof-proof as a layer of security [28].
4. Temperature cannot be backtracked. Figure out how to backtrack the temperature and link that with the person into the database efficiently.
5. If this application is deployed on the cloud. It will help with scaling the application.
6. Face Recognition frameworks like FaceNet and OpenFace can be used.
7. A system to ensure the security and privacy of the data in the future.

9. Result and Conclusion

We are glad that the application is working, and we learned a lot. We learned about frameworks in Python, ways to send an email, schedule an email, face recognition, and manage databases, among many others. Though there are a lot of places where this application can be improved, it should be a good starting point for someone who wants to work on this application at a bigger level. In the future, we can use a better face-recognition algorithm, a cloud computing platform, or apply action recognition to identify harmful actions and capture the person's name for enhanced security.

Acknowledgment We want to thank Mr. Pranav Nerurkar (https://orcid.org/0000-0002-9100-6437), our research paper mentor for his valuable guidance on how to write a research paper.

We would also like to thank Mr. Yash Tomar, our Project Deep Blue Mentor (of Mastek), who brainstormed many ideas with us.

Declaration of competing interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] D. R.S, "Attendance authentication system using face recognition", *Journal of Advanced Research in Dynamical and Control Systems*, vol. 12, pp. 1235–1248, 2020, doi:10.5373/JARDCS/V12SP4/20201599.

[2] T. Sollu, Alamsyah, M. Bachtiar, B. Bontong, "Monitoring system heartbeat and body temperature using raspberry pi", *E3S Web of Conferences*, vol. 73, p. 12003, 2018, doi:10.1051/e3sconf/20187312003.

[3] A. Cabani, K. Hammoudi, H. Benhabiles, M. Melkemi, "Maskedface-net – a dataset of correctly/incorrectly masked face images in the context of covid-19", *Smart Health*, 2020, doi:https://doi.org/10.1016/j.smhl.2020.100144.

[4] F. Schroff, D. Kalenichenko, J. Philbin.

[5] T. Benoit-Cattin, D. Velasco-Montero, J. Fernández-Berni, "Impact of thermal throttling on long-term visual inference in a cpu-based edge device", 2020.

[6] M. Saifuzzaman, A. Hossain, N. Nessa, F. Nur, "Smart security for an organization based on iot", *International Journal of Computer Applications*, vol. 165, pp. 33–38, 2017, doi:10.5120/ijca2017913982.

[7] T. Gunawan, M. Gani, F. Rahman, M. Kartiwi, "Development of face recognition on raspberry pi for security enhancement of smart home system", *Indonesian Journal of Electrical Engineering and Informatics*, vol. 5, pp. 317–325, 2017, doi:10.11591/ijeei.v5i4.361.

[8] K. Naik, "Deep-learning-face-recognition", 2020.

[9] A. Geitgey, "Save face encodings", 2018.

[10] "miscmonggodb raspberry pi installation, unable to locate package mongodb-org", .

[11] S. AGNEW, "How to send emails in python with sendgrid,", 2019.

[12] K. P. R. Eliot Horowitz, Dwight Merriman, "Mongodb", 2009.

[13] G. Van Rossum, F. L. Drake Jr, *Python tutorial*, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[14] C. K. F. A. J. A. J. R. J. M. N. S. T. R. Ashish Shukla, Charly Wargnier, "Streamlit - the fastest way to build and share data apps", 2019.

[15] J. L. Isaac Saldana, T. Jenkins, "Sendgrid", 2009.

[16] I. Intel Corporation, Willow Garage, "Opencv", 2000.

[17] W. McKinney, "Pandas", 2008.

[18] A. Sottile, "Pymongo", 2017.

[19] Nyahua, "Face recognition from webcam using streamlit", 2020.

[20] A. Geitgey, "Face recognition", 2017.

[21] D. E. King, "Dlib-ml: A machine learning toolkit", *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[22] T. Oliphan, "Numpy - the fundamental package for scientific computing with python.", 2006.

[23] E. Krupesh, "How to connect raspberry pi to pc (smartphone hotspot, no lan cable)", 2019.

[24] educ8s.tv, "Raspberry pi remote desktop connection", 2019.

[25] "I need this code to use the pi camera and not a webcam", .

[26] "Mlx90614 non-contact ir temperature sensor", .

[27] S. W. Adrian Holovaty, "Django (web framework)", 2005.

[28] S. Kumar, S. Singh, J. Kumar, "A comparative study on face spoofing attacks", "2017 International Conference on Computing, Communication and Automation (ICCCA)", pp. 1104–1108, 2017, doi: 10.1109/CCAA.2017.8229961.