**Project Report**

**AC Control System with Real-Time Monitoring**

**Table of Contents**

1. Introduction
2. Project Overview
3. System Architecture
   * Server
   * AC Client
   * Web Client
   * People Counting Module
   * Firebase Integration
4. Key Features
5. Setup and Installation
6. Usage
7. Code Explanation
   * Server
   * AC Client
   * Web Client
   * People Counting Module
   * Firebase Integration
8. Testing and Validation
9. Future Improvements
10. Conclusion

**1. Introduction**

This project aims to develop an automated AC control system that adjusts the temperature and fan speed based on real-time monitoring of people count in a given area. By integrating multiple components, including a server, client applications, and cloud-based data storage, the system ensures efficient and automated management of the AC unit.

**2. Project Overview**

The system consists of the following major components:

1. **Server**: Manages client connections, processes requests, and communicates with Firebase.
2. **AC Client**: A Tkinter-based application for controlling the AC unit.
3. **Web Client**: A Flask-based web interface for real-time data visualization.
4. **People Counting Module**: Utilizes YOLO for real-time people counting.
5. **Firebase Integration**: Provides persistent data storage.

**3. System Architecture**

**Server**

The server is responsible for handling client connections, processing their requests, and managing data transfer between clients and Firebase.

**AC Client**

The AC Client is a Tkinter-based GUI application that dynamically adjusts the AC settings based on real-time people count data.

**Web Client**

The Web Client is a Flask-based web application that provides a user-friendly interface to visualize real-time AC status and historical data.

**People Counting Module**

The People Counting Module uses YOLO for object detection to count the number of people in the camera's view.

**Firebase Integration**

Firebase is used to store and retrieve data, ensuring persistent and accessible records of AC status and people count.

**4. Key Features**

1. **Server-Client Architecture**
   * Multi-threaded server handles multiple client connections.
   * Clients communicate with the server to send and receive data.
2. **Real-Time People Counting**
   * YOLO model for real-time object detection and people counting.
3. **AC Control**
   * Dynamic adjustment of temperature and fan speed based on people count.
   * Tkinter GUI for real-time control.
4. **Cloud Integration**
   * Firebase for data storage and retrieval.
5. **Web Interface**
   * Flask web application for real-time data visualization.
   * Responsive design with Bootstrap.

**5. Setup and Installation**

**Clone the repository:**

sh

Copy code

git clone https://github.com/yourusername/ac-control-system.git

cd ac-control-system

**Install dependencies:**

sh

Copy code

pip install -r requirements.txt

Ensure you have OpenCV and the YOLO model installed and configured.

**Configure Firebase:**

Add your Firebase credentials JSON file to the project directory.

**Run the server:**

sh

Copy code

python server.py

**Run the AC client:**

sh

Copy code

python ac\_client.py

**Run the web client:**

sh

Copy code

python ui\_client.py

**6. Usage**

* **AC Control**: The Tkinter GUI shows the current status of the AC unit and adjusts settings based on the real-time people count.
* **Web Interface**: Access the Flask web application through your browser to visualize real-time AC status and historical data.

**7. Code Explanation**

**Server**

The Server class handles client connections and processes their requests. It also interacts with the Firebase database to store and retrieve data.

python

Copy code

class Server:

def \_\_init\_\_(self, host='127.0.0.1', port=65432):

# Initialization code

def start(self):

# Code to start the server and accept client connections

def handle\_client(self, client\_socket):

# Code to handle client requests

def close(self):

# Code to close the server

**AC Client**

The AC\_Client class is a Tkinter-based application that controls the AC unit. It dynamically updates the temperature and fan speed based on the real-time people count.

python

Copy code

class AC\_Client:

def \_\_init\_\_(self, root, host='127.0.0.1', port=65432, people\_per\_degree=2):

# Initialization code

def update\_status(self):

# Code to update the AC status in the GUI

def begin(self, update\_delay=5, status\_delay=10):

# Code to start the AC control loop

**Web Client**

The UI\_client class connects to the server to fetch data and display it using Flask.

python

Copy code

class UI\_client:

def \_\_init\_\_(self, server\_ip='127.0.0.1', port=65432):

# Initialization code

def receive\_data(self):

# Code to receive data from the server

def close\_connection(self):

# Code to close the client connection

**People Counting Module**

The People class uses YOLO for object detection to count the number of people in the camera's view.

python

Copy code

class People:

def \_\_init\_\_(self):

# Initialization code

def get\_count(self):

# Code to count the number of people in the camera's view

**Firebase Integration**

The FBDB class manages interactions with Firebase.

python

Copy code

class FBDB:

def \_\_init\_\_(self, path\_to\_service\_key='firebase\_credentials.json', databaseURL='https://your-database-url.firebaseio.com/'):

# Initialization code

def push(self, path="status", value=None):

# Code to push data to Firebase

def get(self, path="status"):

# Code to retrieve data from Firebase

**8. Testing and Validation**

* **Unit Testing**: Test individual components (e.g., server, clients, Firebase interactions) to ensure they work as expected.
* **Integration Testing**: Test the entire system to ensure all components work together seamlessly.
* **Performance Testing**: Test the system under various loads to ensure it can handle multiple clients and real-time data processing.

**9. Future Improvements**

1. **Enhanced Error Handling**: Implement more robust error handling throughout the system.
2. **Advanced Control Algorithms**: Develop more sophisticated algorithms for optimizing AC control based on people count.
3. **User Authentication**: Add user authentication for secure access to the web interface.
4. **Scalability**: Improve the system's scalability to handle more clients and larger datasets.

**10. Conclusion**

The AC Control System with Real-Time Monitoring is a comprehensive solution for automating AC control based on real-time people count data. By integrating multiple components, including server-client architecture, YOLO-based people counting, and Firebase for cloud storage, the system ensures efficient and automated management of the AC unit. With further improvements and optimizations, this system can be a valuable tool for energy management in various settings.