SAFE ZONE AI POWERED AND AUTOMATED SURVEILLANCE SYSTEM

A PROJECT REPORT

Submitted by

ABISHEK M [211420104007]

NAVEEN KUMAR S [211420104330]

LAKSHMI PRIYAN P [211420104143]

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University,

Chennai)

APRIL 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report "**SAFE ZONE AI POWERED AND AUTOMATED SURVEILLANCE SYSTEM**" the bonafide work of "**ABISHEK M (211420104007), NAVEEN KUMAR S (211420104330), LAKSHMI PRIYAN P (211420104143)**" who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA,M.E.,Ph.D.,	**Dr.L.JABASHEELA,M.E.,Ph.D.,**
HEAD OF THE DEPARTMENT	**HEAD OF THE DEPARTMENT**
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,	NASARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above mentioned students were examined in End Semester Project

Viva- Voce held on...........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **ABISHEK M (211420104007), NAVEEN KUMAR S (211420104330), LAKSHMI PRIYAN P (211420104143)** hereby declare that this project report titled "**SAFE ZONE AI POWERED AND AUTOMATED SURVEILLANCE SYSTEM**" , under the guidance of **DR L.JABASHEELA M.E., Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

Name of the students

ABISHEK M (211420104007)

NAVEEN KUMAR S (211420104330)

LAKSHMI PRIYAN P (211420104143)

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for the support throughout the project and granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Dr. G. SENTHILKUMAR M.E.,Ph.D.,** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

ABSTRACT

The use of face recognition technology in smart door closure systems is becoming increasingly popular because of its convenience, security and accuracy. In this article, we offer a smart door system connected with windows that uses face recognition and surveillance.The system consists of a camera module, a microcontroller Raspberry Pi, and an advanced locking mechanism. Motion sensor senses and turns on camera power. The module captures an image of the person standing at the gate, and our system processes the image using a facial recognition algorithm to identify the person. If the person is authorized, the door will unlatch automatically. The proposed system has been tested using a dataset of faces, and the windows are monitored. The system also includes a backup mechanism, such as a keypad or key, in the event of a malfunction or unavailability of facial recognition. If friends and family come, the owner will be notified on time. If an unfamiliar person reaches the door, the system begins recording. The proposed system can be used in different locations such as homes, offices and hotels, to provide a secure and convenient way to access the premises throughout the building. The system is cost-effective, scalable and easy to install, making it an appealing option for smart door locking solutions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	**-**	Convolutional Neural Network
RTSP	**-**	Real Time Streaming Protocol
PIR	**-**	Passive Infrared Sensor
YOLO	**-**	You Only Look Once
HTTP	**-**	Hyper Text Transfer Protocol
FCM	**-**	Firebase Cloud Messaging
S3	**-**	Simple Storage Service
NMS	**-**	Non-Maximum Suppression

CHAPTER 1

INTRODUCTION

1.1 Overview:

Access control systems are vital for safeguarding premises and ensuring the security of individuals and assets. Traditional methods like mechanical keys and magnetic stripe cards are becoming outdated due to their limitations in security, convenience, and scalability. With advancements in technology, biometric identification systems, particularly facial recognition, are gaining popularity due to their precision, reliability, and user-friendliness.

In this article, we propose an advanced gate system utilizing facial recognition technology to authorize access. Additionally, windows are equipped with sensors that connect to our system, providing continuous monitoring. The system comprises a camera module for facial image capture, a processing unit for facial detection, and a mechanism to unlock the door for authorized individuals.

1.2 Problem Definition:

CCTV systems, integral to modern security, can encounter various issues that may affect their performance. Common problems include hardware malfunctions, such as camera freezes or unresponsiveness, often resolved by rebooting the system or updating firmware. Incompatibility issues with outdated firmware can lead to communication errors and decreased performance, necessitating firmware upgrades. Video loss, corrupted footage, and recording failures are also prevalent, sometimes requiring data recovery tools for lost footage. Image quality issues may arise from camera errors or environmental factors, which can be mitigated by regular maintenance and appropriate camera placement. Connectivity issues, another frequent challenge, can be addressed by checking network configurations and ensuring stable internet connections. It's crucial to regularly update security software on computers running CCTV systems to prevent operational disruptions. Additionally, safeguarding against security vulnerabilities is essential; using encrypted and redundant systems can help protect against unauthorized access. Understanding and promptly addressing these common problems can significantly enhance the reliability and effectiveness of CCTV systems in security applications.

Chapter 2

LITERATURE REVIEW

1. **TITLE**: Ranking Security of IoT-Based Smart Home Consumer Devices

YEAR: 2022

AUTHORS: Naba M. Allifah and Imran A. Zualkernan

ABSTRACT: This study introduces a methodology to rank the security of IoT-based smart home consumer devices, focusing on their vulnerabilities from an IoT perspective. Utilizing Analytic Hierarchy Process (AHP), the research ranks common home consumer devices based on their relative security. Network security emerges as the primary driver of smart home device security.

2. **TITLE**: Smart Door System Using Facial and Fingerprint Recognition for Access Control

YEAR: 2022

AUTHORS: Zhang et al.

ABSTRACT: Zhang et al. propose a smart door system employing facial and fingerprint recognition for access control. The system, comprising a camera module, fingerprint sensor, microcontroller unit, and locking mechanism, achieves high accuracy and security.

3. **TITLE**: Smart Door-Locking System Using Facial and Voice Recognition for Access Control

YEAR: 2022

AUTHORS: Chen et al.

ABSTRACT: Chen et al. present a smart door-locking system utilizing facial and voice recognition for access control. The system, equipped with a camera module, microphone, microcontroller unit, and locking mechanism, achieves high accuracy and convenience.

4. **TITLE**: A Comprehensive Survey of Security Issues of Smart Home System: ‘‘Spear’’ and ‘‘Shields,’’ Theory and Practice

YEAR: 2022

AUTHORS: Jian Yang and Liu Sun

ABSTRACT: In this comprehensive survey, the authors scrutinize the security issues surrounding smart home systems (SHSs). They analyze existing definitions, cyber-attack methods ('spears'), countermeasures ('shields'), security frameworks, evaluation technologies, and practical research in SHSs. Future research avenues include unification of architecture, resource limitation, fragmentation, and code and firmware security.

5. **TITLE**: Performance Evaluation of Facial Recognition Algorithms for Access Control Applications

YEAR: 2020

AUTHORS: Jain and Ross

ABSTRACT: Jain and Ross evaluate the accuracy and speed of various facial recognition algorithms for access control applications. Their study concludes that the Eigenface algorithm exhibits the best performance among the evaluated algorithms.

6. **TITLE**: Deep Convolutional Neural Network for Facial Recognition in Access Control Systems

YEAR: 2023

AUTHORS: Li et al.

ABSTRACT: Li et al. propose a facial recognition system based on a deep convolutional neural network, achieving high accuracy in recognizing faces for access control purposes.

7. **TITLE**: Systematic Survey on Smart Home Safety and Security Systems Using the Arduino Platform

YEAR: 2020

AUTHORS: Qusay I. Sarhan

ABSTRACT: This systematic survey delves into smart home safety and security systems employing the Arduino platform. Extracting insights from 63 research papers, the study analyzes the state-of-the-art applications, architectures, enabling technologies, components, and challenges in these systems. It emphasizes the role of smart home systems in ensuring safety and security.

Chapter 3

THEORETICAL BACKGROUND

3.1 Implementation Environment

AI surveillance cameras can be deployed in various environments and scenarios to enhance security, safety, and efficiency. Here are some common places where AI surveillance cameras can be used:

Home Security: AI surveillance cameras can monitor homes, apartments, and residential areas to detect intruders, monitor entry points, and alert homeowners to potential threats.

Commercial Buildings: Offices, retail stores, warehouses, and other commercial buildings can benefit from AI surveillance cameras to prevent theft, monitor employee activity, and ensure workplace safety.

Public Spaces: AI surveillance cameras can be installed in public spaces such as parks, streets, and transportation hubs to enhance public safety, monitor crowd behavior, and detect suspicious activities.

Schools and Educational Institutions: Educational facilities can use AI surveillance cameras to monitor campus grounds, secure entry points, and ensure the safety of students and staff members.

Healthcare Facilities: Hospitals, clinics, and healthcare facilities can deploy AI surveillance cameras to monitor patient areas, protect medical equipment, and enhance security in sensitive areas such as emergency rooms and operating theaters.

Transportation Systems: AI surveillance cameras can be integrated into transportation systems, including airports, train stations, and bus terminals, to enhance passenger safety, monitor luggage handling, and detect security threats.

Industrial Sites: Factories, manufacturing plants, and industrial sites can utilize AI surveillance cameras to monitor production lines, ensure workplace safety, and detect equipment malfunctions or anomalies.

Smart Cities: AI surveillance cameras can play a role in smart city initiatives by monitoring traffic flow, detecting accidents or congestion, and providing valuable data for urban planning and management.

Critical Infrastructure: AI surveillance cameras can be deployed in critical infrastructure facilities such as power plants, water treatment plants, and telecommunications centers to enhance security and prevent unauthorized access.

Border Security: Border checkpoints, immigration centers, and border control agencies can use AI surveillance cameras to monitor border areas, detect illegal crossings, and enhance border security.

3.2 System Architecture

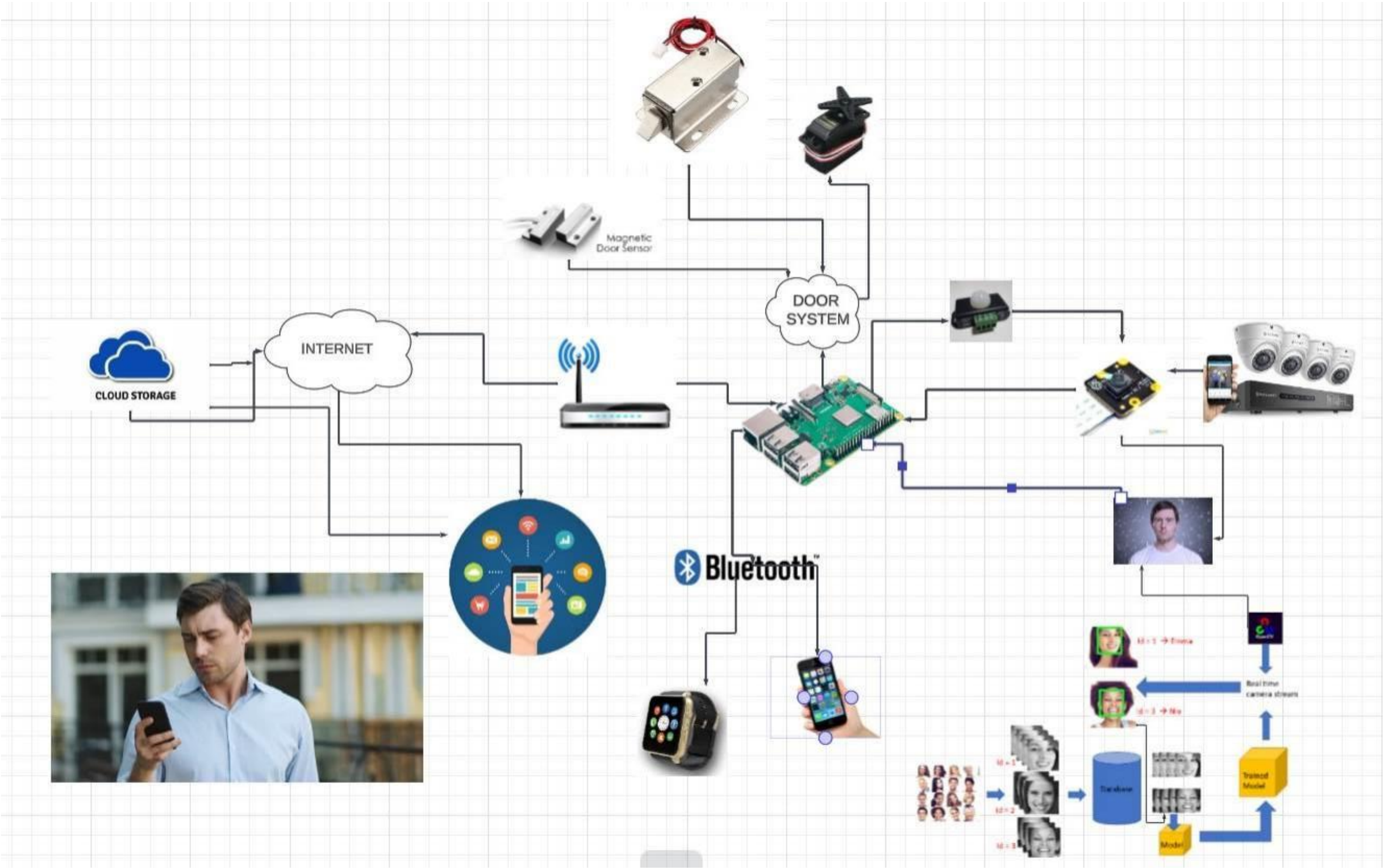


Fig no 3.1 System Architecture

This advanced security system combines cutting-edge AI and cloud storage for complete peace of mind. An AI-powered camera with facial recognition captures clear footage and identifies individuals. Magnetic door sensors and infrared sensors detect intrusion attempts, even in low-light conditions. The ESP32 microcontroller and Raspberry Pi process data, while motion detection software triggers recordings and alerts. Facial recognition software identifies authorized personnel. All data, including video, sensor readings, and recognition results, are stored securely in a local MySQL database and backed up to an AWS S3 cloud storage bucket for remote access. This system offers enhanced security, improved access control through facial recognition, and remote monitoring capabilities. Remember, strong security protocols and clear data policies are essential for a well-rounded security solution.

3.3 Existing Systems

Residential security now includes home surveillance and CCTV (Closed-Circuit Television) systems. Here's an overview of the existing system:

Analog CCTV Systems: Traditional CCTV systems use analog cameras that send signals to a specific set of monitors through a direct cable connection. Although these systems are cost-effective, they lack advanced features and lower image quality.

Digital CCTV Systems: These systems use digital cameras that provide high-resolution images and videos. They can be networked and accessed remotely, offering greater flexibility and scalability.

IP Cameras: Internet Protocol cameras are digital cameras that send and receive data via the internet. They offer high-quality footage and can be accessed remotely from any location with internet access.

DVR/NVR Systems: Digital Video Recorders (DVR) and Network Video Recorders (NVR) are used to record and store footage from analog and IP cameras respectively. They offer features like motion detection, scheduled recording, and real-time notifications.

Wireless CCTV Systems: These systems use wireless cameras, reducing the need for complex wiring. However, they are susceptible to signal interference.

Integrated Home Security Systems: These systems integrate CCTV cameras with other security measures like alarms, door sensors, and smart locks. They often come with a centralized control panel and offer remote access through mobile apps.

Cloud-Based Systems: Some modern systems offer cloud storage options for CCTV footage.

This eliminates the need for physical storage devices and allows for easy retrieval of data.

Smart Home Integration: Many CCTV systems can be integrated with smart home systems, allowing users to control their security systems along with other smart devices through a single interface.

In conclusion, the existing systems for home surveillance and CCTV offer a range of options to cater to different needs and budgets. However, they also present challenges such as privacy concerns and the need for regular maintenance. These systems continue to evolve as technology advances, providing enhanced security solutions for homes.

3.4 Proposed Methodology
3.4.1 Database Design

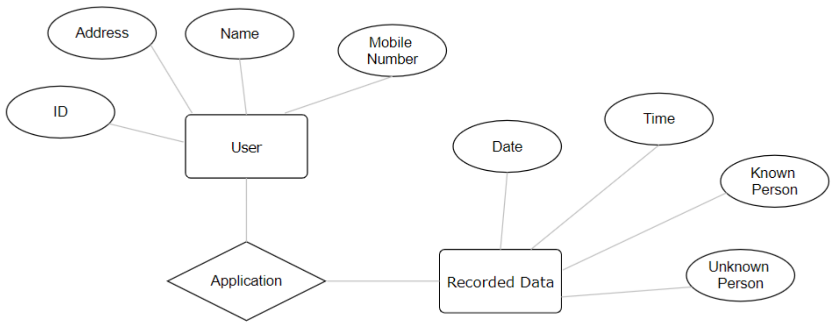


Fig no 3.2 Entity Relationship

Three primary tasks:

1. Train a new face recognition model.
2. Validate the model.
3. Test the model.

When training, your face recognizer will need to open and read many image files. It’ll also need to know who appears in each one. To accomplish this, you’ll set up a directory structure to give your program information about the data. Specifically, your project directory will contain three data directories:

1. training/
2. validation/
3. output/

You can put images directly into validation/. For training/, you should have images separated by subject into directories with the subject’s name.Setting your training directory up this way will allow you to give your face recognizer the information that it needs to associate a label—the person pictured—with the underlying image data.

3.4.2 Input Design (UI)

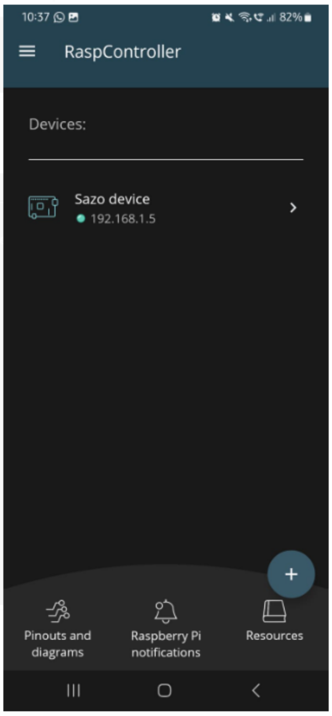


Fig no 3.3 Home Page

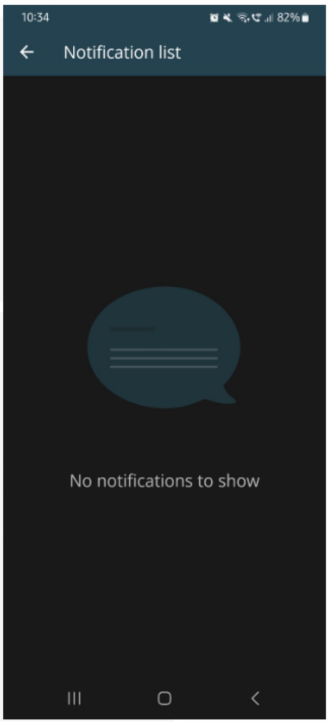


Fig no 3.4 Notification Page

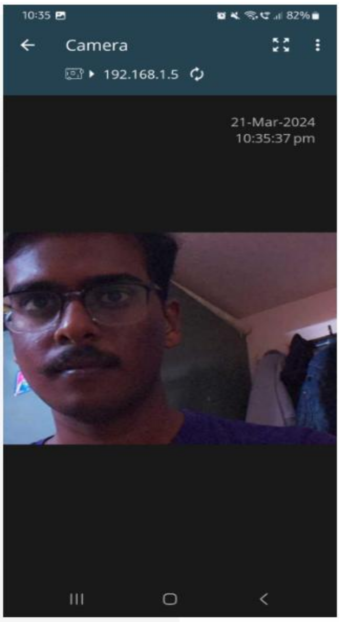


Fig no 3.5 Streaming Page

3.4.3 Module Design

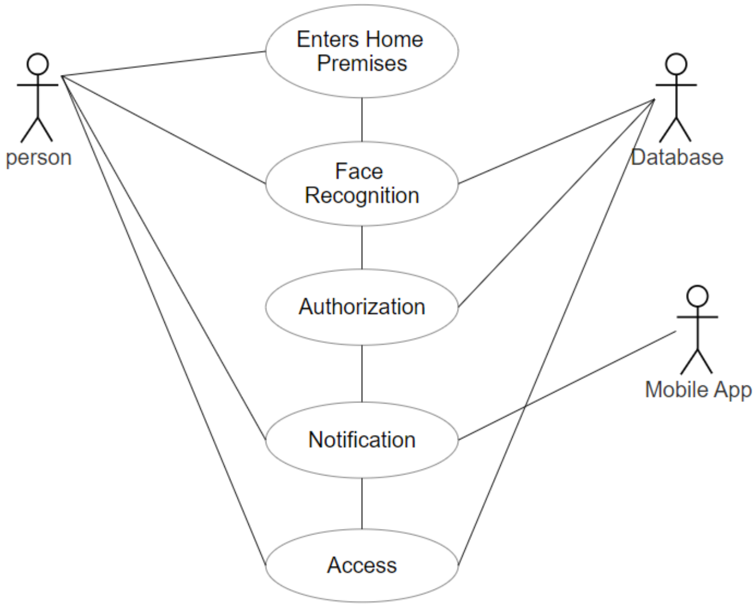


Fig no 3.6 Use Case Diagram

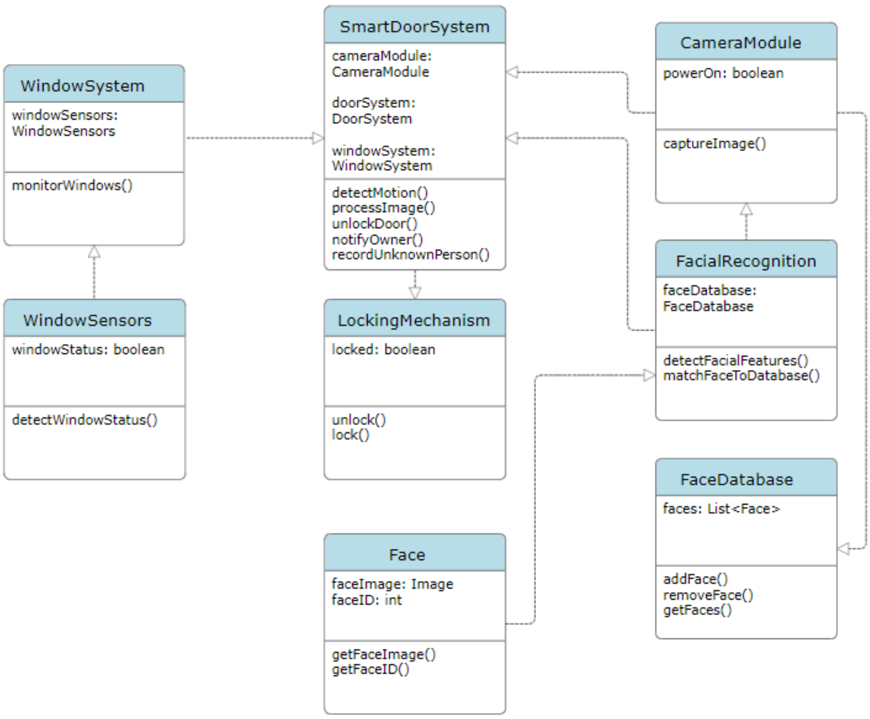


Fig no 3.7 Class Diagram

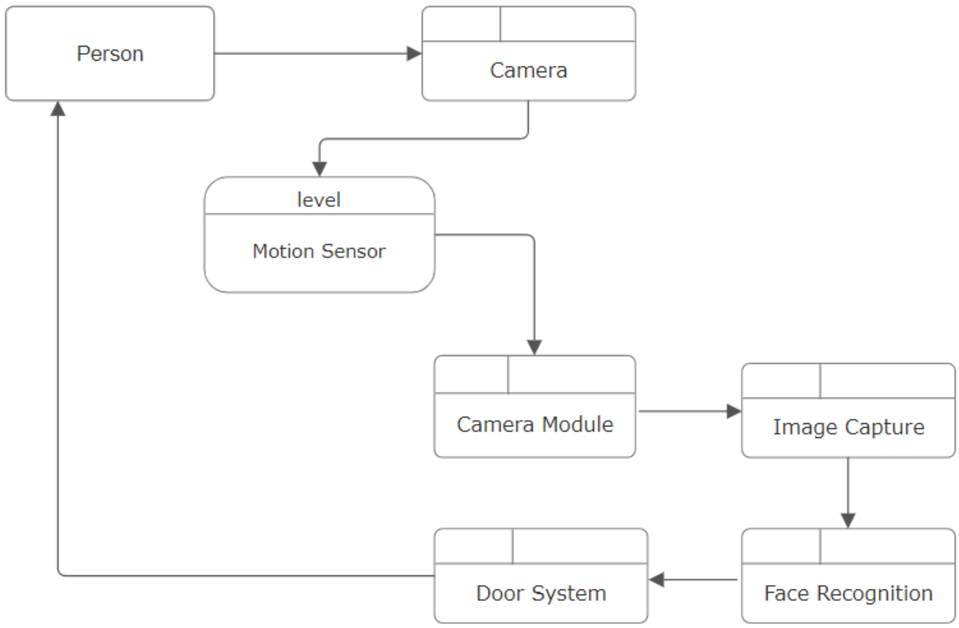


Fig no 3.8 Data Flow Diagram

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Hardware Components

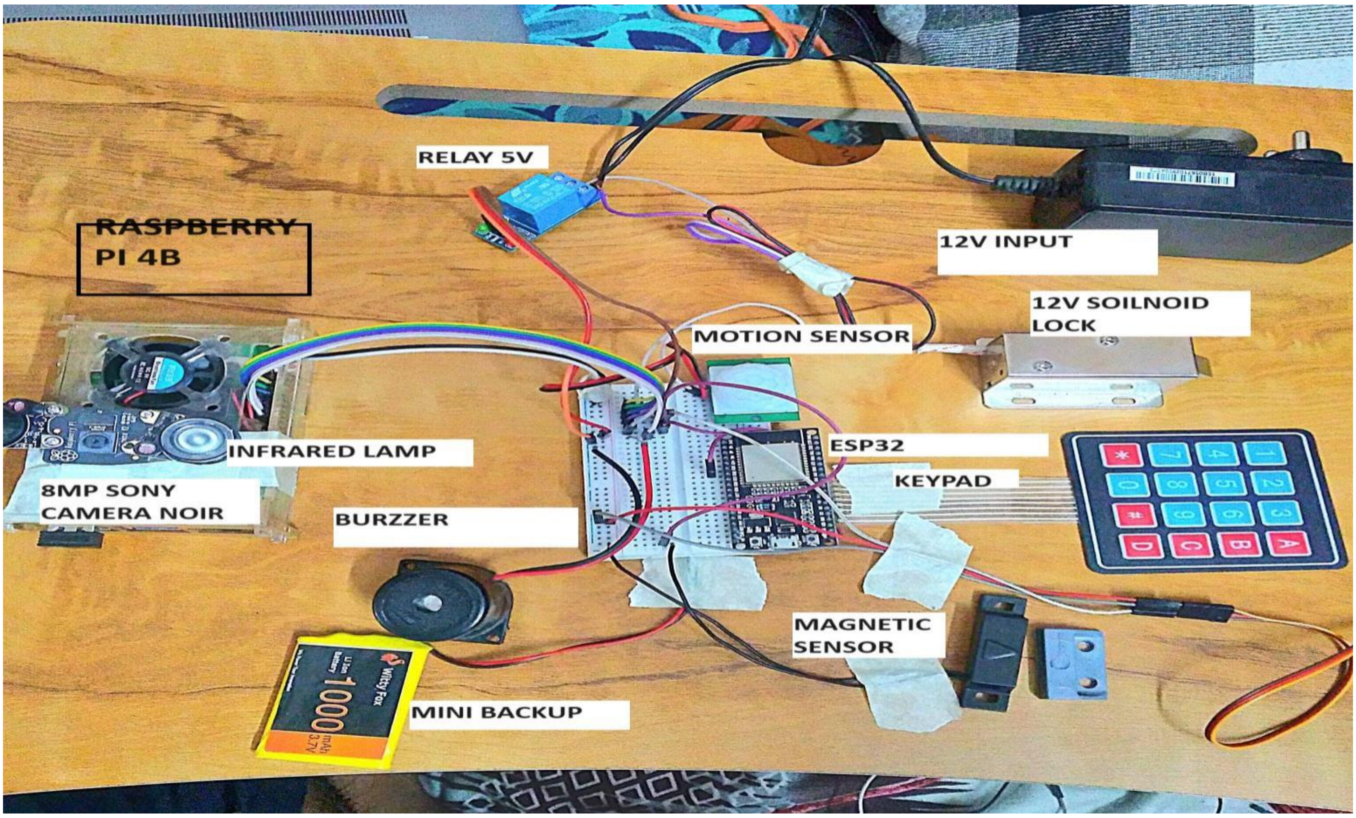


Fig no 4.1 Hardware Components

The Raspberry Pi 4 Model B is a versatile single-board computer that offers significant improvements over its predecessors. Here are some key features and details about the Raspberry Pi 4:

1. Performance and Specifications:

The Raspberry Pi 4 Model B features a 64-bit quad-core processor (Broadcom BCM2711B0) and a 4K-capable Broad-com VideoCore VI video processor.

It comes with dual-display support at resolutions up to 4K via a pair of micro HDMI ports. The board is 8GB of LPDDR4 SDRAM1 .

2. Desktop Experience:

For the first time, the Raspberry Pi 4 provides a complete desktop experience.

Whether you're editing documents, browsing the web, working with spreadsheets, or creating presentations, the experience is smooth and familiar.

3. Silent and Energy-Efficient:

The Raspberry Pi 4 runs fan less and consumes far less power than traditional computers.

It's an excellent choice for projects where silent operation and energy efficiency are essential.

4. Fast Networking:

The board includes Gigabit Ethernet for high-speed wired networking.

It also supports onboard wireless networking and Bluetooth.

5. USB 3.0 Ports:

Along with two USB 2.0 ports, the Raspberry Pi 4 has two USB 3.0 ports for faster data transfer.

6. Compatibility:

The Raspberry Pi 4 is designed to be backward-compatible with older models. Projects created on a Raspberry Pi 4 will work on older models as well.

7. Getting Started:

To set up your Raspberry Pi 4, you'll need a 15W USB-C power supply, a micro SD card with Raspberry Pi OS installed, a keyboard, a mouse, and cables to connect to one or two displays via the micro HDMI ports.

4.2 Debian bookworm /bullseye OS latest:

Debian 11, code named "Bullseye," was released as the stable version in August 2021. It brought numerous updates, improvements, and new features across various aspects of the operating system, including:

Updated software packages: Bullseye includes updated versions of software packages, libraries, and utilities, providing users with the latest features and security enhancements.

Secure Boot support: Debian 11 added support for Secure Boot, allowing users to boot the operating system on UEFI-based systems with Secure Boot enabled.

Improved accessibility: Bullseye focused on improving accessibility features, making the operating system more usable for users with disabilities.

Updated desktop environments: Bullseye offers multiple desktop environment options, including GNOME, KDE Plasma, Xfce, LXQt, and others, each updated to their latest versions.

Enhanced hardware support: The Bullseye release included updated drivers and kernel improvements, providing better support for a wide range of hardware configurations.

Updated installer: The Debian Installer was updated with improvements to the installation process, hardware detection, and usability enhancements.

Security updates: Bullseye includes security updates and patches to address vulnerabilities and ensure a secure computing environment.

4.3 Visual studio code:

Visual Studio Code (VS Code) is a free, standalone source-code editor developed by Microsoft. It supports many programming languages, including Python, Java, C++, and JavaScript.

VS Code offers a comprehensive suite of development tools, including debugging, task running, and version control capabilities. Its primary aim is to furnish developers with everything necessary for a streamlined code-build-debug cycle.

VS Code is a top pick for JavaScript and web developers, with extensions to support almost any programming language.

HTML, short for Hyper Text Markup Language, is the fundamental code utilized to organize the structure of a web page along with its content. It enables the structuring of content through various elements such as paragraphs, bulleted lists, images, and data tables.

CSS stands for Cascading Style Sheets. It's a computer language that's used to structure and lay out web pages. CSS is a key technology of the World Wide Web, along with HTML and JavaScript. Using this CSS, the front-end of the web page is built.

JavaScript is a text-based programming language used for creating interactive web pages. Alongside HTML and CSS, it forms the core technologies of the World Wide Web, and it's essential for building both client-side and server-side functionality.

4.4 Python3/pip3:

Python 3: Python is a versatile programming language known for its simplicity and readability. Python 3 is the latest version of Python, succeeding Python 2. It includes many new features and improvements over Python 2 and is the recommended version for new projects. Python 3 introduced syntax changes to improve consistency and remove redundancies, as well as performance enhancements.

pip3: pip is the package installer for Python, allowing you to easily install and manage Python packages from the Python Package Index (PyPI) and other repositories. pip3 is the version of pip specifically for Python 3. It simplifies the process of installing, upgrading, and uninstalling Python packages and their dependencies.

4.5 VNC sever:

VNC (Virtual Network Computing): VNC is a remote desktop sharing system that allows you to remotely access and control another computer's desktop environment over a network connection. It enables users to view and interact with the graphical desktop of a remote computer as if they were sitting in front of it.

VNC Server: The VNC server is the software component installed on the computer whose desktop you want to access remotely. It captures the graphical desktop output and transmits it over the network to VNC client software running on another computer. There are various implementations of VNC servers available, such as TightVNC, RealVNC, and TigerVNC, each with its own features and capabilities.

4.6 MySQL sever:

MySQL: MySQL is an open-source relational database management system (RDBMS) known for its speed, reliability, and ease of use. It is widely used in web development, data analytics, and other applications requiring structured data storage and retrieval.

MySQL Server: The MySQL server is the core component of the MySQL database system. It manages databases, tables, users, and permissions, and provides access to data through SQL (Structured Query Language) queries. The MySQL server listens for client connections over the network and handles requests to perform operations such as querying, inserting, updating, and deleting data.

Features: MySQL Server offers features such as ACID-compliant transactions, support for various storage engines (e.g., InnoDB, MyISAM), replication for high availability and scalability, stored procedures and triggers for advanced database logic, and robust security features to protect data integrity and confidentiality.

4.7 AWS:

Building a facial recognition app on AWS involves several steps that integrate various AWS services to create a robust and scalable application. The process begins with setting up an AWS S3 bucket to store images that will be used for facial recognition. Once the S3 bucket is in place, AWS Rekognition is employed to analyze the images and perform the facial recognition tasks. This service uses machine learning algorithms to compare faces in images with a database of known faces and identify matches.

Next, AWS Lambda functions are created to handle the processing logic. These functions are triggered by events, such as the uploading of a new image to the S3 bucket, and they use the Rekognition service to analyze the images. The results from Rekognition, which include the identification of faces and any matches found, are then stored in AWS DynamoDB, a NoSQL database service that provides fast and flexible data storage.

To make the facial recognition functionality accessible, an API is set up using AWS API Gateway. This service allows for the creation of RESTful APIs that can be called from a web or mobile application. The API acts as a front door to the Lambda functions, allowing them to be invoked with HTTP requests. The API Gateway is configured to handle these requests, invoke the appropriate Lambda function, and return the results to the caller.

Finally, the entire process is monitored and managed through the AWS Management Console, which provides tools for deploying, configuring, and monitoring the various AWS services involved in the application. This includes setting up permissions and security measures to ensure that the facial recognition app is secure and that only authorized users can access the sensitive data it processes.

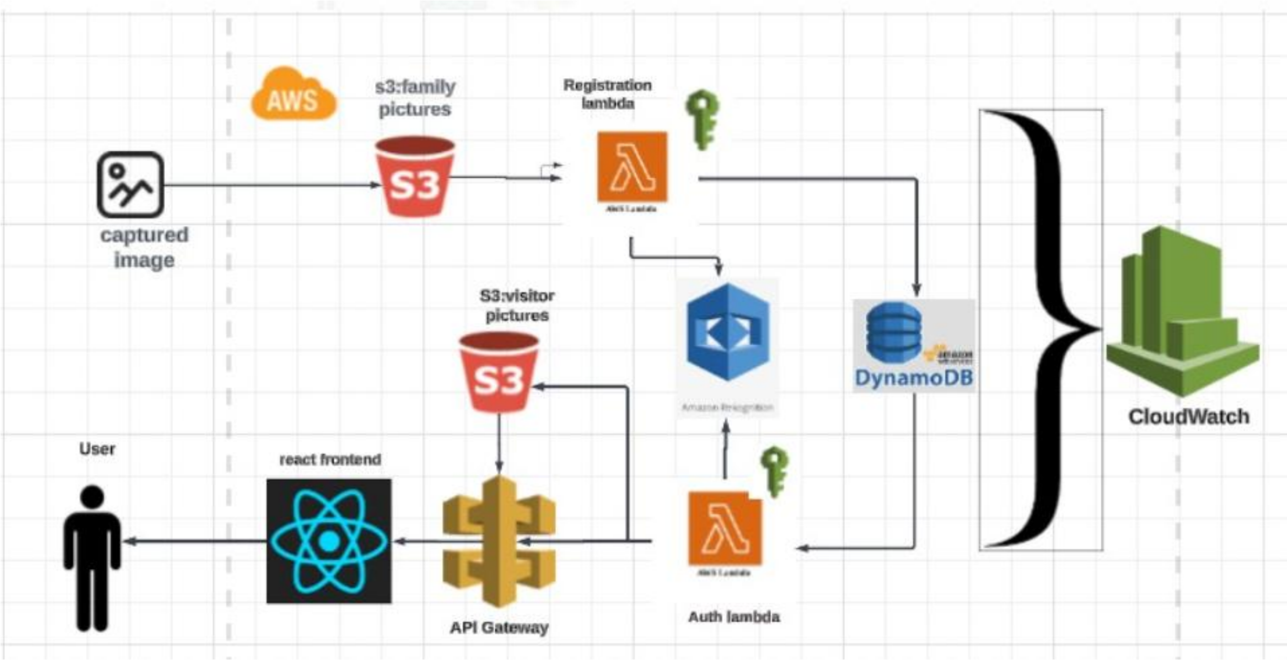


Fig no 4.2 AWS Architecture

CHAPTER 5

RESULTS & DISCUSSION

Face Recognition and Video Recording:

This script combines face recognition with video recording using OpenCV and the DeepFace library. It begins by setting up the video capture from a webcam and configuring frame dimensions. Haar cascade classifiers are loaded for face and body detection. The script also loads reference images of known persons from a specified directory. It then enters a continuous loop to capture frames from the video stream and detect faces and bodies using the loaded classifiers. When a face or body is detected, the script initiates video recording for a specified duration. Asynchronous face recognition is performed using threading to prevent blocking the main loop. If a match is found between a detected face and a reference image, it annotates the video feed accordingly. The script provides real-time feedback by displaying the video feed with annotations indicating whether a match is found or not. Users can terminate the script by pressing 'q'. Upon termination, it releases resources including the video capture object and closes OpenCV windows. This script enables real-time monitoring and recording when known persons are detected, making it suitable for applications such as surveillance or access control.

Complexity:

The complexity of this script is moderate, combining face detection, face recognition, and video recording functionalities. Face detection and recognition involve relatively complex algorithms, especially when using deep learning-based approaches like the DeepFace library. Threading is employed for asynchronous face recognition, adding some complexity to the script but improving overall performance by avoiding blocking the main loop during recognition. Video recording functionality is relatively straightforward and adds minimal complexity to the script.

Code Re-usability:

The script demonstrates moderate code re-usability. The face recognition and video recording functionalities can be reused in various applications requiring access control, attendance systems, or personalized user experiences. However, the script's re usability might be limited by its dependency on external libraries such as OpenCV and DeepFace, as well as specific hardware requirements for face recognition tasks (e.g., webcam or camera access, GPU acceleration for deep learning inference). With appropriate modularization and abstraction, the face recognition and video recording components of the script could be separated for easier reuse in different contexts or integrated into larger systems.

License Plate Detection:

This Python script leverages OpenCV and a pre-trained Haar cascade classifier to detect license plates within a live video stream or from a specified video file. The script begins by defining the path to the Haar cascade XML file specifically trained for detecting license plates. It then initializes a video capture object, setting the resolution of the captured video stream. Additionally, it specifies the minimum area threshold for a detected license plate and loads the cascade classifier for license plate detection. During execution, the script continuously captures frames from the video stream. Each frame is converted to grayscale to facilitate license plate detection. The Haar cascade classifier is then used to detect potential license plates within the frame. If a detected region exceeds the minimum area threshold, it is considered as a potential license plate region. For each detected license plate region, the script extracts the region of interest (ROI) containing the license plate. It generates a unique file name for the saved ROI and checks whether the ROI has already been saved to prevent duplicate entries. The script saves the ROI as an image in a specified directory. It also maintains a list of saved ROIs to ensure that each ROI is saved only once. Finally, the script displays the resulting video feed with annotated license plate regions. Users have the option to exit the script by pressing 'q'. Upon termination, the video capture object is released, and OpenCV windows are closed.

Complexity:

The complexity of this script primarily depends on the efficiency of the Haar cascade classifier for license plate detection. Haar cascades are relatively fast for object detection but might not be as accurate as deep learning-based approaches. The complexity of the license plate detection algorithm itself is relatively low, as it involves simple image processing operations such as grayscale conversion, object detection using Haar cascades, and region extraction.

Code Re-usability:

The script demonstrates moderate code re usability. The license plate detection functionality can be reused in various applications requiring license plate recognition, such as automated toll collection, parking management systems, or traffic surveillance. However, the script's re-usability might be limited by the specific use of the Haar cascade classifier for license plate detection. If higher accuracy or more robust detection is required, integrating more advanced techniques such as deep learning-based object detection models might be necessary.

YOLOv5 Object Detection:

This script implements object detection using YOLOv5, a state-of-the-art deep learning model renowned for its accuracy and speed in real-time object detection tasks. The script is highly versatile, supporting inference on various input sources, including images, videos, directories, URLs, webcams, and streams. Upon execution, the script parses command-line arguments to configure inference options and model configurations. Users can specify parameters such as model weights, input source, confidence threshold, NMS IoU threshold, and more. These options allow users to customize the inference process according to their requirements. The script loads the YOLOv5 model and initializes the inference pipeline. It iterates through the input source, performing object detection on each frame or image. Detected objects are annotated with bounding boxes and class labels. Users can visualize the detection results in real-time, save them to disk, or stream them depending on the specified options. Additionally, the script provides functionalities for saving detection results in text or CSV format. It also includes options for augmenting inference, hiding labels or confidences, and using half-precision inference for faster processing. Upon completion, the script prints detailed information about the inference speed and the number of labels saved. It ensures that all requirements are met before executing the model inference.

Complexity:

The complexity of this script is primarily determined by the YOLOv5 model, which employs a deep convolutional neural network for object detection. YOLOv5 is computationally intensive due to its deep architecture and the need for GPU acceleration for real-time performance. The script's complexity increases with the variety of input sources and the flexibility to configure various inference options such as confidence thresholds, NMS parameters, and input image sizes.

Code Re-usability:

The script exhibits high code re-usability owing to the versatility of YOLOv5 for object detection tasks. It can be reused in a wide range of applications requiring real-time object detection, such as surveillance systems, autonomous vehicles, and industrial automation. Users can easily modify the script to adapt to different datasets, model variants (e.g., YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x), or inference configurations based on their specific requirements.

AWS:

Amazon **CloudWatch** is a service used for monitoring and observing resources in real-time, built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers. CloudWatch provides users with data and actionable insights to monitor their respective applications, stimulate system-wide performance changes, and optimize resource utilization. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, providing its users with an aggregated view of AWS resources, applications, and services that run on AWS. The CloudWatch can also be used to detect anomalous behavior in the environments, set warnings and alarms, visualize logs and metrics side by side, take automated actions, and troubleshoot issues.

Amazon **Simple Storage Service (Amazon S3)** is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

AWS provides a powerful **face recognition service** that allows for accurate and efficient face comparison functions. This service utilizes advanced machine learning algorithms to compare faces in images and videos, identifying unique facial features and expressions with high precision. It can detect, analyze, and compare faces for a variety of user verification, people counting, and public safety applications. The AWS face comparison function is designed to be robust, supporting the processing of large volumes of images while maintaining quick response times and high accuracy rates. This makes it an invaluable tool for developers looking to integrate facial recognition capabilities into their applications or services.

5.1 System Testing

TESTCASE ID	TESTCASE/ ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	RESULT
1.	Selecting IP address, UserID, and password.	Display homepage	Displays home page	Pass
2	The server parses the required data & checks the online availability.	The server parses the file by database & connected message	The server parses the file by database & connected message	Pass
3.	The application should display information about status of security to the user.	All the required data is displayed to the user once logged in successfully	The required data is displayed to the user correctly	Pass
4.	The sensors collect the data from the door and the camera is ready to capture.	The collection of data is periodical	The server receives the data periodically from the Iot device	Pass
5.	Email should be sent if a person is detected or any breach the door or windows.	Gmail sent to individual's email	Gmail sent to individual's email	Pass
6.	Video backup is stored in the cloud and sent in mail,if person is unknown	The video is back up and sent to the user. logs are stored	The video is backup and sent to the user. Logs are stored	Pass
7.	The image processing algorithm detects the objects carried by the person & the vehicles in front of the home.	If the algorithm detects objects /number plate and store the data along with person data.	If the algorithm detects objects /number plate and store the data along with person data.	Pass

8.	Live stream the camera from the device.	Using IP address and user id the device allows to view live	Using IP address and user id the device allows to view live	pass
9.	Rapid Bluetooth device monitoring	Device connected successfully	Device connected successfully	pass
10.	Rapid magnetic sensor status monitoring	Working! The door or window is locked /opened.	Working! The door or window is locked /opened.	pass
11.	Aws s3 cloud /API/logs service continuous updating.	Data are sent and received in time.	Data are sent and received in time.	pass

Chapter 6

CONCLUSION AND FUTURE WORK

CONCLUSION

This system is an innovative solution that combines the latest recognition technology with advanced access control to improve safety at home. Makes sense to home. The system is interconnected and manages the doors and windows of the house. The camera module captures live video (if the camera is 30 frames per second, we can get 30 pictures to study) and divides the frame to analyze learning on the device. These images are saved to the database. The magnetic anti-theft magnetic device with Wi-Fi module inbuilt is fixed the windows and by granting access only to authorized persons, the system provides a safer environment and peace of mind for the homeowner. Biometrics may also be added for additional security.

The storage space is well managed. Whereas in old CCTV cameras run 24/7 so the storage should be most. In contrast, our device stores the image of the person who is detected by the camera module. The Sixfab 3G/4G& LTE Base HAT grants device (Raspberry Pi) interface bridge between mini PCIe cellular modems. This makes the device 24/7 online. In addition, alert functionality for break and enter attempts increases the overall security of the system.

The system goes beyond passive surveillance by incorporating a magnetic anti-theft device with an inbuilt Wi-Fi module to secure windows. Access is meticulously controlled, limited exclusively to authorized individuals, thereby fortifying the safety of the home environment and instilling a sense of peace for the homeowner. The integration of biometric technology, a potential addition, adds an extra layer of security and personalization to the access control system.

In addition to its primary security functions, the system incorporates an alert mechanism designed to notify homeowners in the event of break-in attempts, significantly augmenting the overall security infrastructure. This proactive feature not only serves as a deterrent but also ensures prompt response in critical situations, further reinforcing the system's efficacy.

Looking towards the future, the system envisions enhancements that elevate user interaction and friendliness. The integration of an artificial intelligence robot, coupled with integrated voice capabilities, enhances the system's interactivity. Moreover, the inclusion of speech recognition technology amplifies its effectiveness, presenting a user-friendly interface that aligns with evolving security requirements. This forward-looking approach positions the system as a dynamic and adaptive solution, not merely addressing current security needs but actively anticipating and preparing for future advancements in home security technology.

FUTURE WORK

In future iterations, several enhancements can elevate the capabilities of the smart home security system even further. One avenue of development lies in the utilization of artificial intelligence (AI) algorithms. Integrating AI into the system can enable more sophisticated analysis of captured images and video footage, allowing for more accurate detection of potential security threats and unauthorized access attempts. Additionally, AI-powered algorithms can facilitate the implementation of predictive analytics, enabling the system to anticipate and proactively address security risks before they escalate.

Another promising area for improvement is the integration of voice-controlled interfaces and speech recognition technology. By incorporating these features, homeowners can interact with the security system using natural language commands, enhancing user experience and convenience. Voice-controlled interfaces can enable hands-free operation, allowing users to easily manage and monitor their home security system while performing other tasks.

Furthermore, the system can benefit from the integration of advanced biometric authentication methods. In addition to facial recognition, incorporating technologies such as fingerprint or iris scanning can further enhance security by providing multiple layers of authentication. This multi-factor authentication approach strengthens access control measures, ensuring that only authorized individuals are granted entry to the premises.

Moreover, to enhance the system's connectivity and ensure continuous operation, leveraging cellular connectivity solutions such as the Sixfab 3G/4G & LTE Base HAT can be instrumental. By enabling the device to remain online 24/7, regardless of the availability of traditional internet connections, the system can provide real-time alerts and notifications to homeowners, enhancing situational awareness and responsiveness to security events.

Additionally, the integration of robotics technology can introduce new capabilities and functionalities to the smart home security system. For instance, deploying AI-powered robots equipped with cameras and sensors can enable autonomous patrolling of the premises, enhancing surveillance and detection capabilities. These robots can also serve as interactive interfaces, providing users with real-time updates and responding to commands and inquiries.

Overall, the future development of the smart home security system holds immense potential for enhancing safety, convenience, and peace of mind for homeowners. By leveraging cutting-edge technologies such as AI, voice recognition, biometrics, and robotics, the system can evolve into a highly sophisticated and proactive security solution, capable of adapting to emerging threats and providing comprehensive protection for residential properties.

APPENDICES

A1: SDG Goals

Goal 3: Good Health and Well-being - A secure home environment supports the well-being of individuals and families by reducing stress and anxiety related to safety concerns.

Goal 5: Gender Equality - Enhanced security at home can help promote gender equality by creating safer environments for women and girls.

Goal 7: Affordable and Clean Energy **-** Leveraging energy-efficient components and technologies in the system can contribute to achieving sustainable energy goals.

Goal 8: Decent Work and Economic Growth - The development and implementation of innovative security solutions create opportunities for employment and economic growth in the technology sector.

Goal 9: Industry, Innovation, and Infrastructure **-** The system contributes to this goal by leveraging innovative technologies such as facial recognition, AI, and cellular connectivity to enhance home security infrastructure.

Goal 11: Sustainable Cities and Communities **-** By providing advanced security measures for residential properties, the system contributes to creating safer and more resilient communities.

Goal 12: Responsible Consumption and Production **-** By optimizing resource utilization and minimizing waste, the system aligns with goals related to sustainable consumption and production.

Goal 16: Peace, Justice, and Strong Institutions **-** Enhancing home security helps promote peace and stability within communities by deterring crime and ensuring the safety of residents.

A2: Source Code

Email Notification with Face ID

```
import cv2
import time
import datetime
import os
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from deepface import DeepFace

# Initialize video capture
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# Load Haar cascades for face and body detection
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
body_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_fullbody.xml")

# Initialize detection variables
detection = False
detection_stopped_time = None
timer_started = False
SECONDS_TO_RECORD_AFTER_DETECTION = 5
# Initialize video writer
frame_size = (int(cap.get(3)), int(cap.get(4)))
fourcc = cv2.VideoWriter_fourcc(*"mp4v")
out = None

# Load reference images from a directory
reference_images_dir = r"E:\final project\known persons\naveen"
reference_images = {}  # Dictionary to store reference images

for filename in os.listdir(reference_images_dir):
    if filename.endswith(".jpg"):
        person_name = os.path.splitext(filename)[0]
        reference_img = cv2.imread(os.path.join(reference_images_dir, filename))
        reference_images[person_name] = reference_img
```

```
def check_face(frame):
    global reference_images

    for person_name, reference_img in reference_images.items():
        try:
            if DeepFace.verify(frame, reference_img.copy())['verified']:
                print(f"Match found for {person_name}!")
                return True
        except ValueError:
            pass

    return False

def send_email_with_attachment(image_path):
    sender_email = 'yourusername@gmail.com'
    sender_password = 'your_app_password'  # Use the App Password generated earlier
    recipient_email = 'recipient@example.com'

    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = recipient_email
    msg['Subject'] = 'Person Detected: Image Attachment'

    body = 'A known person was detected. Attached is the captured image.'
    msg.attach(MIMEText(body, 'plain'))

    img_data = open(image_path, 'rb').read()
    image = MIMEImage(img_data, name=os.path.basename(image_path))
    msg.attach(image)

    try:
        with smtplib.SMTP('smtp.gmail.com', 587) as server:
            server.starttls()
            server.login(sender_email, sender_password)
            server.sendmail(sender_email, recipient_email, msg.as_string())
            print('Email sent successfully!')
    except Exception as e:
        print(f'Error sending email: {e}')

while True:
    _, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    bodies = body_cascade.detectMultiScale(gray, 1.3, 5)
```

```
    if len(faces) + len(bodies) > 0:
        if detection:
            timer_started = False
        else:
            detection = True
            current_time = datetime.datetime.now().strftime("%d-%m-%Y-%H-%M-%S")
            out = cv2.VideoWriter(r"C:\Users\navee\final project\recorded 
videos\{}.mp4".format(current_time),
                                  fourcc, 20, frame_size)

    if detection:
        out.write(frame)

        if not timer_started:
            detection_stopped_time = time.time()
            timer_started = True

        elapsed_time = time.time() - detection_stopped_time
        if elapsed_time > SECONDS_TO_RECORD_AFTER_DETECTION:
            detection = False
            out.release()
            send_email_with_attachment(r"C:\Users\navee\final project\recorded 
videos\{}.mp4".format(current_time))

cap.release()
cv2.destroyAllWindows()
```

Face Recognition, Video Recording, and License Plate Detection Code:

```
import threading
import cv2
from deepface import DeepFace
import os
import time
import datetime
```

```
known_persons_dir = r"E:\final project\known persons"
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
counter = 0

def load_reference_images():
    reference_images = {}
    for person_folder in os.listdir(known_persons_dir):
        person_folder_path = os.path.join(known_persons_dir, person_folder)
        if os.path.isdir(person_folder_path):
            for filename in os.listdir(person_folder_path):
                if filename.endswith(".jpg"):
                    person_name = person_folder
                    reference_img = cv2.imread(os.path.join(person_folder_path, filename))
                    reference_images[person_name] = reference_img
    return reference_images

def check_face(frame, reference_images):
    match_folder = None
    for person_name, reference_img in reference_images.items():
        try:
            if DeepFace.verify(frame, reference_img.copy())['verified']:
                match_folder = person_name
                break
        except ValueError:
            pass
    return match_folder

def face_recognition():
    global counter

    while True:
        ret, frame = cap.read()

        if ret:
            if counter % 38 == 8:
                threading.Thread(target=check_face, args=(frame.copy(), reference_images)).start()
            counter += 1

            match_folder = check_face(frame, reference_images)
            if match_folder:
                cv2.putText(frame, f"Match found for {match_folder}!", (20, 450),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            else:
```

```
            cv2.putText(frame, "NO MATCH!", (20, 450), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)

         cv2.imshow("Face Recognition", frame)

      key = cv2.waitKey(1)
      if key == ord("q"):
         break
   cv2.destroyAllWindows()

def video_recording():
   face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
   body_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_fullbody.xml")

   detection = False
   detection_stopped_time = None
   timer_started = False
   SECONDS_TO_RECORD_AFTER_DETECTION = 10

   frame_size = (int(cap.get(3)), int(cap.get(4)))
   fourcc = cv2.VideoWriter_fourcc(*"mp4v")

   while True:
      _, frame = cap.read()

      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
      faces = face_cascade.detectMultiScale(gray, 1.3, 5)
      bodies = body_cascade.detectMultiScale(gray, 1.3, 5)

      if len(faces) + len(bodies) > 0:
         if detection:
            timer_started = False
         else:
            detection = True
            current_time = datetime.datetime.now().strftime("%d-%m-%Y-%H-%M-%S")
            out = cv2.VideoWriter(r"C:\Users\navee\final project\recorded
videos\{}.mp4".format(current_time),
                           fourcc, 20, frame_size)
            print("Started Recording!")
      elif detection:
         if timer_started:
            if time.time() - detection_stopped_time >=
SECONDS_TO_RECORD_AFTER_DETECTION:
               detection = False
               timer_started = False
```

```
                out.release()
                print('Stop Recording!')
          else:
              timer_started = True
              detection_stopped_time = time.time()

        if detection:
            out.write(frame)

        key = cv2.waitKey(1)
        if key == ord('q'):
            break

    out.release()

# Load reference images initially
reference_images = load_reference_images()

# Start the threads
threading.Thread(target=face_recognition).start()
threading.Thread(target=video_recording).start()
```

Object Detection Code:

Detect.py

```
import argparse
import csv
import os
import platform
import sys
from pathlib import Path

import torch

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
```

```
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from ultralytics.utils.plotting import Annotator, colors, save_one_box

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadScreenshots,
LoadStreams
from utils.general import (
    LOGGER,
    Profile,
    check_file,
    check_img_size,
    check_imshow,
    check_requirements,
    colorstr,
    cv2,
    increment_path,
    non_max_suppression,
    print_args,
    scale_boxes,
    strip_optimizer,
    xyxy2xywh,
)
from utils.torch_utils import select_device, smart_inference_mode


@smart_inference_mode()
def run(
    weights=ROOT / "yolov5s.pt",  # model path or triton URL
    source=ROOT / "data/images",  # file/dir/URL/glob/screen/0(webcam)
    data=ROOT / "data/coco128.yaml",  # dataset.yaml path
    imgsz=(640, 640),  # inference size (height, width)
    conf_thres=0.25,  # confidence threshold
    iou_thres=0.45,  # NMS IOU threshold
    max_det=1000,  # maximum detections per image
    device="",  # cuda device, i.e. 0 or 0,1,2,3 or cpu
    view_img=False,  # show results
    save_txt=False,  # save results to *.txt
    save_csv=False,  # save results in CSV format
    save_conf=False,  # save confidences in --save-txt labels
    save_crop=False,  # save cropped prediction boxes
    nosave=False,  # do not save images/videos
    classes=None,  # filter by class: --class 0, or --class 0 2 3
    agnostic_nms=False,  # class-agnostic NMS
```

```python
    augment=False,  # augmented inference
    visualize=False,  # visualize features
    update=False,  # update all models
    project=ROOT / "runs/detect",  # save results to project/name
    name="exp",  # save results to project/name
    exist_ok=False,  # existing project/name ok, do not increment
    line_thickness=3,  # bounding box thickness (pixels)
    hide_labels=False,  # hide labels
    hide_conf=False,  # hide confidences
    half=False,  # use FP16 half-precision inference
    dnn=False,  # use OpenCV DNN for ONNX inference
    vid_stride=1,  # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith(".txt")  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(("rtsp://", "rtmp://", "http://", "https://"))
    webcam = source.isnumeric() or source.endswith(".streams") or (is_url and not is_file)
    screenshot = source.lower().startswith("screen")
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / "labels" if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    bs = 1  # batch_size
    if webcam:
        view_img = check_imshow(warn=True)
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
        bs = len(dataset)
    elif screenshot:
        dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
```

```
    seen, windows, dt = 0, [], (Profile(device=device), Profile(device=device), Profile(device=device))
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
            im /= 255  # 0 - 255 to 0.0 - 1.0
            if len(im.shape) == 3:
                im = im[None]  # expand for batch dim
            if model.xml and im.shape[0] > 1:
                ims = torch.chunk(im, im.shape[0], 0)

        # Inference
        with dt[1]:
            visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
            if model.xml and im.shape[0] > 1:
                pred = None
                for image in ims:
                    if pred is None:
                        pred = model(image, augment=augment, visualize=visualize).unsqueeze(0)
                    else:
                        pred = torch.cat((pred, model(image, augment=augment,
visualize=visualize).unsqueeze(0)), dim=0)
                pred = [pred, None]
            else:
                pred = model(im, augment=augment, visualize=visualize)
        # NMS
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)

        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

        # Define the path for the CSV file
        csv_path = save_dir / "predictions.csv"

        # Create or append to the CSV file
        def write_to_csv(image_name, prediction, confidence):
            """Writes prediction data for an image to a CSV file, appending if the file exists."""
            data = {"Image Name": image_name, "Prediction": prediction, "Confidence": confidence}
            with open(csv_path, mode="a", newline="") as f:
                writer = csv.DictWriter(f, fieldnames=data.keys())
                if not csv_path.is_file():
                    writer.writeheader()
                writer.writerow(data)
```

```
        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f"{i}: "
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, "frame", 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / "labels" / p.stem) + ("" if dataset.mode == "image" else f"_{frame}")  #
im.txt
            s += "%gx%g " % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness, example=str(names))
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, 5].unique():
                    n = (det[:, 5] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    c = int(cls)  # integer class
                    label = names[c] if hide_conf else f"{names[c]}"
                    confidence = float(conf)
                    confidence_str = f"{confidence:.2f}"

                    if save_csv:
                        write_to_csv(p.name, label, confidence_str)

                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  # normalized
xywh
                        line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
                        with open(f"{txt_path}.txt", "a") as f:
                            f.write(("%g " * len(line)).rstrip() % line + "\n")

                    if save_img or save_crop or view_img:  # Add bbox to image
                        c = int(cls)  # integer class
                        label = None if hide_labels else (names[c] if hide_conf else f"{names[c]} {conf:.2f}")
```

```
                        annotator.box_label(xyxy, label, color=colors(c, True))
                    if save_crop:
                        save_one_box(xyxy, imc, file=save_dir / "crops" / names[c] / f"{p.stem}.jpg",
BGR=True)

            # Stream results
            im0 = annotator.result()
            if view_img:
                if platform.system() == "Linux" and p not in windows:
                    windows.append(p)
                    cv2.namedWindow(str(p), cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)  #
allow window resize (Linux)
                    cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == "image":
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                        vid_path[i] = save_path
                        if isinstance(vid_writer[i], cv2.VideoWriter):
                            vid_writer[i].release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path = str(Path(save_path).with_suffix(".mp4"))  # force *.mp4 suffix on results
videos
                        vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*"mp4v"), fps, (w,
h))
                    vid_writer[i].write(im0)

        # Print time (inference-only)
        LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt * 1E3:.1f}ms")

    # Print results
    t = tuple(x.t / seen * 1e3 for x in dt)  # speeds per image
    LOGGER.info(f"Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at shape {(1,
3, *imgsz)}" % t)
    if save_txt or save_img:
```

```
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt else
""
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    """Parses command-line arguments for YOLOv5 detection, setting inference options and model
configurations."""
    parser = argparse.ArgumentParser()
    parser.add_argument("--weights", nargs="+", type=str, default=ROOT / "yolov5s.pt", help="model
path or triton URL")
    parser.add_argument("--source", type=str, default=ROOT / "data/images",
help="file/dir/URL/glob/screen/0(webcam)")
    parser.add_argument("--data", type=str, default=ROOT / "data/coco128.yaml", help="(optional)
dataset.yaml path")
    parser.add_argument("--imgsz", "--img", "--img-size", nargs="+", type=int, default=[640],
help="inference size h,w")
    parser.add_argument("--conf-thres", type=float, default=0.25, help="confidence threshold")
    parser.add_argument("--iou-thres", type=float, default=0.45, help="NMS IoU threshold")
    parser.add_argument("--max-det", type=int, default=1000, help="maximum detections per image")
    parser.add_argument("--device", default="", help="cuda device, i.e. 0 or 0,1,2,3 or cpu")
    parser.add_argument("--view-img", action="store_true", help="show results")
    parser.add_argument("--save-txt", action="store_true", help="save results to *.txt")
    parser.add_argument("--save-csv", action="store_true", help="save results in CSV format")
    parser.add_argument("--save-conf", action="store_true", help="save confidences in --save-txt labels")
    parser.add_argument("--save-crop", action="store_true", help="save cropped prediction boxes")
    parser.add_argument("--nosave", action="store_true", help="do not save images/videos")
    parser.add_argument("--classes", nargs="+", type=int, help="filter by class: --classes 0, or --classes 0
2 3")
    parser.add_argument("--agnostic-nms", action="store_true", help="class-agnostic NMS")
    parser.add_argument("--augment", action="store_true", help="augmented inference")
    parser.add_argument("--visualize", action="store_true", help="visualize features")
    parser.add_argument("--update", action="store_true", help="update all models")
    parser.add_argument("--project", default=ROOT / "runs/detect", help="save results to project/name")
    parser.add_argument("--name", default="exp", help="save results to project/name")
    parser.add_argument("--exist-ok", action="store_true", help="existing project/name ok, do not
increment")
    parser.add_argument("--line-thickness", default=3, type=int, help="bounding box thickness (pixels)")
    parser.add_argument("--hide-labels", default=False, action="store_true", help="hide labels")
    parser.add_argument("--hide-conf", default=False, action="store_true", help="hide confidences")
    parser.add_argument("--half", action="store_true", help="use FP16 half-precision inference")
    parser.add_argument("--dnn", action="store_true", help="use OpenCV DNN for ONNX inference")
    parser.add_argument("--vid-stride", type=int, default=1, help="video frame-rate stride")
    opt = parser.parse_args()
```

```
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt


def main(opt):
    """Executes YOLOv5 model inference with given options, checking requirements before running the
model."""
    check_requirements(ROOT / "requirements.txt", exclude=("tensorboard", "thop"))
    run(**vars(opt))


if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

detect.py

```
import argparse
import csv
import os
import platform
import sys
from pathlib import Path

import torch

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from ultralytics.utils.plotting import Annotator, colors, save_one_box

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages,
LoadScreenshots, LoadStreams
from utils.general import (
    LOGGER,
    Profile,
    check_file,
    check_img_size,
```

```
    check_imshow,
    check_requirements,
    colorstr,
    cv2,
    increment_path,
    non_max_suppression,
    print_args,
    scale_boxes,
    strip_optimizer,
    xyxy2xywh,
)
from utils.torch_utils import select_device, smart_inference_mode


@smart_inference_mode()
def run(
    weights=ROOT / "yolov5s.pt",  # model path or triton URL
    source=ROOT / "data/images",  # file/dir/URL/glob/screen/0(webcam)
    data=ROOT / "data/coco128.yaml",  # dataset.yaml path
    imgsz=(640, 640),  # inference size (height, width)
    conf_thres=0.25,  # confidence threshold
    iou_thres=0.45,  # NMS IOU threshold
    max_det=1000,  # maximum detections per image
    device="",  # cuda device, i.e. 0 or 0,1,2,3 or cpu
    view_img=False,  # show results
    save_txt=False,  # save results to *.txt
    save_csv=False,  # save results in CSV format
    save_conf=False,  # save confidences in --save-txt labels
    save_crop=False,  # save cropped prediction boxes
    nosave=False,  # do not save images/videos
    classes=None,  # filter by class: --class 0, or --class 0 2 3
    agnostic_nms=False,  # class-agnostic NMS
    augment=False,  # augmented inference
    visualize=False,  # visualize features
    update=False,  # update all models
    project=ROOT / "runs/detect",  # save results to project/name
    name="exp",  # save results to project/name
    exist_ok=False,  # existing project/name ok, do not increment
    line_thickness=3,  # bounding box thickness (pixels)
    hide_labels=False,  # hide labels
    hide_conf=False,  # hide confidences
    half=False,  # use FP16 half-precision inference
    dnn=False,  # use OpenCV DNN for ONNX inference
```

```
    vid_stride=1,  # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith(".txt")  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(("rtsp://", "rtmp://", "http://", "https://"))
    webcam = source.isnumeric() or source.endswith(".streams") or (is_url and not is_file)
    screenshot = source.lower().startswith("screen")
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / "labels" if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    bs = 1  # batch_size
    if webcam:
        view_img = check_imshow(warn=True)
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
        bs = len(dataset)
    elif screenshot:
        dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
    seen, windows, dt = 0, [], (Profile(device=device), Profile(device=device),
Profile(device=device))
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
            im /= 255  # 0 - 255 to 0.0 - 1.0
```

```
        if len(im.shape) == 3:
            im = im[None]  # expand for batch dim
        if model.xml and im.shape[0] > 1:
            ims = torch.chunk(im, im.shape[0], 0)

    # Inference
    with dt[1]:
        visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
        if model.xml and im.shape[0] > 1:
            pred = None
            for image in ims:
                if pred is None:
                    pred = model(image, augment=augment, visualize=visualize).unsqueeze(0)
                else:
                    pred = torch.cat((pred, model(image, augment=augment, visualize=visualize).unsqueeze(0)), dim=0)
            pred = [pred, None]
        else:
            pred = model(im, augment=augment, visualize=visualize)
    # NMS
    with dt[2]:
        pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms, max_det=max_det)

    # Second-stage classifier (optional)
    # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

    # Define the path for the CSV file
    csv_path = save_dir / "predictions.csv"

    # Create or append to the CSV file
    def write_to_csv(image_name, prediction, confidence):
        """Writes prediction data for an image to a CSV file, appending if the file exists."""
        data = {"Image Name": image_name, "Prediction": prediction, "Confidence": confidence}
        with open(csv_path, mode="a", newline="") as f:
            writer = csv.DictWriter(f, fieldnames=data.keys())
            if not csv_path.is_file():
                writer.writeheader()
            writer.writerow(data)

    # Process predictions
    for i, det in enumerate(pred):  # per image
```

```
        seen += 1
        if webcam:  # batch_size >= 1
            p, im0, frame = path[i], im0s[i].copy(), dataset.count
            s += f"{i}: "
        else:
            p, im0, frame = path, im0s.copy(), getattr(dataset, "frame", 0)

        p = Path(p)  # to Path
        save_path = str(save_dir / p.name)  # im.jpg
        txt_path = str(save_dir / "labels" / p.stem) + ("" if dataset.mode == "image" else
f"_{frame}")  # im.txt
        s += "%gx%g " % im.shape[2:]  # print string
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
        imc = im0.copy() if save_crop else im0  # for save_crop
        annotator = Annotator(im0, line_width=line_thickness, example=str(names))
        if len(det):
            # Rescale boxes from img_size to im0 size
            det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

            # Print results
            for c in det[:, 5].unique():
                n = (det[:, 5] == c).sum()  # detections per class
                s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

            # Write results
            for *xyxy, conf, cls in reversed(det):
                c = int(cls)  # integer class
                label = names[c] if hide_conf else f"{names[c]}"
                confidence = float(conf)
                confidence_str = f"{confidence:.2f}"

                if save_csv:
                    write_to_csv(p.name, label, confidence_str)

                if save_txt:  # Write to file
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  #
normalized xywh
                    line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
                    with open(f"{txt_path}.txt", "a") as f:
                        f.write(("%g " * len(line)).rstrip() % line + "\n")

                if save_img or save_crop or view_img:  # Add bbox to image
                    c = int(cls)  # integer class
```

```
                    label = None if hide_labels else (names[c] if hide_conf else f"{names[c]}
{conf:.2f}")
                    annotator.box_label(xyxy, label, color=colors(c, True))
                if save_crop:
                    save_one_box(xyxy, imc, file=save_dir / "crops" / names[c] / f"{p.stem}.jpg",
BGR=True)

        # Stream results
        im0 = annotator.result()
        if view_img:
            if platform.system() == "Linux" and p not in windows:
                windows.append(p)
                cv2.namedWindow(str(p), cv2.WINDOW_NORMAL |
cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
                cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond

        # Save results (image with detections)
        if save_img:
            if dataset.mode == "image":
                cv2.imwrite(save_path, im0)
            else:  # 'video' or 'stream'
                if vid_path[i] != save_path:  # new video
                    vid_path[i] = save_path
                    if isinstance(vid_writer[i], cv2.VideoWriter):
                        vid_writer[i].release()  # release previous video writer
                    if vid_cap:  # video
                        fps = vid_cap.get(cv2.CAP_PROP_FPS)
                        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                        h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                    else:  # stream
                        fps, w, h = 30, im0.shape[1], im0.shape[0]
                    save_path = str(Path(save_path).with_suffix(".mp4"))  # force *.mp4 suffix on
results videos
                    vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*"mp4v"),
fps, (w, h))
                vid_writer[i].write(im0)

    # Print time (inference-only)
    LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt * 1E3:.1f}ms")

  # Print results
```

```
    t = tuple(x.t / seen * 1e3 for x in dt)  # speeds per image
    LOGGER.info(f"Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at
shape {(1, 3, *imgsz)}" % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if
save_txt else ""
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    """Parses command-line arguments for YOLOv5 detection, setting inference options and
model configurations."""
    parser = argparse.ArgumentParser()
    parser.add_argument("--weights", nargs="+", type=str, default=ROOT / "yolov5s.pt",
help="model path or triton URL")
    parser.add_argument("--source", type=str, default=ROOT / "data/images",
help="file/dir/URL/glob/screen/0(webcam)")
    parser.add_argument("--data", type=str, default=ROOT / "data/coco128.yaml",
help="(optional) dataset.yaml path")
    parser.add_argument("--imgsz", "--img", "--img-size", nargs="+", type=int, default=[640],
help="inference size h,w")
    parser.add_argument("--conf-thres", type=float, default=0.25, help="confidence threshold")
    parser.add_argument("--iou-thres", type=float, default=0.45, help="NMS IoU threshold")
    parser.add_argument("--max-det", type=int, default=1000, help="maximum detections per
image")
    parser.add_argument("--device", default="", help="cuda device, i.e. 0 or 0,1,2,3 or cpu")
    parser.add_argument("--view-img", action="store_true", help="show results")
    parser.add_argument("--save-txt", action="store_true", help="save results to *.txt")
    parser.add_argument("--save-csv", action="store_true", help="save results in CSV format")
    parser.add_argument("--save-conf", action="store_true", help="save confidences in --save-txt
labels")
    parser.add_argument("--save-crop", action="store_true", help="save cropped prediction
boxes")
    parser.add_argument("--nosave", action="store_true", help="do not save images/videos")
    parser.add_argument("--classes", nargs="+", type=int, help="filter by class: --classes 0, or --
classes 0 2 3")
    parser.add_argument("--agnostic-nms", action="store_true", help="class-agnostic NMS")
    parser.add_argument("--augment", action="store_true", help="augmented inference")
    parser.add_argument("--visualize", action="store_true", help="visualize features")
    parser.add_argument("--update", action="store_true", help="update all models")
```

```
    parser.add_argument("--project", default=ROOT / "runs/detect", help="save results to project/name")
    parser.add_argument("--name", default="exp", help="save results to project/name")
    parser.add_argument("--exist-ok", action="store_true", help="existing project/name ok, do not increment")
    parser.add_argument("--line-thickness", default=3, type=int, help="bounding box thickness (pixels)")
    parser.add_argument("--hide-labels", default=False, action="store_true", help="hide labels")
    parser.add_argument("--hide-conf", default=False, action="store_true", help="hide confidences")
    parser.add_argument("--half", action="store_true", help="use FP16 half-precision inference")
    parser.add_argument("--dnn", action="store_true", help="use OpenCV DNN for ONNX inference")
    parser.add_argument("--vid-stride", type=int, default=1, help="video frame-rate stride")
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt


def main(opt):
    """Executes YOLOv5 model inference with given options, checking requirements before running the model."""
    check_requirements(ROOT / "requirements.txt", exclude=("tensorboard", "thop"))
    run(**vars(opt))


if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

A3: Screenshots

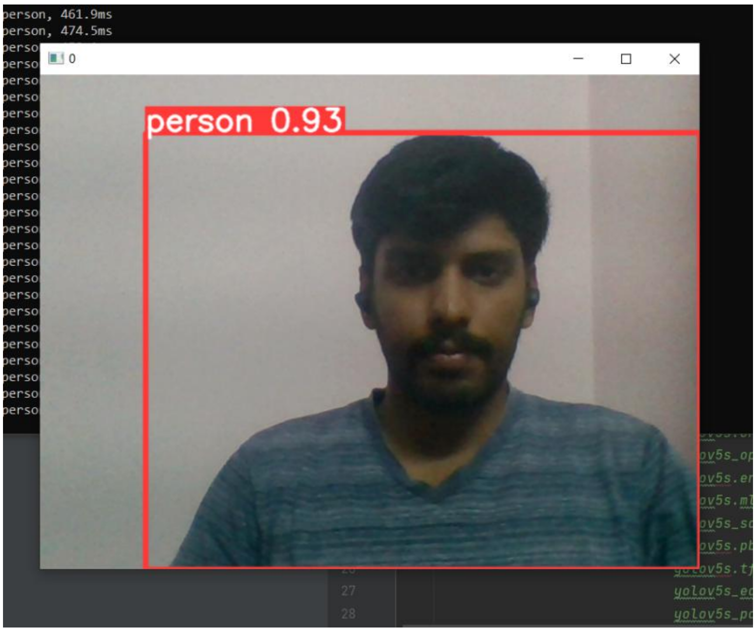


Fig no A3.1 Object Detection & Human Detection

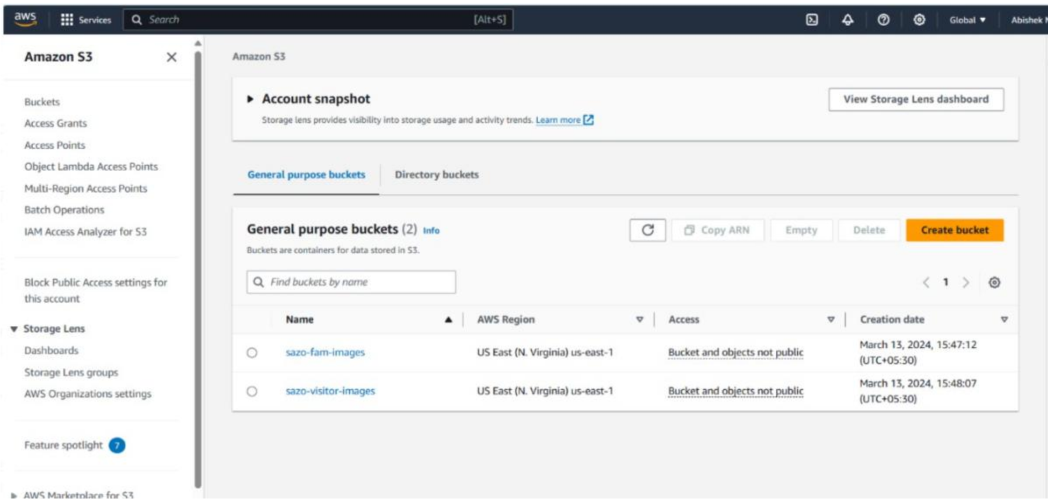


Fig no A3.2 AWS S3 Cloud Storage

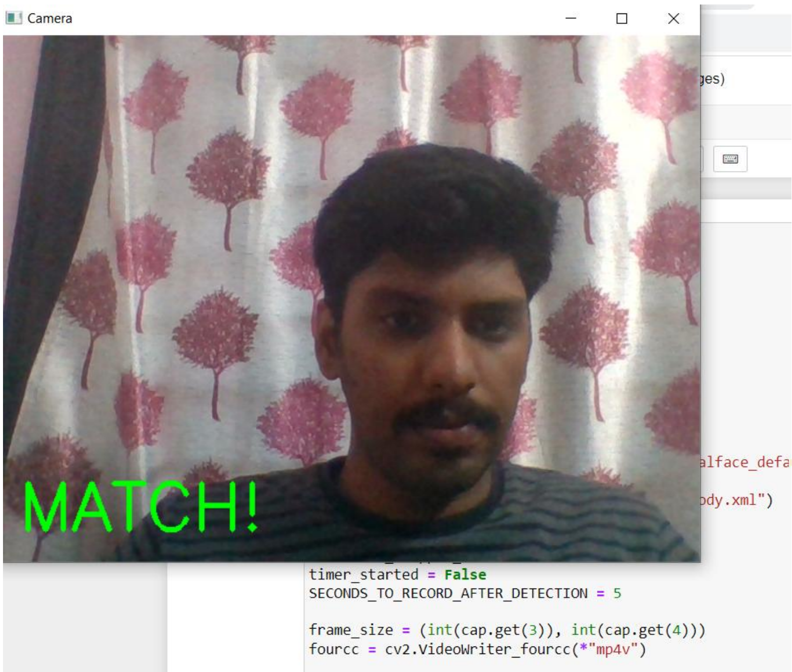


Fig no A3.3 Face Recognition

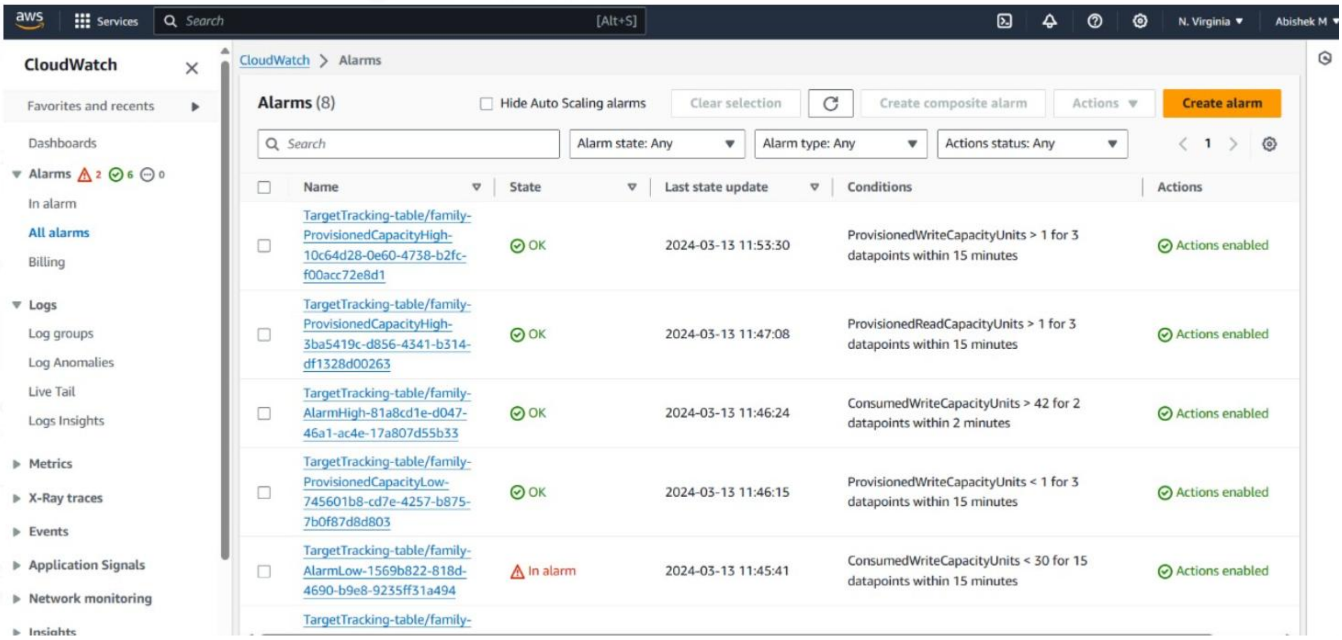


Fig no A3.4 CloudWatch

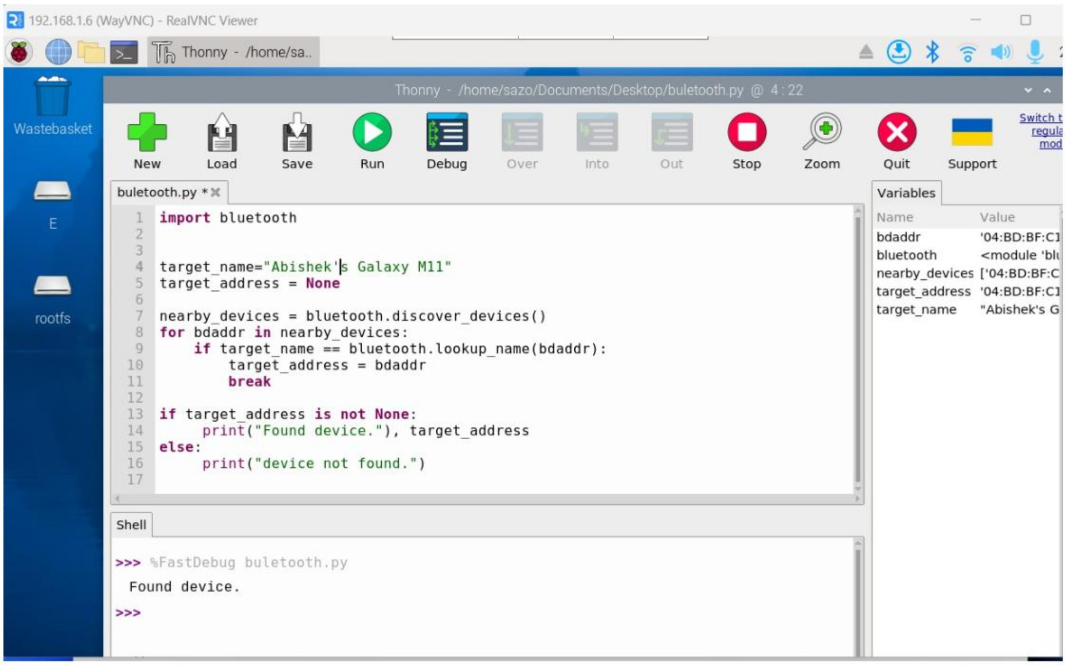


Fig no A3.5 Bluetooth Connectivity

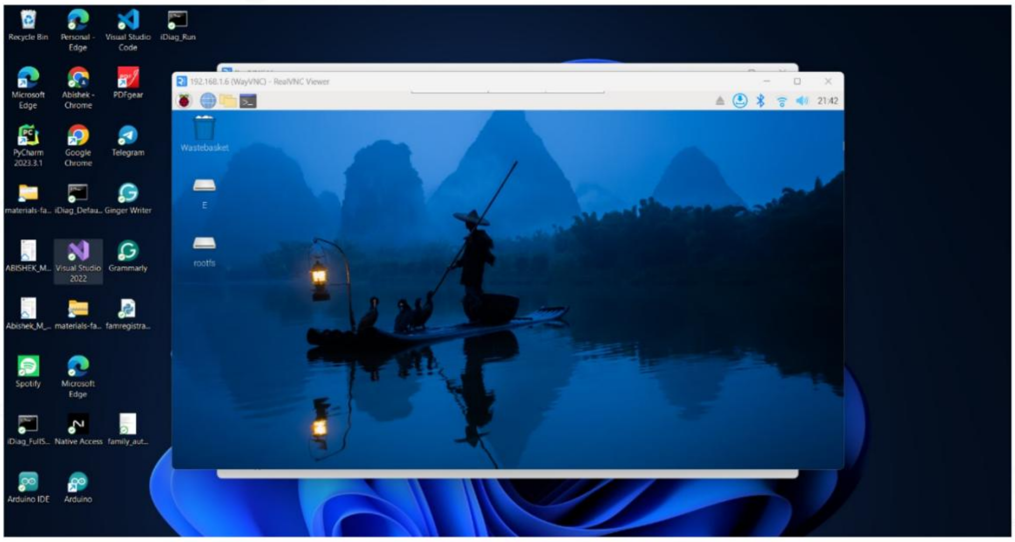


Fig no A3.6 VNC Viewer

A4: Plagiarism Report

RE-2022-222098 - Turnitin Plagiarism Report

by Abishek M

Submission date: 25-Mar-2024 07:44AM (UTC+0300)

Submission ID: 271711361848

File name: RE-2022-222098.pdf (532.26K)

Word count: 2999

Character count: 16680

NEW APPROACH FOR HOME AUTOMATION SYSTEM

Abishek M
Student cse Dept
Panimalar Engineering College
Chennai, India
abishek140518@gmail.com

S Naveen Kumar
Student cse Dept
Panimalar Engineering College
Chennai, India
naveenkumarpes2018@gmail.com

Lakshmi Priyan P
Student cse Dept
Panimalar Engineering College
Chennai, India
priyandev2002@gmail.com

Dr. Jabasheela L
Panimalar Engineering College
Chennai, India
csehod@panimalar.ac.in

Abstract — The use of face recognition technology in smart door closure systems is becoming increasingly popular because of its convenience, security and accuracy. In this article, we offer a smart door system connected with windows that uses face recognition and surveillance. The system consists of a camera module, a microcontroller Raspberry Pi, and an advanced locking mechanism. Motion sensor senses and turns on camera power. The module captures an image of the person standing at the gate, and our system processes the image using a facial recognition algorithm to identify the person. If the person is authorized, the door will unlatch automatically. The proposed system has been tested using a dataset of faces, and the windows are monitored. The system also includes a backup mechanism, such as a keypad or key, in the event of a malfunction or unavailability of facial recognition. If friends and family come, the owner will be notified on time. If an unfamiliar person reaches the door, the system begins recording.

The proposed system can be used in different locations such as homes, offices and hotels, to provide a secure and convenient way to access the premises throughout the building. The system is cost-effective, scalable and easy to install, making it an appealing option for smart door locking solutions.

Keywords—recognition, RaspberryPi, Locking mechanism, Camera module, Access control, Security

I. INTRODUCTION

Access control systems are an essential aspect of securing premises and ensuring the safety of people and assets. Traditional door-locking systems, such as mechanical keys and magnetic stripe cards, are becoming obsolete due to their limitations in terms of security, convenience, and scalability. As technology evolves, biometric identification systems such as facial recognition are gaining in popularity due to their precision, reliability and ease of use.

In this article, [we offer Gates with Advancement which uses facial recognition technology to grant access to authorized persons and windows are connected and watched by sensors. these details are sent to our proposed system which informs the guest and status of home to the owner by notifying him.] The system consists of a camera module that captures the facial image of the person standing in front of the door, the door system that processes the image and performs facial detection, and unlocks the door if the person is authorized.

The system offers several advantages over traditional door lock systems. (A) Facial recognition provides a more secure and reliable way of unlocking compared to mechanical keys or magnetic stripe cards, in which the key can be lost, stolen, or duplicated. (B) The system is easy to use, as individuals do not need to carry any additional items such as keys or cards. (C), the system is scalable and can be easily integrated with other security systems to provide a comprehensive access control solution. (D)Windows of the home are also connected through Wi-Fi to the system. (E) The appliance had a backup power supply and a direct connection to the cellar network, which keeps the appliance alive all the time.

The proposed system has been tested using a dataset of faces, and the results show that it achieves high accuracy and reliability. The system also includes a backup mechanism, such as a keypad or a key, in case facial recognition fails or is unavailable.

Overall, the lock system using facial recognition technology offers a cost-effective, scalable, and secure way of granting access to authorized persons, making it an attractive option for various applications, such as homes, offices, and hotels.

II. LITERATURE SURVEY

Facial recognition technology has been widely researched and applied in various fields, including security, biometrics, and computer vision. In recent years, facial recognition has become more common in access control systems because of its accuracy, reliability and ease of use.

A number of studies have been conducted on the performance of facial recognition systems for access control applications. In a study by Jain and Ross, the authors evaluated the accuracy and speed of various facial recognition algorithms and concluded that the Eigenface algorithm achieved the best performance for access control applications. Similarly, in a study by Li et al., the authors proposed a face recognition system based on a deep convolutional neural network and achieved high accuracy in recognizing faces for access control.

, the authors proposed a facial recognition system based on a deep convolution neural network and achieved high accuracy in the recognition of faces for access control. In a study by Zhang et al., the authors proposed a smart door system that uses facial recognition and fingerprint recognition for access control. The system consisted of a camera module, a fingerprint sensor, a microcontroller unit, and a locking mechanism, and achieved high accuracy and security. Similarly, in a study by Chen et al., the authors proposed a smart door-locking system that uses facial recognition and voice recognition for access control. The system consisted ofa camera module, a microphone, a microcontroller unit, and a locking mechanism, and achieved high accuracy and convenience.

Overall, the literature survey indicates that facial recognition technology is a promising and effective solution for access control systems, and that door systems using facial recognition technology have been successfully implemented and tested. The proposed smart door-locking system using facial recognition technology builds on these studies and offers a cost-effective, scalable, and secure solution for granting access to authorized people.

III. OBJECTIVE

The main objective of this project is to design and implementa door system using facial recognition technology for access control and for security purposes notification is sent. Specifically, the project aims to:

Develop a camera module that can capture high-quality facial images for facial recognition. Implement a facial recognition algorithm that can accurately recognize authorized individuals and deny access to unauthorized individuals.

This system can process facial images and control the lock mechanism of the door.

An integrated system with a locking mechanism provides a secure and convenient access solution.

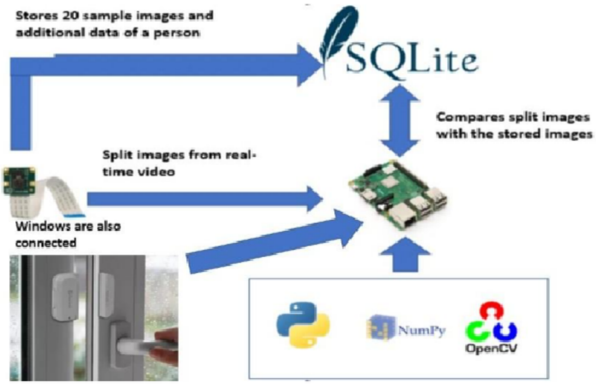
This system sends an alert message to the owner's phone when a family member or friend comes to the home (or) an unauthorized person attempts to enter the home which makes your home more interactive and secure.

Using a live updating dataset of faces. The performance of the system's accuracy, reliability, and scalability is increased.

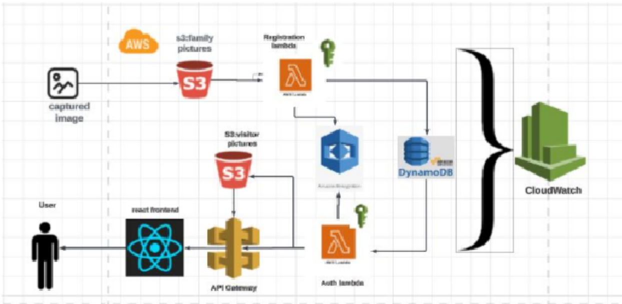
Overall, the objective of this project is to develop a smart door system using recognition and artificial intelligence that can provide a secure solution for homes while alerting the homeowner when an unauthorized individual attempts to enter the home.

IV. METHODOLOGY

Camera module: The first step is to set up a camera at the entrance of the door to capture images of people trying to enter. The camera should be positioned in a way to capture clear facial images.



This advanced security system combines cutting-edge AI and cloud storage for complete peace of mind. An AI-powered camera with facial recognition captures clear footage and identifies individuals. Magnetic door sensors and infrared sensors detect intrusion attempts, even in low-light conditions. The ESP32 microcontroller and Raspberry Pi process data, while motion detection software triggers recordings and alerts. Facial recognition software identifies authorized personnel. All data, including video, sensor readings, and recognition results, are stored securely in a local MySQL database and backed up to an AWS S3 cloud storage bucket for remote access. This system offers enhanced security, improved access control through facial recognition, and remote monitoring capabilities. Remember, strong security protocols and clear data policies are essential for a well-rounded security solution.



Building a facial recognition app on AWS involves several steps that integrate various AWS services to create a robust and scalable application. The process begins with setting up an AWS S3 bucket to store images that will be used for facial recognition. Once the S3 bucket is in place, AWS Rekognition is employed to analyze the images and perform the facial recognition tasks. This service uses machine learning algorithms to compare faces in images with a database of known faces and identify matches.

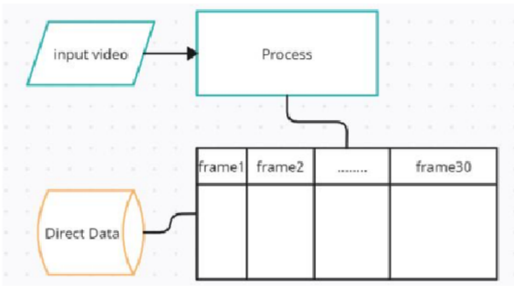
Next, AWS Lambda functions are created to handle the processing logic. These functions are triggered by events, such as the uploading of a new image to the S3 bucket, and they use the

Rekognition service to analyze the images. The results from Rekognition, which include the identification of faces and any matches found, are then stored in AWS DynamoDB, a NoSQL database service that provides fast and flexible data storage.

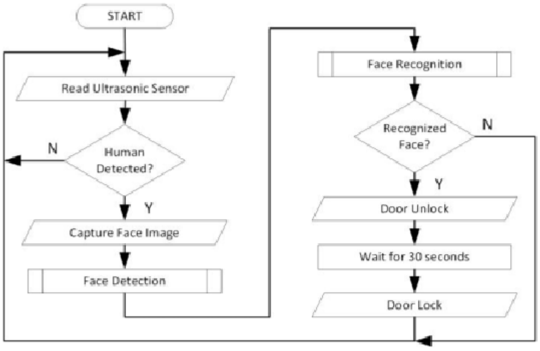
To make the facial recognition functionality accessible, an API is set up using AWS API Gateway. This service allows for the creation of RESTful APIs that can be called from a web or mobile application. The API acts as a front door to the Lambda functions, allowing them to be invoked with HTTP requests. The API Gateway is configured to handle these requests, invoke the appropriate Lambda function, and return the results to the caller.

Finally, the entire process is monitored and managed through the AWS Management Console, which provides tools for deploying, configuring, and monitoring the various AWS services involved in the application. This includes setting up permissions and security measures to ensure that the facial recognition app is secure and that only authorized users can access the sensitive data it processes.

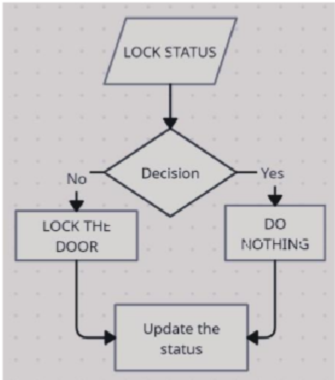
In summary, building a facial recognition app on AWS involves the integration of S3 for image storage, Rekognition for image analysis, Lambda for processing logic, DynamoDB for data storage, and API Gateway for API management. Each service plays a critical role in the overall functionality and performance of the application, and AWS provides the necessary tools to manage and monitor the system effectively. The result is a powerful facial recognition app that leverages the best of AWS's cloud capabilities.Create database: We create a better and more efficient hive DB which uses machine learning and storage of authorized persons along with their images that will be used to comparewith the captured image. The database should be stored and encrypted in a secure way to prevent unauthorized access.



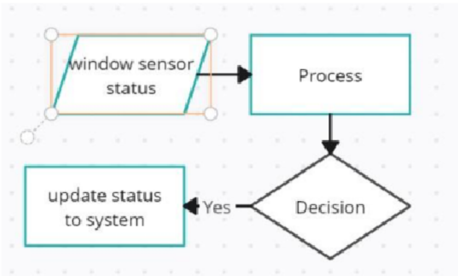
Facial recognition: We will use a facial recognition library or API to compare the captured image with the images in the database. The algorithm will use machine learning and deep learning techniques to identify and verify the individual's identity.



Facial recognition: We will use a facial recognition library or API to compare the captured image with the images in the database. The algorithm will use machine learning and deep learning techniques to identify and verify the individual's identity.

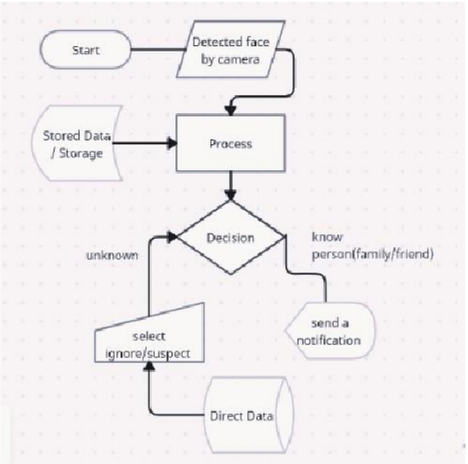


Interface with door lock: Once the face recognition system is deployed, we will interface it with the door lock mechanism. The system will automatically control the lock based on the recognition results. If there's a match to the database, the door will unlock, and if there isn't a match, the door will remain locked.



Interface with windows:

The windows are watched frequently and their status is sent to system. Which improve the safety of the entire house.

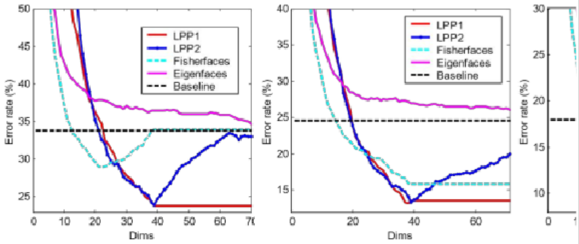


Test and evaluate: We will test system performance by capturing images of a diverse group of people and assessing system accuracy and reliability. We will also compare the system with existing access control solutions to evaluate its effectiveness.

Overall, the methodology for this project involves setting up a camera to capture images, creating a database of authorized individuals, using a facial recognition library or API to verify their identity, interfacing with the door lock mechanism to control access, implementing an alert system in case of unauthorized access, and testing and evaluating the system to ensure its accuracy and reliability.

Mysql /S3 Database

The project aimed to develop an efficient and cost-effective face recognition system by leveraging the capabilities of a Raspberry Pi single-board computer, a local MySQL database, and Amazon Web Services (AWS). The Raspberry Pi served as the core processing unit, running Python scripts that interfaced with the AWS SDK. A MySQL database was deployed on the Raspberry Pi to store facial recognition data, including encoded face vectors and associated metadata. Images captured by the system were temporarily stored on the Raspberry Pi and then uploaded to an Amazon S3 bucket for persistent storage and future retrieval. The AWS Rekognition service was integrated to perform face detection and recognition on the uploaded images, leveraging its pre-trained deep learning models. The recognized facial data, along with relevant information, was then stored in the local MySQL database for subsequent analysis and retrieval. This setup allowed for real-time face recognition, scalable image storage, and efficient data management, demonstrating the potential of combining edge computing with cloud services for computer vision applications.



Number plate detection using YOLOv5 involves a series of steps that leverage deep learning techniques to accurately identify and read vehicle license plates from digital images. YOLOv5, which stands for 'You Only Look Once version 5', is a state-of-the-art object detection algorithm known for its speed and accuracy. The process begins with the input image being passed through a series of convolutional layers, which help in feature extraction. These features are then used by the algorithm to predict bounding boxes and class probabilities. For number plate detection, the model has been trained on a vast dataset of images containing various number plates in different conditions and angles, enabling it to recognize patterns and characteristics specific to number plates. Once the number plate is detected, the region of interest is cropped from the image, and optical character recognition (OCR) techniques are applied to extract the alphanumeric characters on the plate. This information can then be used for various applications such as automated toll collection, traffic monitoring, and vehicle tracking systems. The effectiveness of YOLOv5 in number plate detection lies in its ability to process images in real-time while maintaining high accuracy, making it an invaluable tool in the field of computer vision and automated surveillance systems.

V. RESULT & CONCLUSION

This system is an innovative solution that combines the latest recognition technology with advanced access control to improve safety at home. Makes sense to home. The system is interconnected and manages the doors and windows of the house. The camera module captures live video (if the camera is 30 frames per second, we can get 30 pictures to study) and divides the frame to analyze learning on the device. These images are saved to the database. The magnetic anti-theft magnetic device with Wi-Fi module inbuild is fixed the windows and by granting access only to authorized persons, the system provides a safer environmentand peace of mind for the homeowner. The biometric may also be added as of purpose.

The storage space is well managed. Where as in old CCtv's camera runs 24/7 so the storage should be most. In contrast,our device stores the image of the person who is detected by the camera module. The Sixfab 3G/4G& LTE Base HAT grants device(Raspberry Pi)interface bridge betweenmini PCIe cellular modems. This makes the device 24/7 online .In Additional alert functionality for break and enter attempts increases the overall security of the system.

The future improvement provides artificial intelligence robot with integrated voice that makes the system more interactive and user-friendly. In addition, speech recognitionis added to the system, which will be more effective. As perthe security requirements the device can be modified.

VI. REFERENCE

1. G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. Kumar Mandula, Ramu Parupalli, Murty CHAS, Magesh E, Rutul Lunagariya. "Mobile based Horne Automation using Internet of Things (IoT)". 2015 International Conference on Control, instrumentation, Communication and Computational Technologies (lCCICCT), 2015 IEEE 978- 1-4673-9825-1.
3. raka, Marc Ghobril, Sami Malek, Rouwaida Kanj, Ayman Kayssi. "Low cost Arduino/Android-based Energy-Efficient Home Automation System with Smart Task Scheduling". Fifth International Conference on Computational Intelligence, Communication Systems and Networks. 2013. Crossref.
4. Atzori Luigi, Antonio Iera and Giacomo Morabito. "The internet of things: A survey". Computer Networks. 2010; 54(15):2787-2805.I.S.
5. Byeongkwan Kang, Sunghoi Park, Tacklim Lee and Sehyun Park. "TheIoT based Monitoring System uses Tri-level Context Making Model for Smart Home Services". 2015 IEEE International Conference on Consumer Electronics (ICCE). 2015. Cross.
6. Andreas Kamilaris, Andreas Pitsillides. "Towards Interoperable and Sustainable Smart Homes". Proceedings, Paul Cunningham and Miriam Cunningham (Eds) IIMC International Information Management Corporation. 2013.
7. Pranay P Gaikwad, Jyotsna P Gabhane, Snehal S Golait. "A Survey based on Smart Homes System Using Internet-of Things". 2015 International Conference on Computation of Power, Information and Communication. 2015.
8. [TP91] M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71-86, 1991.
9. [ZKC+98] W. Zhao, A. Krishnaswamy, R. Chellappa, D. Swets and J. Weng. Discriminant analysis of principal components for face recognition, pages 73-85. Springer Verlag Berlin, 1998.
10. [GHW12] M. Günther, D. Haufe and R.P. Würtz. Face recognition with disparity corrected Gabor phase differences. In Artificial neural networks and machine learning, volume 7552 of Lecture Notes in Computer Science, pages 411-418. 9/2012.
11. [ZSG+05] W. Zhang, S. Shan, W. Gao, X. Chen and H. Zhang. Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition. Computer Vision, IEEE International Conference on, 1:786-791, 2005.
12. [MM09] C. McCool, S. Marcel. Parts-based face verification using local frequency bands. In Advances in biometrics, volume 5558 of Lecture Notes in Computer Science. 2009.
13. [WMM+12] R. Wallace, M. McLaren, C. McCool and S. Marcel. Cross-pollination of normalisation techniques from speaker to face authentication using Gaussian mixture models. IEEE Transactions on Information Forensics and Security, 2012.
14. [WMM+11] R. Wallace, M. McLaren, C. McCool and S. Marcel. Inter-session variability modelling and joint factor analysis for face authentication. International Joint Conference on Biometrics. 2011.

RE-2022-222098-plag-report

ORIGINALITY REPORT

9%	2%	6%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Y. Jeevan Nagendra Kumar, G. Shiva Kumar, Kurikelly Rishik, D. Dinesh Chaitanya, Sri Charan. "Face Recognition and Raspberry Pi Powered Smart Door Unlocking System", E3S Web of Conferences, 2023 Publication	3%
2	mafiadoc.com Internet Source	1%
3	Sajad Rezaei, Jafar Tanha, Zahra Jafari, SeyedEhsan Roshan, Mohammad-Amin Memar Kochebagh. "A Deep CNN Model Based Ensemble Approach for Semantic and Instance Segmentation of Indoor Environment", 2023 13th International Conference on Computer and Knowledge Engineering (ICCKE), 2023 Publication	1%
4	Submitted to University of East London Student Paper	1%
5	Submitted to The University of the West of Scotland	1%

Student Paper

6	Submitted to Arab Academy for Science, Technology & Maritime Transport CAIRO Student Paper	1%
7	S Surya, P Kausik, G Diyaneshwaran, R Raffik. "Railway Gate Automation Using Onboard Computing Techniques", 2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), 2023 Publication	<1%
8	Wei He, Yang Mi, Xiangdong Ding, Gang Liu, Tao Li. "Two-stream cross-attention vision Transformer based on RGB-D images for pig weight estimation", Computers and Electronics in Agriculture, 2023 Publication	<1%
9	Submitted to Massey University Student Paper	<1%
10	hrcak.srce.hr Internet Source	<1%
11	ijece.iaescore.com Internet Source	<1%
12	www.ijeat.org Internet Source	<1%

13 Tin Nguyen Chung Ta, Duy Pham-Khac, Quan Le-Trung. "Development of Libelium-based Reconfigurable Solutions For Smart City Applications", 2022 RIVF International Conference on Computing and Communication Technologies (RIVF), 2022
Publication
<1%

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] Atzori Luigi, Antonio Iera and Giacomo Morabito. “The internet of things: A survey”. Computer Networks. 2010; 54(15):2787-2805.

[3] Byeongkwan Kang, Sunghoi Park, Tacklim Lee and Sehyun Park. “The IoT based Monitoring System uses Tri-level Context Making Model for Smart Home Services”. 2015 IEEE International Conference on Consumer Electronics (ICCE). 2015.

[4] Kumar Mandula, Ramu Parupalli, Murty CHAS, Magesh E, Rutul Lunagariya. “Mobile based Horne Automation using Internet of Things (IoT)”. 2015 International Conference on Control, instrumentation, Communication and Computational Technologies (lCCICCT), 2015 IEEE 978- 1-4673-9825-1.

[5] raka, Marc Ghobril, Sami Malek, Rouwaida Kanj, Ayman Kayssi. “Low cost Arduino/Android-based Energy-Efficient Home Automation System with Smart Task Scheduling”. Fifth International Conference on Computational Intelligence, Communication Systems and Networks. 2013. Crossref.

[6] Pranay P Gaikwad, Jyotsna P Gabhane, Snehal S Golait. “A Survey based on Smart Homes System Using Internet-of Things”. 2015 International Conference on Computation of Power, Information and Communication. 2015.

[7] Andreas Kamilaris, Andreas Pitsillides. "Towards Interoperable and Sustainable Smart Homes". Proceedings, Paul Cunningham and Miriam Cunningham (Eds) IIMC International Information Management Corporation. 2013.

[8] [ZKC+98] W. Zhao, A. Krishnaswamy, R. Chellappa, D. Swets and J. Weng. Discriminant analysis of principal components for face recognition, pages 73- 85. Springer Verlag Berlin, 1998.

[9] [TP91] M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71-86, 1991.

[10] [MM09] C. McCool, S. Marcel. Parts-based face verification using local frequency bands. In Advances in biometrics, volume 5558 of Lecture Notes in Computer Science. 2009.

[11] [ZSG+05] W. Zhang, S. Shan, W. Gao, X. Chen and H. Zhang. Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition. Computer Vision, IEEE International Conference on, 1:786-791, 2005.

[12] [WMM+11] R. Wallace, M. McLaren, C. McCool and S. Marcel. Inter-session variability modelling and joint factor analysis for face authentication. International Joint Conference on Biometrics. 2011.

[13] [WMM+12] R. Wallace, M. McLaren, C. McCool and S. Marcel. Cross-pollination of normalisation techniques from speaker to face authentication using Gaussian mixture models. IEEE Transactions on Information Forensics and Security, 2012.

[14] [GHW12] M. Günther, D. Haufe and R.P. Würtz. Face recognition with disparity corrected Gabor phase differences. In Artificial neural networks and machine learning, volume 7552 of Lecture Notes in Computer Science, pages 411-418. 9/2012.