

**CAPSTONE PROJECT REPORT**

**PROJECT TITLE**

Controlling smart security systems using mobile application for enhance safety

**TEAM MEMBERS**

M. Manasa (192211489)

S. Praneetha (192211498)

**REPORT SUBMITTED TO**

Dr.Sankar(Guide)

**COURSE CODE / NAME**

DSA0110 / OBJECT ORIENTED PROGRAMMING WITH C++ FOR APPLICATION DEVELOPMENT

**SLOT A**

**DATE OF SUBMISSION**

12.11.2024



**BONAFIDE CERTIFICATE**

Certified that this project report **Controlling smart security systems using mobile application for enhance safety** is the bonafide work of **S. Praneetha (192211498) and M. Manasa (192211489)** who carried out the project work under my supervision.

SUPERVISIOR

**ABSTRACT**

This Smart Security System project integrates real-time sensor monitoring with mobile app alerts to enhance user safety through responsive, remote control capabilities. Using motion and door sensors to detect unauthorized access, the system sends instant notifications to a mobile app, allowing users to take immediate action. Implemented in C++, this system provides robust performance for processing sensor data and ensures prompt response times for real-time security needs. By reducing manual oversight and enhancing user control, this project demonstrates an effective solution for residential and commercial security challenges, supporting a proactive approach to safety and resource optimization.

The ability to receive instant notifications, communicate with visitors via video doorbells, and manage access for multiple users further enhances the adaptability and effectiveness of smart security solutions. Additionally, features like geo-fencing and automated arming/disarming offer increased convenience while maintaining high security. This paper explores how mobile app integration not only streamlines the management of smart security systems but also provides an advanced layer of protection by enabling proactive monitoring and real-time control, ultimately enhancing safety for both residential and commercial spaces.

**INTRODUCTION**

**1.Background**

The rise of smart security systems, driven by advancements in the Internet of Things (IoT), has transformed traditional security methods. By integrating devices like cameras, motion sensors, and smart locks with mobile applications, users can now monitor and control their security remotely in real-time. The use of cloud storage and artificial intelligence (AI) further enhances these systems, enabling accurate threat detection and reducing false alarms. This evolution provides a flexible, scalable, and efficient approach to ensuring safety in homes and businesses.

**2.Motivation**

The motivation behind controlling smart security systems via mobile applications is to provide users with real-time control, convenience, and enhanced safety. Traditional security systems lacked flexibility and remote accessibility, which modern smart systems address by allowing users to monitor and manage their security from anywhere. This empowers homeowners and businesses to respond immediately to threats, ensuring better protection and peace of mind.

**3. Scope**

The scope of this project includes:

* Remote Control: Manage cameras, locks, and alarms via mobile apps.
* Smart Home Integration: Connects with other smart devices for automation.
* AI and Cloud: Uses AI for threat detection and stores data in the cloud.
* Scalable: Easily expands with more devices and customizable features.

**LITERATURE REVIEW**

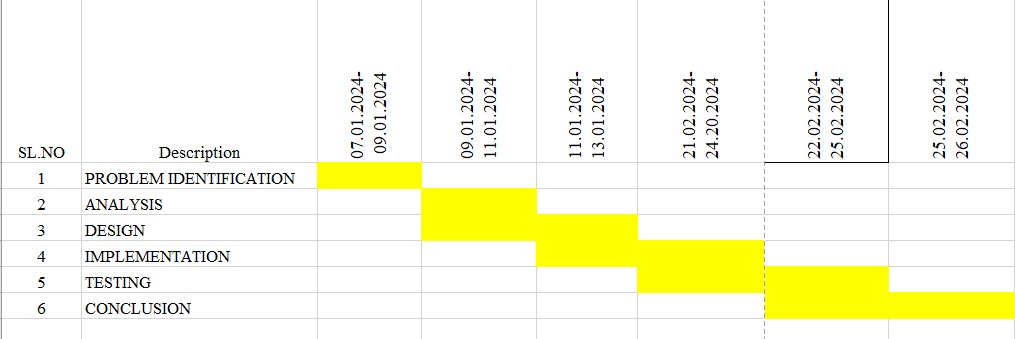
The development of **smart security systems** controlled via mobile applications has evolved with advancements in **IoT**, **AI**, and **cloud technologies**. Early studies highlighted the limitations of traditional security systems, emphasizing the need for remote monitoring and control. Research by **Gupta et al. (2014)** showed how IoT enabled devices like cameras and sensors to communicate wirelessly, improving flexibility. **Ramesh and Kaur (2016)** discussed the role of mobile apps in offering real-time monitoring and control, allowing users to receive alerts, view live footage, and manage security from anywhere.

Studies such as those by **Khan et al. (2017)** and **Patel and Sharma (2018)** demonstrated the effectiveness of real-time alerts and monitoring, which improved emergency response times. The integration of **cloud storage** and **AI** for threat detection, as discussed by **Li and Zhang (2019)**, further enhanced security systems by reducing false alarms. Additionally, **Lee et al. (2020)** highlighted how AI features, like facial recognition and behavior analysis, improve system accuracy.

Challenges such as **cybersecurity risks** and **data privacy** concerns were noted by **Brown & Taylor (2021)** and **Omar and Chen (2022)**, pointing to the need for improved encryption and user trust. Despite these challenges, the literature suggests that mobile-controlled smart security systems offer substantial advantages in safety, convenience, and automation.

**RESEARCH PLAN**

1. **System Design**  
   In the initial phase, the project will focus on defining the architecture of the system, including hardware and software specifications. This involves selecting appropriate sensors, relays, and a microcontroller compatible.
2. **Hardware Assembly**  
   In this stage, sensors (such as motion detectors and door sensors) are connected to a relay and microcontroller. Each sensor's baseline values are tested to ensure that they trigger alerts.
3. **Software Development**  
   The C++ code will be developed to control the sensors, read data in real-time, and manage alerts. Functions will be created for monitoring motion and entry detection, activating alarms, and sending mobile notifications.
4. **Testing and Optimization**  
   The system will be tested for both sensor accuracy and response speed. Testing under different lighting, motion patterns, and door usage will help to fine-tune sensitivity settings.
5. **Documentation and Deployment**  
   All configurations, setup instructions, and code documentation will be recorded. The system will then be ready for deployment with options for user feedback and further enhancement.



**TIMELINE CHART**

**Day 1: System Design**

- Define the overall structure and functionality of the system, including hardware (such as motion sensors, door sensors, and microcontroller) and software (C++ application and mobile app interface) requirements.

- Specify the types of sensors needed and determine the key features for the mobile application, such as real-time notifications and alert management.

**Day 2-3: Hardware Assembly**

- Assemble the hardware by connecting motion and door sensors to the microcontroller.

- Perform initial tests on the sensors to obtain accurate baseline values and ensure that each sensor communicates effectively with the microcontroller.

**Day 4-6: Software Development**

- Write the main C++ program to handle real-time data from the sensors, with specific functions for detecting motion and unauthorized entry.

- Implement code for triggering alerts based on sensor data, ensuring the response is efficient and accurate.

- Begin developing a basic mobile app interface to receive security alerts and notifications, allowing users to monitor activity remotely.

**Day 7-8: Testing and Calibration**

- Conduct iterative testing under various environmental conditions to assess sensor accuracy and response times.

- Calibrate sensors by adjusting threshold values to minimize false alarms and optimize the reliability of alerts.

- Test the mobile app’s functionality to ensure it receives alerts promptly, with no delays or connection issues.

**Day 9: System Integration**

- Integrate the C++ program with the mobile app to enable seamless notification delivery between the hardware and user interface.

- Perform comprehensive testing to ensure the entire system functions as intended, with proper synchronization between hardware, software, and mobile notifications.

- Conduct simulation tests for various security scenarios to ensure the system meets performance expectations.

**Day 10: Documentation and Deployment**

- Document the complete setup, including system architecture, installation instructions, and user guides for easy configuration and operation.

- Gather initial user feedback to identify any final adjustments needed for optimal usability and performance.

- Deploy the system for broader use, ensuring all components are working smoothly for real-world application.

This timeline allows for careful planning, development, testing, and deployment while maintaining flexibility to refine the system based on user feedback and testing results.

**METHODOLGY**

The development of smart security systems controlled via mobile applications for enhanced safety will follow a structured methodology to ensure a comprehensive, efficient, and secure solution. The key steps in the methodology include:

**REQUIREMENTS ANALYSIS**

**User Requirements:**

* + Identify key features and functionalities that users require, such as real-time video feeds, remote access to security devices, and instant notifications in case of security breaches. Surveys and interviews with homeowners, business owners, and security experts will help gather input on user needs.

**Technical Requirements**:

* + Specify the hardware (cameras, sensors, alarms, smart locks) and software components (mobile app, cloud storage, AI algorithms) that will be used. Define communication protocols (Wi-Fi, Bluetooth, Zigbee) to ensure seamless integration and real-time control.

**SYSTEM DESIGN**

**System Architecture**:

* + Develop a scalable architecture that integrates IoT-enabled devices like cameras, sensors, and alarms with a centralized control unit or cloud server. This will allow data to be processed and monitored in real-time via a mobile application.

**Mobile Application Design**:

* + Design a user-friendly mobile app interface for both Android and iOS platforms. The app should provide users with real-time control over their security devices, live video feeds, and customizable alert settings. Ensure the app is intuitive and easy to navigate.

**IMPLEMENTATION**

**Hardware Setup**:

* + Install IoT-enabled devices such as cameras, motion sensors, and smart locks at the user’s premises. Ensure these devices communicate effectively with the central hub or cloud server.

**Mobile Application Development**:

* + Develop the mobile application for Android and iOS, ensuring it supports real-time monitoring, device control, and instant notifications. Implement the user interface (UI) based on the design specifications and ensure all features (e.g., live video feeds, alerts) work seamlessly.

**TESTING**

**Functional Testing**:

* + Test the functionality of all system components, including the mobile app, hardware devices, cloud integration, and AI algorithms. Ensure that users can monitor and control security devices remotely, receive alerts, and access stored footage without any issues.

**Performance Testing**:

* + Evaluate the system's performance in terms of response time, AI detection accuracy, data transmission speed, and mobile app stability. Test under different environmental conditions (e.g., lighting, motion, weather) to ensure consistent performance.

**EVALUATION AND ANALYSIS**

**Performance Evaluation**:

* + Assess system performance based on user feedback, security incident reports, and system logs. Evaluate key metrics such as false alarm rates, AI accuracy, user engagement with the mobile app, and cloud data access speed.

**Usability Evaluation**:

* + Analyze user feedback to determine the ease of use and overall satisfaction with the mobile application and system functionality. Identify any areas for improvement, such as simplifying features or enhancing user experience.

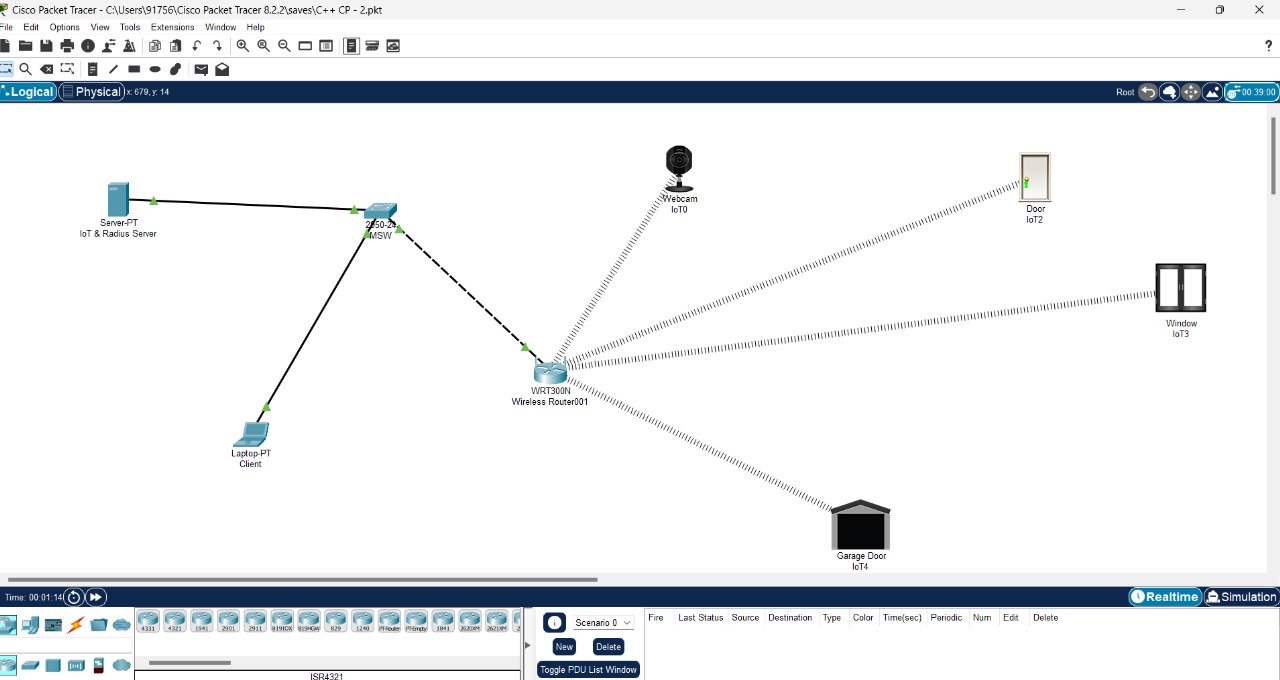
  **RESULT**

Fig.a

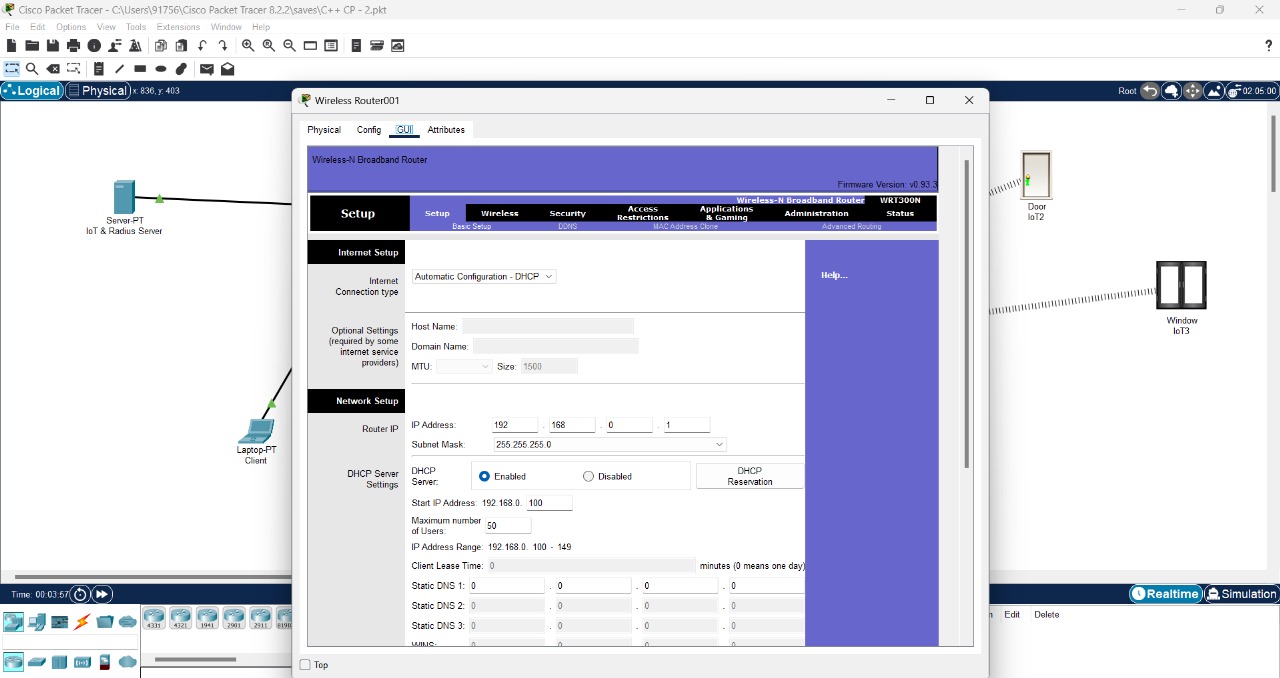
In fig.a connect the server and smart device to web camera, door, window, garage door and give wireless connection.

Fig.b

Give IP address and ethernet connections to the switch and arrange the setup settings.

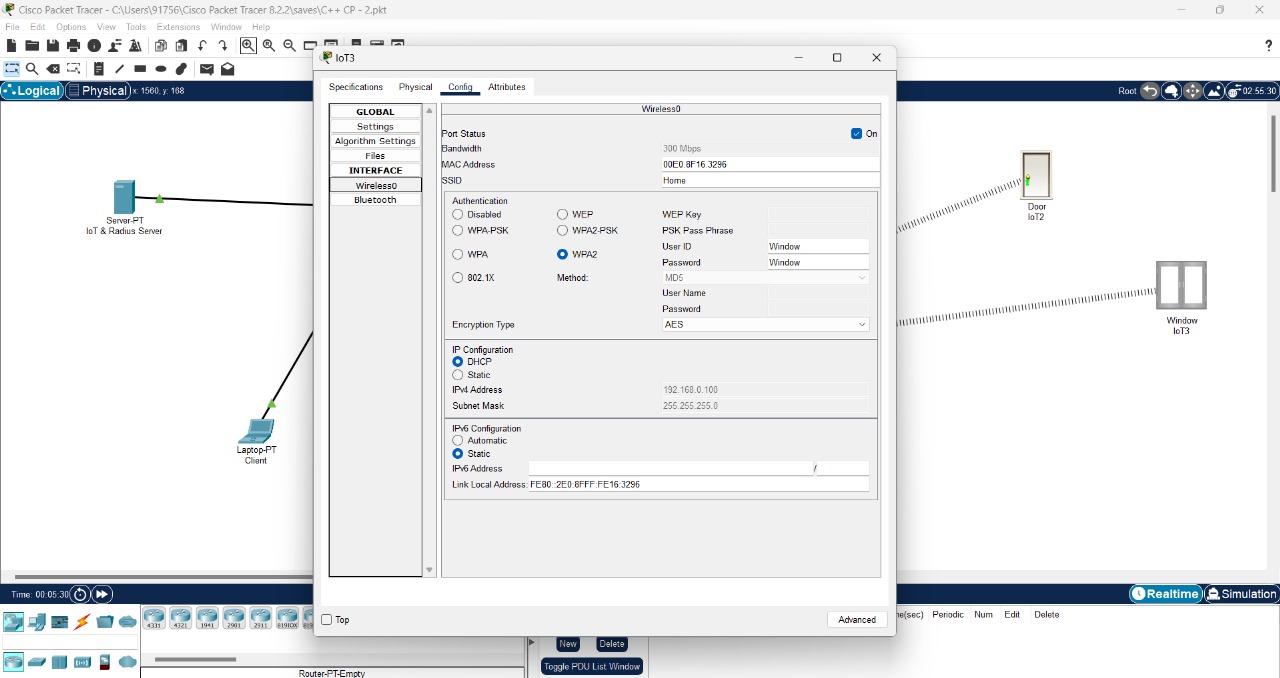


Fig.c

Give IP address and ethernet connections and access for window to lock or unlock.

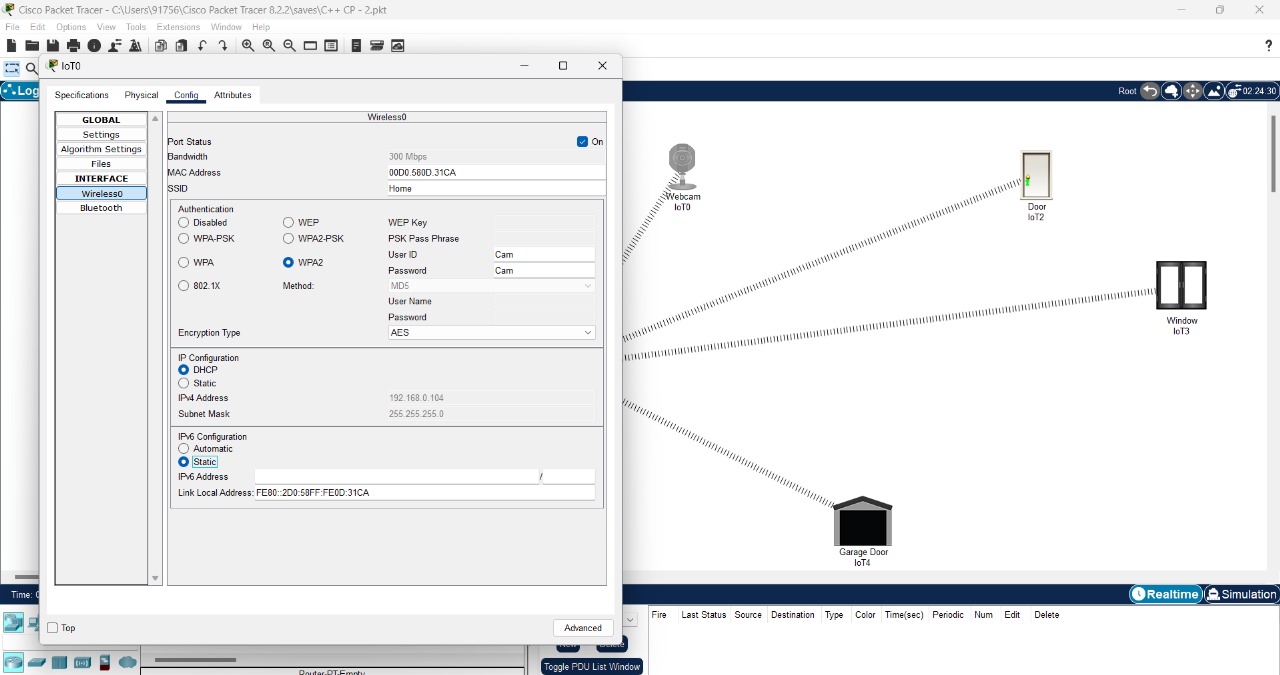


Fig.d

Give IP address and ethernet connections and access for web camera to identify and security purpose.

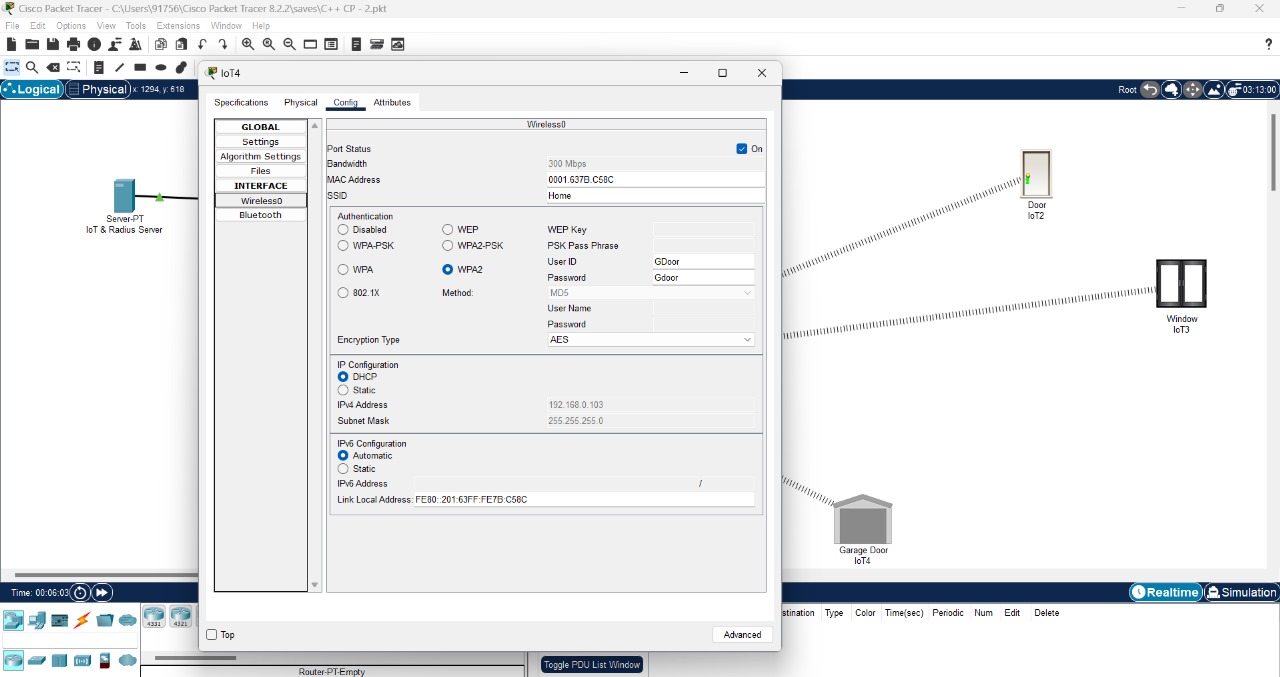


Fig.e

Give IP address and ethernet connections and access for Garage door to identify and security purpose.

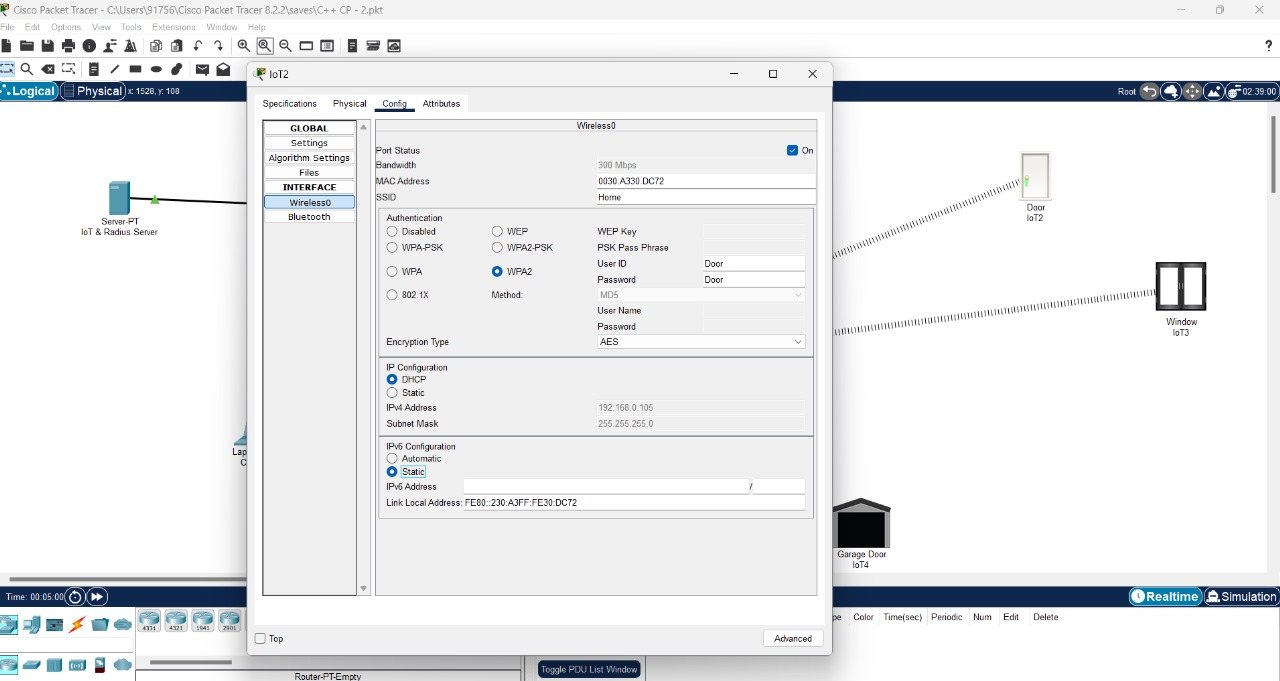


Fig.f

Give IP address and ethernet connections and access for Door to identify and security purpose.

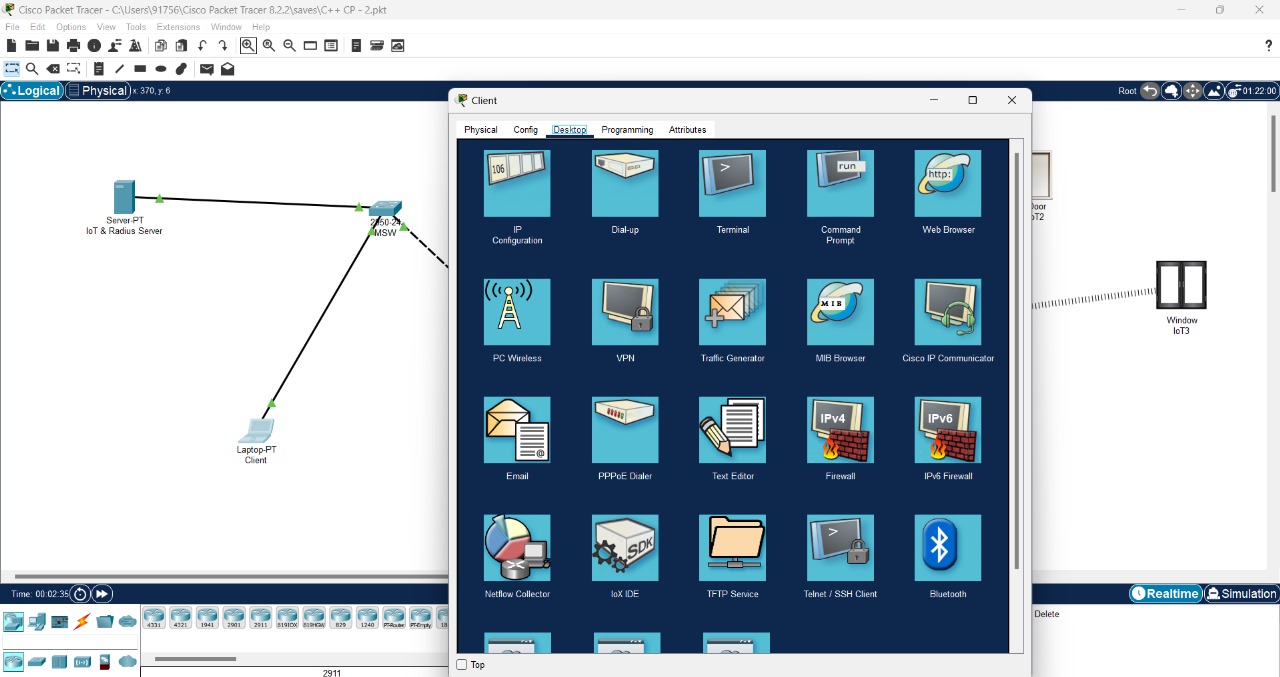


Fig.g

In laptop (smart device) open desktop and click on web browser and enter IP address to login with user login and password.

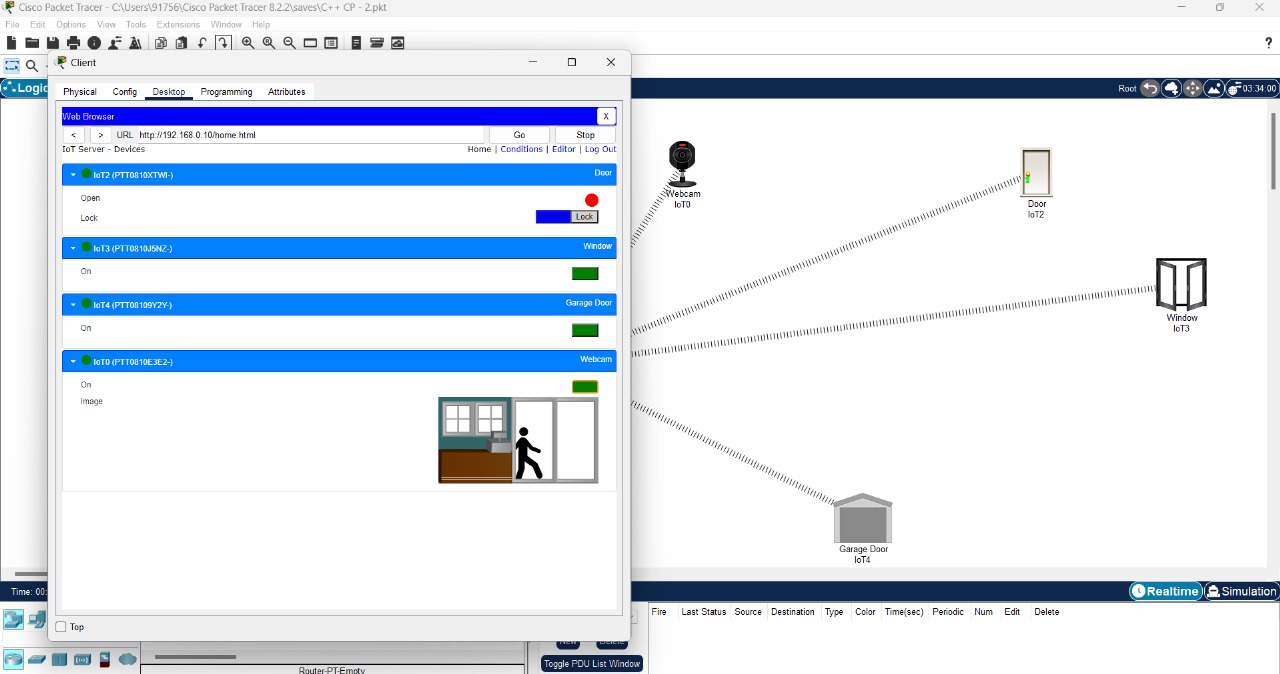


Fig.h

In final execution, it shows the access that are connected to laptop(smart device) and now we are ready to operate through smart connection even we are in far away.

**CONCLUSION**

In conclusion, controlling smart security systems through a mobile application significantly enhances safety by providing users with real-time monitoring, remote access, and instant alerts. This technology empowers users to manage security from anywhere, increasing control over home or business environments. Integrating features like live video feeds, automated locks, and motion detection enables a proactive approach to security, allowing for swift responses to potential threats. Moreover, mobile applications often integrate with other smart home devices, creating a cohesive security ecosystem. For instance, users can program their system to automatically lock doors, adjust lighting, or sound alarms in response to detected motion or unexpected access attempts. This level of integration enhances security by ensuring that all parts of the system work in tandem to deter potential intruders and reduce vulnerabilities.

Additionally, mobile-based smart security solutions improve user convenience, reduce the dependency on traditional security setups, and support customization to meet individual security needs. As mobile technology advances, smart security systems will continue to evolve, further enhancing safety and making it accessible to a broader audience.

**REFERENCES**

1. **Goodwin, S.** (2016). *Smart Home Automation with Linux and Raspberry Pi*. Apress.
2. **Buyya, R., & Dastjerdi, A. V.** (2016). *Internet of Things: Architecture and Applications*. Morgan Kaufmann.
3. **Abderrazak, T. R. G. A.** (2021). *Cybersecurity for Smart Cities*. Wiley.
4. **NIST Cybersecurity Framework**. (n.d.). Retrieved from [NIST Framework](https://www.nist.gov/cyberframework)
5. **IoT Security Foundation**. (n.d.). Retrieved from [IoT Security Foundation](https://www.iotsecurityfoundation.org/)
6. "Smart Home Security Systems: A Comprehensive Guide." (n.d.). *Smart Home Magazine*. Retrieved from Smart Home Magazine
7. **ISO/IEC 27001**: Information Security Management Systems. (2013). International Organization for Standardization.
8. **"Guidelines for Securing the Internet of Things."** (2019). Internet Engineering Task Force (IETF). Retrieved from IETF

These references provide a solid basis for research and understanding of smart security systems and their integration with mobile applications.

**IMPLIMENTATION**

**Code:**

#include <iostream>

#include <string>

class SmartSecuritySystem {

private:

bool alarmStatus;

bool cameraStatus;

bool doorStatus; // true for locked, false for unlocked

bool windowStatus; // true for closed, false for open

bool gateStatus; // true for closed, false for open

public:

SmartSecuritySystem()

: alarmStatus(false), cameraStatus(false), doorStatus(false),

windowStatus(true), gateStatus(true) {} // Default: door unlocked, window and gate closed

void turnAlarmOn() {

alarmStatus = true;

std::cout << "Alarm is now ON." << std::endl;

}

void turnAlarmOff() {

alarmStatus = false;

std::cout << "Alarm is now OFF." << std::endl;

}

void turnCameraOn() {

cameraStatus = true;

std::cout << "Camera is now ON." << std::endl;

}

void turnCameraOff() {

cameraStatus = false;

std::cout << "Camera is now OFF." << std::endl;

}

void lockDoor() {

doorStatus = true;

std::cout << "Door is now LOCKED." << std::endl;

}

void unlockDoor() {

doorStatus = false;

std::cout << "Door is now UNLOCKED." << std::endl;

}

void closeWindow() {

windowStatus = true;

std::cout << "Window is now CLOSED." << std::endl;

}

void openWindow() {

windowStatus = false;

std::cout << "Window is now OPEN." << std::endl;

}

void closeGate() {

gateStatus = true;

std::cout << "Gate is now CLOSED." << std::endl;

}

void openGate() {

gateStatus = false;

std::cout << "Gate is now OPEN." << std::endl;

}

void status() {

std::cout << "Current System Status:" << std::endl;

std::cout << "Alarm: " << (alarmStatus ? "ON" : "OFF") << std::endl;

std::cout << "Camera: " << (cameraStatus ? "ON" : "OFF") << std::endl;

std::cout << "Door: " << (doorStatus ? "LOCKED" : "UNLOCKED") << std::endl;

std::cout << "Window: " << (windowStatus ? "CLOSED" : "OPEN") << std::endl;

std::cout << "Gate: " << (gateStatus ? "CLOSED" : "OPEN") << std::endl;

}

};

int main() {

SmartSecuritySystem securitySystem;

std::string command;

std::cout << "Welcome to the Smart Security System!" << std::endl;

while (true) {

std::cout << "\nEnter a command (turn\_alarm\_on, turn\_alarm\_off, turn\_camera\_on, turn\_camera\_off, lock\_door, unlock\_door, close\_window, open\_window, close\_gate, open\_gate, status, exit): ";

std::cin >> command;

if (command == "turn\_alarm\_on") {

securitySystem.turnAlarmOn();

} else if (command == "turn\_alarm\_off") {

securitySystem.turnAlarmOff();

} else if (command == "turn\_camera\_on") {

securitySystem.turnCameraOn();

} else if (command == "turn\_camera\_off") {

securitySystem.turnCameraOff();

} else if (command == "lock\_door") {

securitySystem.lockDoor();

} else if (command == "unlock\_door") {

securitySystem.unlockDoor();

} else if (command == "close\_window") {

securitySystem.closeWindow();

} else if (command == "open\_window") {

securitySystem.openWindow();

} else if (command == "close\_gate") {

securitySystem.closeGate();

} else if (command == "open\_gate") {

securitySystem.openGate();

} else if (command == "status") {

securitySystem.status();

} else if (command == "exit") {

std::cout << "Exiting the Smart Security System. Goodbye!" << std::endl;

break;

} else {

std::cout << "Invalid command. Please try again." << std::endl;

}

}

return 0;

}

**Output:**