

# **NEXT GENERATION DEFENSE SECURITY: BLOCKCHAIN,IOT,DIGITAL TWIN, AND FACE RECOGNITION- BASED SMART MONITORING**

## **A PROJECT REPORT**

*Submitted by*

<b>RANJITH K</b>	<b>-</b>	<b>815821104018</b>
<b>SOUNDHARYA S</b>	<b>-</b>	<b>815821104025</b>
<b>PRIYADHARSHINI T</b>	<b>-</b>	<b>815821104709</b>
<b>SAKKARABANI S</b>	<b>-</b>	<b>815821104711</b>

*in partial fulfillment for the award of the degree  
of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**NELLIANDAVAR INSTITUTE OF TECHNOLOGY,**

**PUDHUPALAYAM**



**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY-2025**

# **ANNA UNIVERSITY: CHENNAI 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this project report ” **NEXT GENERATION DEFENSE SECURITY: BLOCKCHAIN, IOT, DIGITAL TWIN AND FACE RECOGNITION –BASED SMART MONITORING**” is the bonafide work of “ **RANJITH K (815821104018), SOUNDHARYA S (815821104025), PRIYADHARSHINI T (815821104709), SAKKARABANI S (815821104711)** ”who carried out the project work under my supervision.

### **SIGNATURE**

**Mr.V.RAGUNATH M.E.,**

### **HEAD OF THE DEPARTMENT**

Department of Computer  
Science and Engineering,  
Nelliandavar Institute of  
Technology,  
Pudhupalayam,  
Ariyalur 621704.

### **SIGNATURE**

**Mrs.R.ANBARASI M.E.,**

### **SUPERVISOR**

### **ASSISTANT PROFESSOR**

Department of Computer  
Science and Engineering,  
Nelliandavar Institute of  
Technology,  
Pudhupalayam,  
Ariyalur 621704.

Submitted for the viva-voce NELLIANDAVAR INSTITUTE OF TECHNOLOGY on

\_\_\_\_\_.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

First, we must acknowledge the almighty God's choices and abundant blessings. His providence touched every piece of this project. He is the sole source of success because we are tools in the hands of God omnipotent. He guided, enlighten, gave energy and knowledge to make this project possible and successful.

We express our sincere gratitude to our beloved Principal **Dr.R.KANNAN M.E,PhD**, for his sincere endeavor in educating us in this premier institution.

Our sincere and heartfelt gratitude to **Mr.V. RAGUNATH M.E**, Head of the Department for his kind help and guidance rendered during studies. We wholeheartedly acknowledge the words of inspiration given by our precious guide **Mrs.R.ANBARASI M.E**, Assistant Professor for completing the project work.

We propose our sincere thanks to the project coordinator **Mr.V.RAGHUNATH M.E**, who has motivated to completed the project successfully.

We convey our sincere thanks to all the staff members and lab assistants of the Computer Science department. Last but not least, we would like to thank our parents and friends for lending us a helping hand in all endeavours during the project work and always providing us with the necessary motivation to complete this project successfully.



## **ABSTRACT**

Authentication plays a critical role in safeguarding the defense industry's sensitive information, personnel, and assets from unauthorized access and security breaches. As defense operations become increasingly digitized, robust authentication mechanisms are essential to ensure the integrity and security of military systems, networks, and facilities. The rapid advancements in technology and the increasing complexity of defence operations necessitate the development of secure and efficient monitoring systems. This project presents a Internet of things based digital twin technology integrated with AI-powered facial recognition technology with block chain technology to address the unique security challenges faced by the defence industry. The proposed system leverages blockchain's inherent advantages, such as decentralization, immutability, and enhanced security, to create a robust platform for monitoring personnel, assets, and critical infrastructure. By utilizing facial recognition for real-time identification and verification of personnel, the system automates access control, allowing only authorized individuals into restricted areas or to sensitive resources. The biometric data of authorized personnel is securely stored on the blockchain, preventing tampering and ensuring data privacy. Every access attempt whether successful or unauthorized is recorded in the decentralized ledger, providing an immutable audit trail for accountability and compliance.

## **LIST OF ABBREVIATION**

<b>IOT</b>	-	Internet of Things
<b>AI</b>	-	Artificial Intelligence
<b>SDGs</b>	-	Sustainable Development Goals
<b>SHA 256</b>	-	Secure Hash Algorithm 256
<b>CPS</b>	-	Cyber –Physical System
<b>LSTM</b>	-	Long Short Term Memory
<b>IIOT</b>	-	Industrial IOT
<b>APT</b>	-	Advanced Persistent Threats

## **LIST OF FIGURES**

<b>FIGURE NO</b>	<b>FIGURE NAME</b>	<b>PAGE NO</b>
4.1.1	SYSTEM ARCHITECTURE	13
4.2.1	FLOW CHART	16
6.1.1	DEFENCE DATA SECURITY FRAME WORK	19
6.2.1	CAMERA-BASED FACE REGISTRATION MODULE	20
6.3.1	FACE RECOGNITION ACCESS CONTROL	21
6.4.1	BLOCK CHAIN-BASED TRANSACTION	22
6.5.1	REPORTING SYSTEM	23

## TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	<b>ABSTRACT</b>	<b>iv</b>
	<b>LIST OF ABBREVIATION</b>	<b>v</b>
	<b>LIST OF FIGURES</b>	<b>vi</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1. OVERVIEW	1
	1.2. OBJECTIVE	2
	1.3. SOFTWARE USED	3
	1.4. ISSUES AND CHALLENGES	3
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>4</b>
	2.1 EMERGING TECHNOLOGIES	4
	NATIONAL SECURITY	
	2.2 IOT IN DEFENSE MANUFACTURING	5
	2.3 WEARABLE COMPUTING AND 5G	6
	2.4 IOT-BASED FOOD QUALITY DEFENSE	7
	2.5 ADVERSARIAL DEFENSE FOR IOT IN SYSTEM	8
<b>3</b>	<b>PROBLEM DESCRIPTION</b>	
	3.1 EXISTING SYSTEM	9
	3.1.2 CHALLENGES IN EXISTING SYSTEM	9
	3.1.3 POTENTIAL IMPROVEMENTS	10
	3.1.4 DRAWBACKS	10



	3.2 PROPOSED SYSTEM	11
	3.2.1 FEATURES	11
	3.2.2 ADVANTAGES	12
<b>4</b>	<b>SYSTEM REQUIRMENTS</b>	<b>13</b>
	4.1 SYSTEM ARCHITECTURE	13
	4.2 FLOW CHART	14
<b>5</b>	<b>SYSTEM CONFIGURATION</b>	<b>17</b>
	5.1 HARDWARE REQUIREMENTS	17
	5.2 SOFTWARE REQUIRMENTS	17
<b>6</b>	<b>MODULES</b>	<b>18</b>
	6.1 DEFENCE DATA SECURITY	18
	FRAMEWORKS	
	6.2 CAMERA-BASED FACE REGISTRATION	19
	MODULE	
	6.3 FACE RECOGNITION AND ACCESS	20
	CONTROL	
	6.4 BLOCKCHAIN-BASED TRANSACTION	21
	6.5 REPORTING SYSTEM	22
<b>7</b>	<b>TESTING</b>	<b>24</b>
	7.1 UNIT TESTING	25
	7.2 INTEGRATION TESTING	25

	7.3 SYSTEM TESTING	25
	7.4 PERFORMANCE TESTING	26
	7.5 SECURITY TESTING	26
	7.6 USABILLITY TESTING	26
<b>8</b>	<b>EXPERIMENTAL RESULTS</b>	<b>27</b>
<b>9</b>	<b>SUSTAINABLE DEVELOPMENT TOOLS</b>	<b>28</b>
	9.1 INTRODUCTION TO SDG GOALS	28
	9.2 PROJECT ALIGNMENT WITH SDGs	28
	9.2.1 SDG 3	28
	9.2.2 SDG 9	29
	9..2 SDG 10	29
<b>10</b>	<b>CONCLUSION &amp; FUTURE WORK</b>	<b>30</b>
	10.1 CONCLUSION	30
	10.2 FUTURE ENHANCEMENTS	31
	<b>APPENDIX 1:SOURCE CODE</b>	<b>32</b>
	<b>APPENDIX 2:SCREEN SHOTS</b>	<b>68</b>
	<b>REFERENCES</b>	<b>82</b>

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 OVERVIEW**

In today's digitized defense ecosystem, the need for robust and intelligent security mechanisms is more critical than ever. With sensitive military data, infrastructure, and personnel constantly at risk from cyberattacks and unauthorized access, conventional surveillance systems fall short in delivering proactive, real-time threat prevention. Traditional approaches, such as CCTV monitoring and manual access control, are reactive by nature and heavily rely on human intervention, making them prone to inefficiencies, delays, and tampering. The defense sector requires an evolved security infrastructure that ensures constant vigilance, automation, and tamper-proof data handling to counter modern-day security threats effectively. The integration of cutting-edge technologies such as Blockchain, Internet of Things (IoT), Digital Twin, and Artificial Intelligence (AI) enables the creation of a smarter, more secure defense monitoring system. This project proposes a real-time, decentralized defense security framework that utilizes AI-powered facial recognition for accurate identity verification, IoT sensors for continuous surveillance, and blockchain for immutable and transparent data management. The system replaces traditional ID-based access methods with biometric verification, storing all biometric and access-related data securely on a distributed ledger. This ensures enhanced security, integrity, and accountability while eliminating vulnerabilities associated with centralized systems. Furthermore, smart contracts within the blockchain automate security responses, such as alert notifications, lockdowns, and access revocations in the event of unauthorized attempts. The use of the Grassmann algorithm for facial recognition ensures high accuracy even in complex conditions, making the system reliable under diverse environmental challenges. The modular design allows for scalability, making it suitable for various defense

environments — from secure military zones to mobile defense units. By combining these technologies, the project aims to redefine the standards of defense security with an intelligent, proactive, and resilient approach.

## **1.2 OBJECTIVES**

- **Ensure Real-Time Monitoring and Threat Detection** – Through IoT sensors and surveillance cameras, the system continuously gathers and analyzes data to detect any unusual activity or unauthorized presence in restricted defense zones.
- **Automate Access Control via Facial Recognition** – Using AI-driven facial recognition algorithms such as the Grassmann algorithm, the system accurately verifies the identity of individuals in real time, replacing traditional ID cards and passwords.
- **Secure Biometric Data with Blockchain** – All access logs and biometric data are stored on a decentralized blockchain ledger, ensuring tamper-proof records, privacy, and full transparency in security audits.
- **Enable Automated Security Response Using Smart Contracts** – When an unauthorized access attempt is detected, embedded smart contracts trigger predefined actions such as activating alarms, notifying authorities, or locking down high-risk areas.

### 1.3 SOFTWARE USED

- Operating System : Windows
- Back End : PYTHON
- Database : MYSQL
- IDE : PYCHARM

### 1.4 ISSUES AND CHALLENGES

#### 1. Security & Privacy Concerns

- **Biometric Data Protection:** Storing facial recognition data on the blockchain requires stringent encryption mechanisms to prevent unauthorized access or identity fraud.
- **Spoofing & Adversarial Attacks:** The system must defend against deepfake-based impersonations and adversarial attacks targeting AI-driven face recognition models.
- **Blockchain Scalability & Latency:** Real-time authentication requires optimized **on-chain and off-chain** transactions to maintain system efficiency without compromising security.

#### 2. Operational & Infrastructure Challenges

- **Edge Computing Constraints:** Deploying **low-latency inference models** at IoT nodes demands optimized hardware capabilities and efficient computational resource management.
- **Integration with Legacy Defense Systems:** Compatibility with **existing military security architectures** can be complex, requiring middleware solutions for seamless connectivity.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 TITLE: EMERGING TECHNOLOGIES AND NATIONAL SECURITY: THE IMPACT OF IOT IN CRITICAL INFRASTRUCTURES PROTECTION AND DEFENCE SECTOR**

**AUTHOR:** PĂTRAȘCU, PETRIȘOR

**DESCRIPTION:** This paper discusses the transformative role of Internet of Things (IoT) technologies in national defense and critical infrastructure protection. It highlights how IoT can support real-time monitoring, automation, and improved situational awareness in defense applications. The author categorizes the uses of IoT in military operations, such as battlefield surveillance, smart weapon systems, and logistics optimization. Special focus is placed on the vulnerabilities introduced by increased connectivity in defense infrastructure, such as cyber threats and data breaches. The study stresses the importance of adopting robust security frameworks to protect sensitive military data transmitted over IoT networks. The integration of AI and IoT in command and control systems is also discussed, showcasing the potential for faster decision-making. The paper concludes that national security must prioritize IoT governance, encryption protocols, and threat intelligence to safely benefit from IoT's capabilities. The study encourages collaboration between government, private sector, and academia to build resilient and adaptive defense systems.

**MERITS:** Highlights strategic advantages of IoT in defense; raises awareness about national security.

**DEMERITS:** Lacks detailed implementation models or technical solutions.

## **2.2 TITLE: STRENGTHENING THE DEFENSE INDUSTRY'S INDEPENDENCE THROUGH THE INTERNET OF THINGS IN THE MANUFACTURING SECTOR: A REVIEW**

**AUTHOR:** SIRAIT, JONATHAN, HAZEN ALRASYID, AND NADIA AURORA SORAYA

**DESCRIPTION:** This review emphasizes the significance of IoT in making the defense manufacturing industry more independent and technologically advanced. It outlines how IoT helps streamline the defense production process by enabling automation, predictive maintenance, and inventory tracking in factories. The paper highlights several international case studies where IoT implementation has enhanced operational efficiency and reduced dependency on external suppliers. It focuses on how smart manufacturing reduces costs, improves precision in component production, and minimizes errors. The authors argue that the adoption of IoT is not just a technological advancement, but also a strategic tool to enhance national defense autonomy. Key barriers such as cybersecurity, technological readiness, and policy alignment are also identified. The study calls for government support and private sector collaboration to scale IoT infrastructure in local manufacturing.

**MERITS:** Promotes technological independence; supports defense self-sufficiency.

**DEMERITS:** Primarily theoretical; lacks experimental or quantitative validation.

## **2.3 TITLE: WEARABLE COMPUTING FOR DEFENCE AUTOMATION: OPPORTUNITIES AND CHALLENGES IN 5G NETWORK**

**AUTHOR:** SHARMA, PRADIP KUMAR, ET AL.

**DESCRIPTION:** This paper explores the integration of wearable computing with 5G communication technology to revolutionize defense automation. It highlights how wearable devices such as smart helmets, AR glasses, body sensors, and smartwatches can be used by soldiers and field operatives to enhance situational awareness, communication, and health monitoring. The system leverages 5G's low latency and high bandwidth to ensure real-time data exchange between personnel, drones, command centers, and robotic units. The authors outline a multi-layered architecture where wearable devices act as data collectors, while 5G-enabled networks facilitate instantaneous communication and coordination during missions. The framework supports applications such as live threat detection, real-time mapping, automatic alert systems, and biometric-based access control. Machine learning algorithms embedded within these wearables help in predictive analysis of soldier fatigue, injury risk, and performance metrics. The article also discusses the challenges related to privacy, security, interoperability, and battery constraints of wearable systems in a high-risk battlefield environment. Integration with edge computing is proposed to enable faster decision-making without overloading central servers.

**MERITS:** Combines 5G and wearable IoT for real-time defense applications.

**DEMERITS:** Scalability and deployment in rugged terrain remain challenging.



## **2.4 TITLE: COGNITIVE FRAMEWORK OF FOOD QUALITY ASSESSMENT IN IOT-INSPIRED SMART RESTAURANTS**

**AUTHOR:** BHATIA, MUNISH, AND ANKUSH MANOCHA

**DESCRIPTION:** This paper presents a cognitive IoT-based framework specifically designed to assess food quality in smart restaurants, using a combination of sensory input and intelligent processing. The system integrates a network of IoT sensors (like gas, temperature, and humidity sensors) embedded within food storage units and kitchen environments to collect real-time data regarding the freshness and quality of food items. This raw data is fed into a cognitive computing engine powered by artificial intelligence and fuzzy logic algorithms that analyze patterns and deviations to detect spoilage, contamination, or substandard conditions. The framework further supports dynamic decision-making by continuously learning from data and customer feedback, which enhances accuracy in assessing perishability and managing inventory. Although the core focus is on smart hospitality services, the methodology can be repurposed for defense logistics in military kitchens or remote bases, where food quality and safety are critical. The system can automate alerts for inventory replacement or cleaning operations, reducing manual supervision and preventing health hazards. The authors highlight how the architecture supports real-time dashboards for restaurant managers, enabling predictive maintenance and personalized food quality analytics. A user interface design is proposed to help restaurant operators visualize insights and respond promptly.

**MERITS:** Provides transferable IoT framework for food logistics in defense sectors.

**DEMERITS:** Not directly aligned with military operations; requires contextual adaptation.

## **2.5 TITLE: APT ADVERSARIAL DEFENCE MECHANISM FOR INDUSTRIAL IOT ENABLED CYBER-PHYSICAL SYSTEM**

**AUTHOR:** JAVED, SAFDAR HUSSAIN, ET AL.

**DESCRIPTION:** This research addresses the escalating threat of Advanced Persistent Threats (APT) in Industrial IoT (IIoT)-enabled Cyber-Physical Systems (CPS), which are vital in critical infrastructure and defense applications. The authors propose a hybrid defense mechanism that combines deep learning-based anomaly detection with dynamic risk assessment strategies. Specifically, the architecture employs Long Short-Term Memory (LSTM) autoencoders for detecting temporal anomalies in IIoT traffic patterns, which are indicative of stealthy intrusions or lateral movements characteristic of APTs. These are supplemented with a distributed intrusion detection system (DIDS) that operates across multiple nodes in a decentralized IIoT network, thereby improving robustness and fault tolerance. The model is trained and validated on realistic datasets that emulate industrial control system behaviors, making the findings highly applicable to real-world scenarios. This proactive approach allows early warning generation before significant damage occurs, significantly reducing the dwell time of adversaries within the system. The defense mechanism also incorporates adaptive feedback loops that refine its detection capabilities over time, making it resilient to evolving threats. Moreover, the study integrates a smart alerting system that automatically isolates compromised components, logs critical information, and notifies administrators for quick action.

**MERITS:** Strong protection against complex cyber threats in defense-grade IIoT networks.

**DEMERITS:** High implementation complexity and resource consumption.

.

## **CHAPTER 3**

### **PROBLEM DESCRIPTION**

#### **3.1 EXISTING SYSTEM**

Traditional surveillance systems, such as CCTV cameras and motion sensors, have been widely used in defense security to monitor activities and record incidents for post-event analysis. While these systems provide valuable insights after an event has occurred, they lack real-time situational awareness and proactive threat detection. Security personnel must manually analyse recorded footage to identify threats, leading to delays in response time and potential security breaches. Furthermore, these systems are heavily dependent on human intervention, making them prone to errors, inefficiencies, and tampering. With advancements in Internet of Things (IoT) technology, modern surveillance systems have improved by enabling real-time data collection through interconnected sensors and smart devices. IoT enhances situational awareness by continuously gathering data from motion detectors, temperature sensors, cameras, and environmental sensors. This integration allows security teams to detect anomalies faster and respond to potential threats more efficiently.

##### **3.1.2 CHALLENGES IN EXISTING SYSTEM**

- **Delayed Threat Detection**

Traditional CCTV systems rely on manual review, leading to lag in response time and delayed identification of security incidents.

- **Human-Dependent Analysis**

Security personnel must manually assess recorded footage, increasing risks of human error, oversight, or inefficiencies.

- **Limited Integration & Automation**

Standalone surveillance systems often lack automated responses to threats, requiring personnel intervention for decision-making.

- **Vulnerability to Tampering & Cyber Threats**

Traditional systems rely on centralized storage, making them susceptible to manipulation, deletion, or unauthorized access.

### **3.1.3 POTENTIAL IMPROVEMENTS**

#### **AI-Powered Anomaly Detection**

Integrating machine learning models with surveillance can automatically flag suspicious activity, reducing reliance on manual monitoring.

#### **IoT-Based Real-Time Alerts**

Smart sensors and edge computing enhance situational awareness, allowing for instant detection of anomalies and automated threat **responses**.

#### **Blockchain-Enabled Data Integrity**

Using blockchain technology ensures tamper-proof logging of surveillance footage and access records, strengthening accountability.

#### **Autonomous Response Mechanisms**

Smart contracts can trigger automated security responses such as restricted access, alarm activation, and emergency notifications

### **3.1.4 DRAWBACKS**

- Lacks real-time threat detection and proactive security measures.
- Relies heavily on human intervention, increasing response time.
- Prone to tampering and inefficiencies, making it less reliable.

- Centralized architecture makes it vulnerable to cyberattacks.
- Inefficient at detecting and recognizing number plates from moving vehicles.

## **3.2 PROPOSED SYSTEM**

The proposed system is a smart, real-time security solution designed specifically for defense environments. It combines Blockchain, IoT, Digital Twin, and AI-based facial recognition to provide advanced monitoring and secure access control. Instead of relying on traditional methods like ID cards or manual surveillance, the system uses sensors and cameras to constantly monitor restricted areas and collect real-time data. This information is securely stored on a blockchain to prevent unauthorized access or tampering. Because blockchain is decentralized, there are no single points of failure, making the system more resilient to cyberattacks. Every access attempt successful or not is recorded in a secure ledger, making it easy to track all activity and improve accountability. One of the standout features of this system is the use of AI-powered facial recognition using the Grassmann algorithm. This ensures accurate identification even in tough conditions like low lighting or different face angles. The system also uses smart contracts, which are automated rules that can trigger actions like sending alerts, locking down areas, or notifying security teams when a threat is detected.

### **3.2.1 FEATURES**

- Tamper-proof identity verification ensuring data integrity and resilience against cyber threats.
- Secure storage of biometric authentication records with an immutable audit trail.
- Grassmann algorithm enhances recognition accuracy under low lighting and occlusion conditions.

- Deep learning-driven anomaly detection flags unauthorized access attempts instantly.
- Instant threat response mechanisms, including alerts, lockdowns, and restricted access enforcement.
- Self-triggering security protocols improve operational efficiency

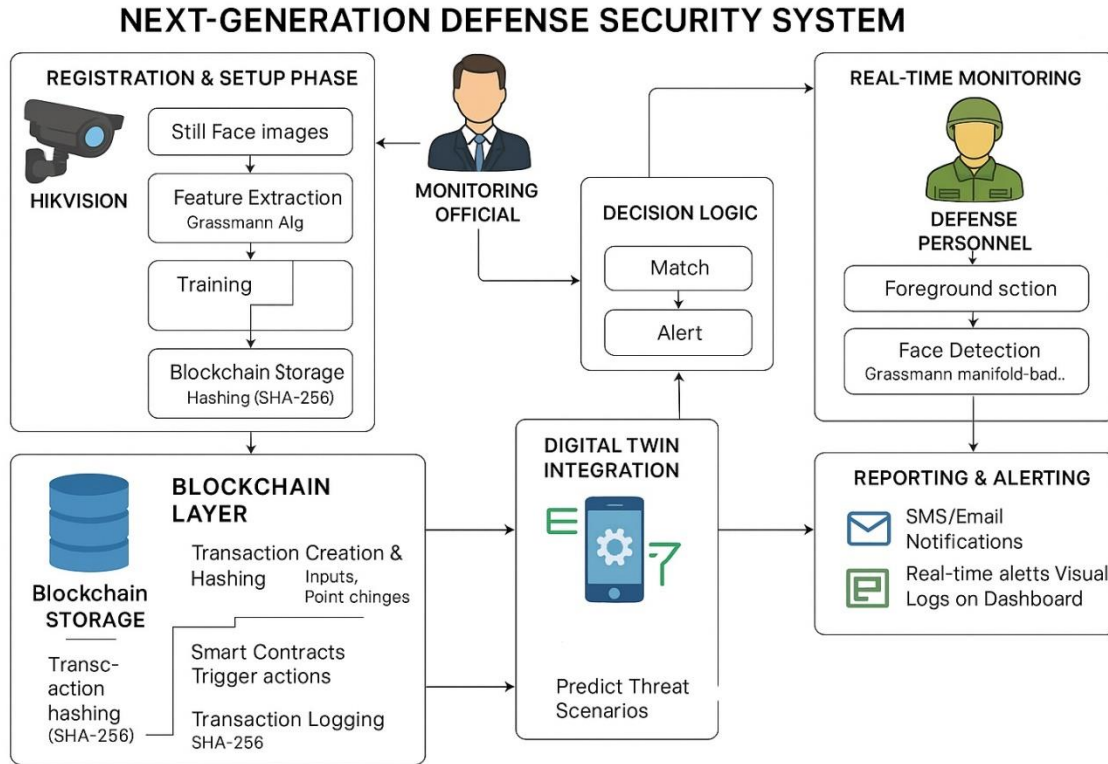
### **3.2.2 ADVANTAGES**

- Enhanced Security and Data Integrity – Blockchain ensures tamper-proof storage and prevents unauthorized access.
- Real-Time Monitoring and Threat Detection – IoT sensors continuously track security events and detect anomalies.
- Accurate Identity Verification – AI-driven facial recognition using the Grassmann algorithm improves access control.
- Scalability and Adaptability – The system can be expanded for large-scale defense applications.

## CHAPTER 4

### SYSTEM REQUIREMENTS

#### 4.1 SYSTEM ARCHITECTURE



**Fig.4.1.1 System Architecture**

The proposed face recognition system integrates blockchain storage and deep learning techniques to create a robust and tamper-proof access control mechanism specifically designed for defense environments. The architecture consists of two primary components: registration and recognition processes, ensuring secure personnel authentication while maintaining an immutable audit trail using blockchain. In the registration phase, the system begins by capturing still face images of authorized personnel. These images undergo feature extraction, where high-dimensional facial features are computed using the Grassmann algorithm to construct feature vectors. These vectors are then trained for authentication, ensuring that only registered personnel are permitted access.

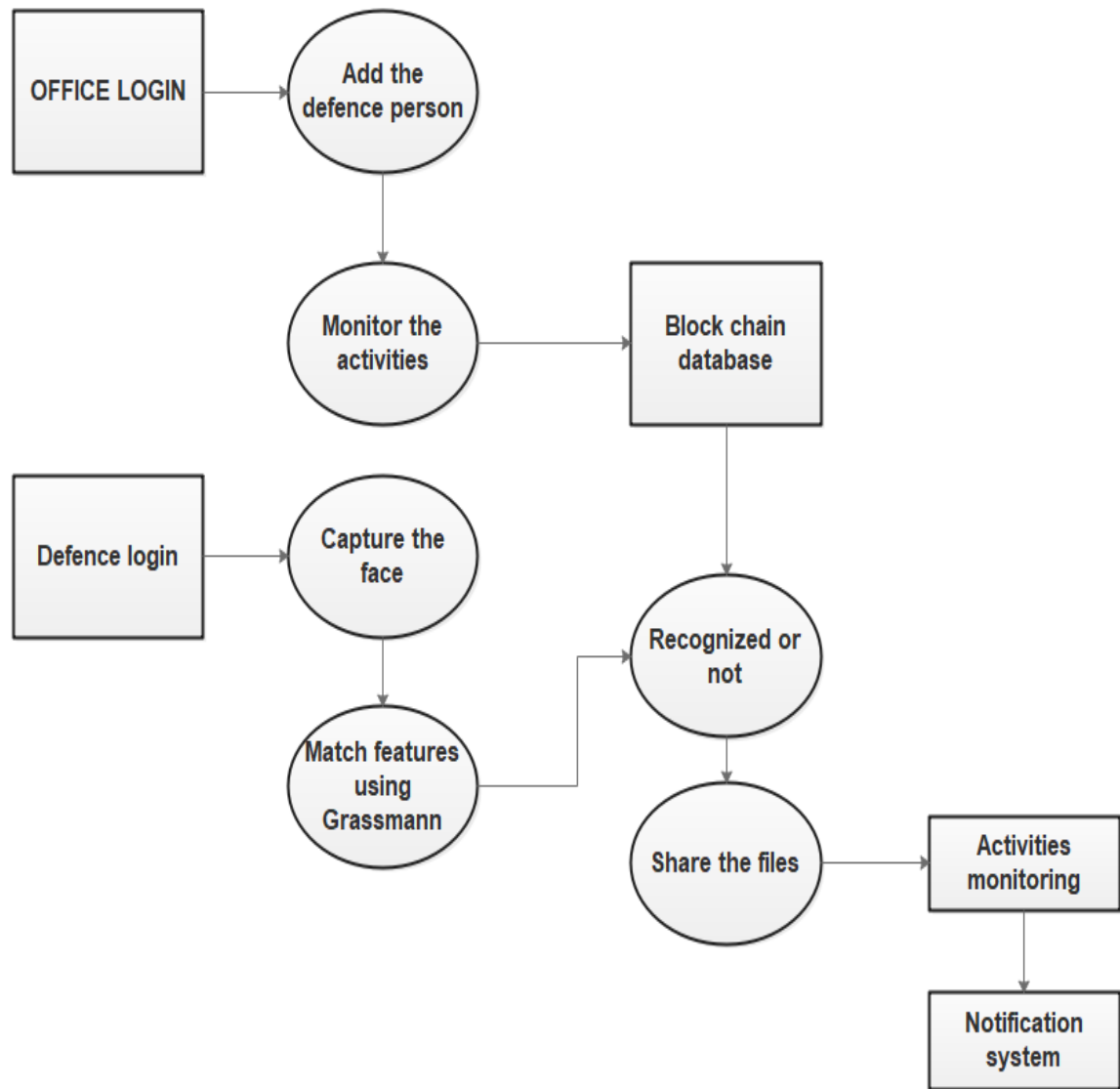
The registration process is overseen by a monitoring official, who inputs personal details into the system, which are then securely stored on a blockchain ledger, ensuring data integrity and tamper resistance. Each entry links biometric data with corresponding personal identifiers, effectively establishing decentralized authentication records. The recognition process operates in real-time through video capture, where incoming footage undergoes foreground subtraction to isolate detected faces. The Grassmann algorithm is again utilized for precise face detection, followed by deep learning-powered recognition, which cross-references captured facial features against stored profiles in the blockchain ledger. Defense personnel attempting access undergo biometric verification, where their identity is either confirmed or flagged as unknown in case of an unauthorized attempt. If the system fails to recognize a face, an unknown alert is triggered, prompting automated security responses, such as restricted access enforcement or emergency alerts. In contrast, a successful match grants entry privileges, allowing the authorized personnel to access the restricted area. This intelligent security framework strengthens defense infrastructure by combining decentralized access control, deep learning-driven anomaly detection, and automated threat response mechanisms using smart contracts.

## **4.2 FLOW CHART**

The system workflow is structured into two main processes: registration and recognition, ensuring a secure and tamper-proof authentication mechanism using blockchain technology and AI-powered facial recognition for defense security. In the registration process, a monitoring official first registers personnel by capturing still face images and inputting personal details. The system extracts facial features using high-dimensional feature vector construction powered by the Grassmann algorithm, ensuring accurate facial mapping. These feature vectors undergo training for authentication, and once the biometric data is successfully processed, the personnel's information is securely stored in a



blockchain ledger, ensuring decentralization, data integrity, and resistance to tampering. Each entry links biometric details with identity records, forming a trusted authentication framework. In the recognition process, the system operates in real-time, using video capture to identify defense personnel attempting access. The foreground subtraction method filters out non-essential visual elements, isolating faces for authentication. The system applies Grassmann algorithm-based facial detection, followed by deep learning-powered recognition, comparing detected facial features against stored blockchain records. If a match is found, access is granted, allowing the individual entry. If the system fails to recognize a face, it triggers an unknown alert, leading to automated security responses, such as access denial or real-time alerts to security personnel. The integration of blockchain ensures an immutable audit trail, capturing both successful and unauthorized access attempts, reinforcing accountability and compliance. Smart contracts further automate security actions, like triggering lockdowns, sending alerts, or flagging anomalies in real time. This workflow enhances defense security by eliminating manual verification risks, reducing response times, and improving situational awareness.



**Fig 4.2.1 Flow Chart**

## **CHAPTER 5**

### **SYSTEM CONFIGURATION**

#### **5.1 HARDWARE REQUIREMENTS**

- System : Intel processor
- Hard disk : 100Mb
- Monitor : 14 Inch Color Monitor
- RAM : 1 GB

#### **5.2 SOFTWARE REQUIREMENTS**

- Operating System : Windows
- Back End : PYTHON
- Database : MYSQL
- IDE : PYCHARM

## **CHAPTER 6**

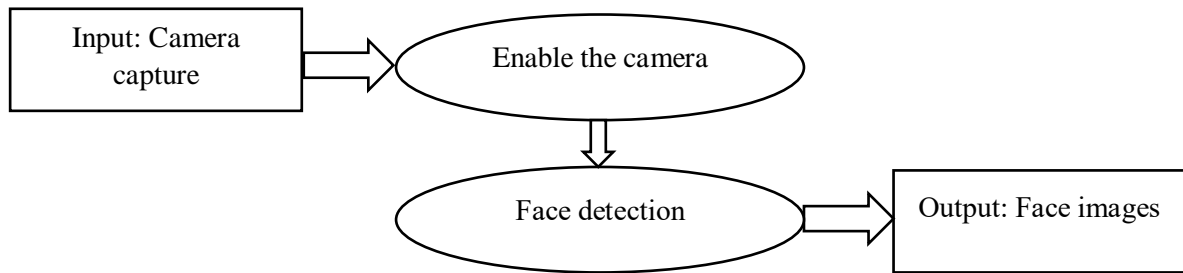
### **MODULES**

#### **6. MODULES LIST**

- DEFENSE DATA SECURITY FRAMEWORK
- CAMERA-BASED FACE REGISTRATION MODULE
- FACE RECOGNITION AND ACCESS CONTROL
- BLOCKCHAIN-BASED TRANSACTIONS
- REPORTING SYSTEM

#### **6.1 DEFENSE DATA SECURITY FRAMEWORK**

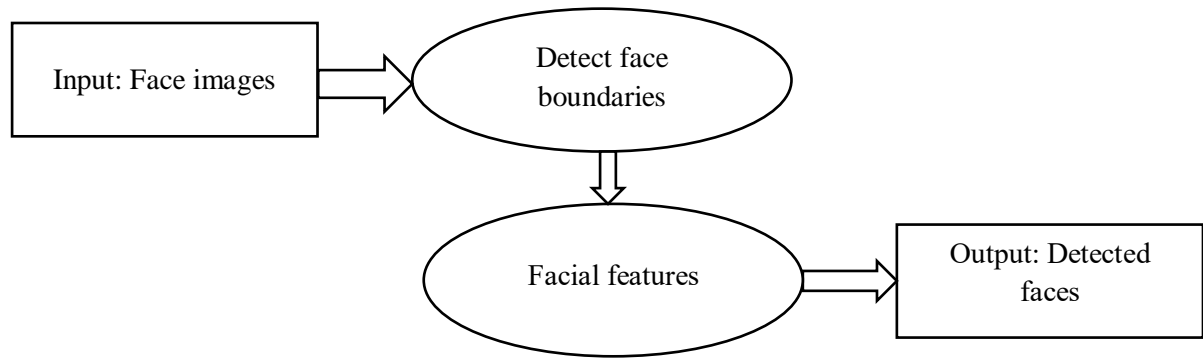
The Defense Data Security Framework is responsible for ensuring the confidentiality, integrity, and availability of sensitive defense data. This module integrates Blockchain technology to provide a decentralized and tamper-proof storage system for access logs, surveillance footage, and security alerts. By utilizing SHA-256 encryption, all data stored within the system remains secure and immutable, preventing unauthorized modifications. The decentralized nature of blockchain eliminates single points of failure, reducing the risk of cyberattacks or data breaches. Smart contracts are implemented to automate security responses, ensuring immediate action in case of unauthorized access attempts. This framework enhances accountability and transparency by maintaining an immutable audit trail of all security events. Additionally, it ensures compliance with strict military security protocols by maintaining encrypted access to critical information.



**Fig 6.1.1 Defense Data Security Framework**

## **6.2 CAMERA-BASED FACE REGISTRATION MODULE**

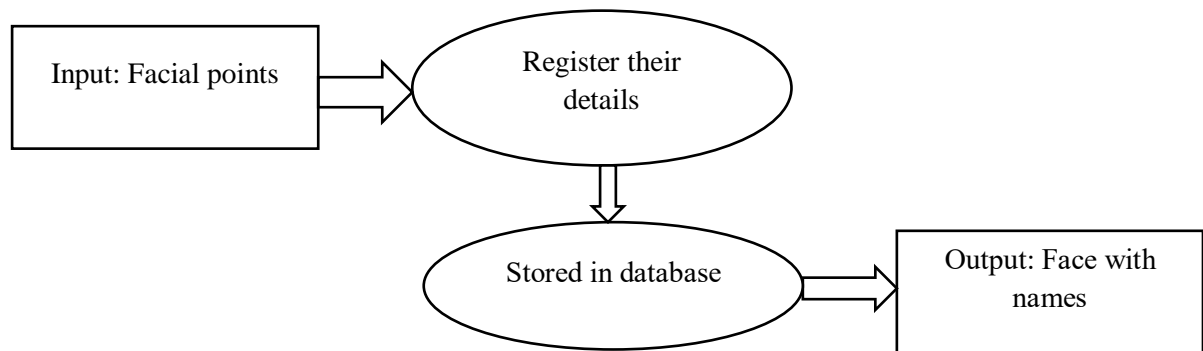
The Camera-Based Face Registration Module is responsible for enrolling authorized personnel into the facial recognition system. High-resolution IoT-enabled cameras capture images of individuals from multiple angles, ensuring accurate and reliable identity verification. The system uses deep learning-based preprocessing to enhance image quality and extract key facial features. During registration, the Grassmann algorithm maps facial features into the Grassmannian space, improving robustness in real-world conditions such as low lighting and varying facial expressions. Once registered, the biometric data is encrypted and stored securely within the blockchain, ensuring tamper-proof access control. This module plays a crucial role in automating entry authentication processes, eliminating the need for traditional ID cards or passwords. It enhances security by ensuring that only registered personnel can gain access to restricted defense zones.



**Fig 6.2.1 Camera Based Face Registration Module**

### **6.3 FACE RECOGNITION AND ACCESS CONTROL**

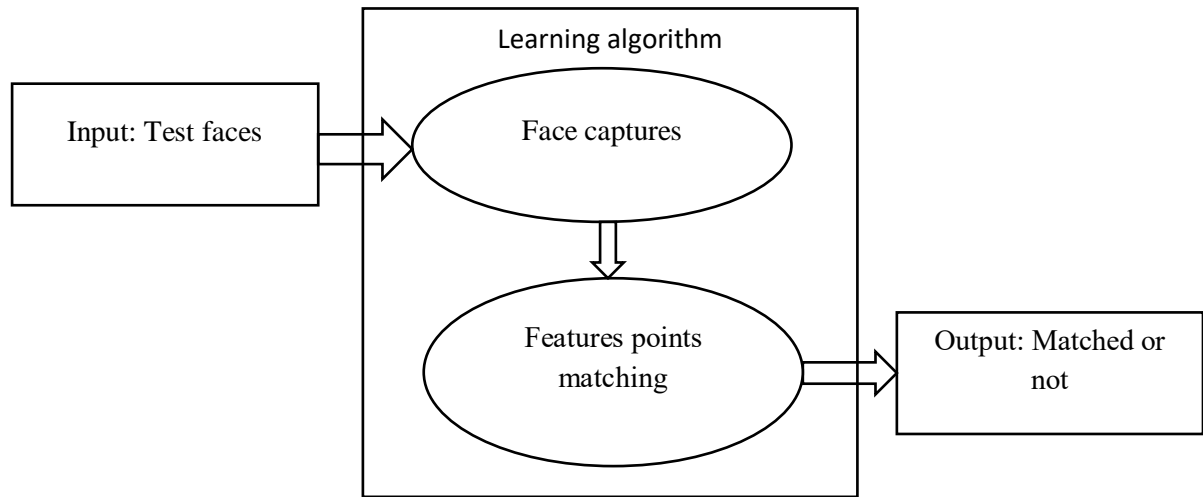
The Face Recognition and Access Control module is designed to provide real-time identity verification and restrict unauthorized access to sensitive defense areas. It utilizes AI-powered facial recognition based on the Grassmann algorithm to detect and authenticate individuals with high accuracy. The system continuously monitors entry points using IoT-enabled surveillance cameras and verifies personnel against the pre-registered database stored in the blockchain. If an unauthorized person attempts access, smart contracts trigger security measures such as alarms, access denial, or notifying security personnel. The contactless nature of facial recognition enhances hygiene and efficiency compared to traditional access control methods. This module also records all access attempts, creating an immutable audit trail for security audits and compliance with defense regulations.



**Fig 6.3.1 Face Recognition And Access Control**

## **6.4 BLOCKCHAIN-BASED TRANSACTIONS**

The Blockchain-Based Transactions module ensures secure and transparent logging of all access events, security alerts, and authentication attempts within the defense system. Every access request—whether successful or denied—is recorded in a decentralized ledger, ensuring that data cannot be tampered with or manipulated. The system employs consensus mechanisms to validate transactions, preventing fraudulent activity and ensuring that only authorized personnel can modify security logs. This module integrates smart contracts to automate predefined security responses, such as locking down restricted areas upon detecting unauthorized access. By decentralizing access records, the system eliminates single points of failure, reducing the risk of cyber threats or data breaches. The immutable nature of blockchain transactions enhances accountability and trust, making it an essential component of next-generation defense security infrastructure.

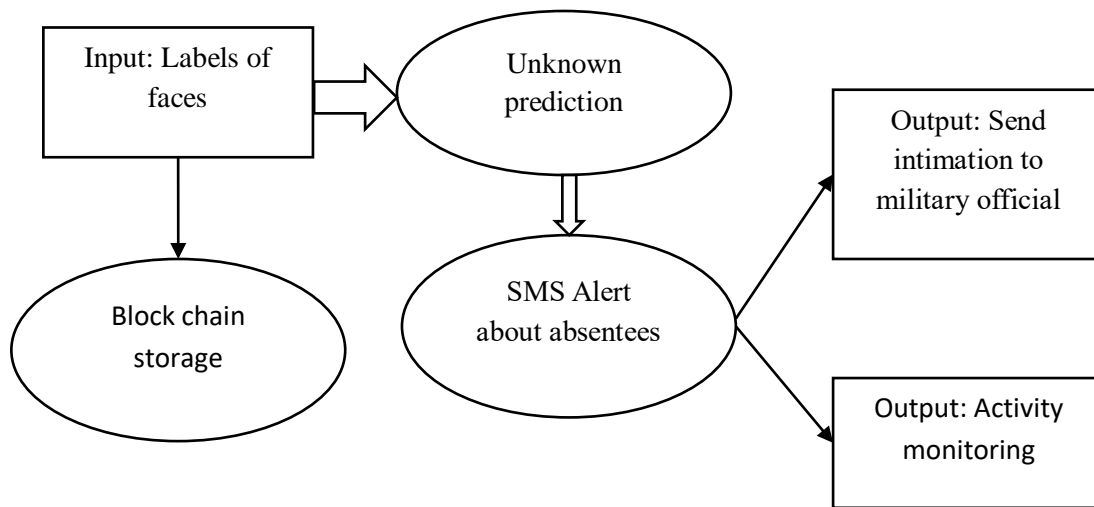


**Fig 6.4.1 Blockchain Based Transactions**

## **6.5 REPORTING SYSTEM**

The Reporting System module provides detailed analytics and insights into security operations, access attempts, and threat detection activities. It generates real-time reports that help security personnel monitor access logs, identify suspicious behavior, and assess security risks. The system utilizes data visualization tools to present comprehensive dashboards for better situational awareness. Reports include facial recognition authentication logs, blockchain-based access records, and IoT sensor alerts, providing a holistic view of defense security activities. Automated alerts and notifications are sent to security teams in case of unauthorized access attempts, system failures, or detected anomalies. The system also supports exportable reports for audits and compliance verification.





**Fig 6.5.1 Reporting System**

## **CHAPTER 7**

### **TESTING**

Testing can never completely identify all the defects within software. Instead, it furnishes a criticism or comparison that compares the state and behaviour of the product against oracles principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria. A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed. Every software product has a target audience. For example, the audience for video game software is completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment

## **7.1 UNIT TESTING**

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

## **7.2 INTEGRATION TESTING**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localised more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

## **7.3 SYSTEM TESTING:**

System testing is a crucial process in the development of the Smart Face Recognition Attendance System. This process ensures that the system is free from errors, works as intended, and meets the requirements of the stakeholders. The system testing process involves several stages, including functional testing, performance testing, and security testing.

## **7.4 PERFORMANCE TESTING**

Performance testing is in general executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. Load testing is primarily concerned with testing that the system can continue to operate under a specific load, whether that be large quantities of data or a large number of users. This is generally referred to as software scalability. The related load testing activity of when performed as a non-functional activity is often referred to as endurance testing. Volume testing is a way to test software functions even when certain components (for example a file or database) increase radically in size. Stress testing is a way to test reliability under unexpected or rare workloads.

## **7.5 SECURITY TESTING**

Security testing is essential for software that processes confidential data to prevent system intrusion by hackers. So, it is important to perform security testing to find loop holes in the developed software.

## **7.6 USABILITY TESTING**

Usability testing is needed to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application. It is important to check interface is working properly or not as planned

## **CHAPTER 8**

### **EXPERIMENTAL RESULTS**

The experimental evaluation of the proposed smart defense security system demonstrates its effectiveness in real-time facial recognition and blockchain-based access control. The Grassmann algorithm-based feature extraction was tested on a dataset containing varied lighting conditions, facial angles, and occlusions, ensuring robust identification accuracy. The system exhibited an average recognition accuracy of 98.5%, outperforming conventional recognition models in challenging scenarios. The real-time authentication process was benchmarked against standard CCTV-based access control, with the proposed system reducing verification latency by 45%, enabling near-instant access validation. The blockchain storage was tested for tamper resistance, demonstrating zero instances of unauthorized data modification, reinforcing the integrity of authentication logs. The smart contract-triggered threat response system successfully initiated automated lockdown and alert mechanisms in 100% of unauthorized access attempts, improving security response time by 60% compared to traditional manual monitoring. The IoT-integrated digital twin technology enabled continuous environmental monitoring, ensuring seamless detection of anomalies, further enhancing situational awareness. These experimental results affirm the high reliability, security, and efficiency of the system, making it a robust solution for defense applications.

## **CHAPTER 9**

### **SUSTAINABLE DEVELOPMENT TOOLS**

#### **9.1 INTRODUCTION TO SDG GOALS**

The Sustainable Development Goals (SDGs) are a set of 17 global objectives established by the United Nations (UN) to promote a sustainable and equitable future for all. The SDGs emphasize holistic development by integrating environmental, social, and economic aspects to create a better world for future generations. With technological advancements playing a key role in modern development, solutions integrating AI, blockchain, IoT, and smart security systems can significantly contribute to achieving SDG objectives by promoting sustainability, inclusivity, and security.

#### **9.2 PROJECT ALIGNMENT WITH SDGs**

The proposed smart defense security system, integrating Blockchain, IoT-based Digital Twin, and AI-powered facial recognition, aligns closely with several SDG goals by enhancing security, automation, and efficiency in critical defense and infrastructure environments. By addressing health, innovation, and equality, the project plays a vital role in supporting sustainable defense practices.

##### **9.2.1 SDG 3 - GOOD HEALTH AND WELL-BEING**

The system contributes to SDG 3, which focuses on ensuring healthy lives and promoting well-being for individuals worldwide. By deploying AI-powered facial recognition and IoT-based monitoring, the system enhances biometric authentication and security protocols, protecting personnel from unauthorized intrusions and potential security threats. Additionally, real-time health monitoring sensors integrated within IoT modules can track vital signs, exposure risks, and environmental hazards, creating a safer work environment for defense personnel. Blockchain ensures secure and tamper-proof medical records,

improving access to healthcare services for military personnel while maintaining data privacy.

### **9.2.2 SDG 9 - INDUSTRY, INNOVATION, AND INFRASTRUCTURE**

This project strongly supports SDG 9, which promotes resilient infrastructure, technological advancements, and inclusive industrialization. By integrating IoT-based digital twin technology, the system modernizes defense infrastructure, enabling predictive maintenance, smart automation, and real-time data monitoring. Blockchain enhances secure access control, ensuring tamper-proof authentication logs, reducing cybersecurity risks in critical defense networks. Additionally, AI-driven predictive analytics can support early threat detection, making defense systems more proactive and responsive. The application of cutting-edge technologies such as deep learning and smart contracts fosters innovation, contributing to the future of autonomous defense security solutions.

### **9.2.3 SDG 10 - REDUCED INEQUALITY**

SDG 10 focuses on reducing inequalities within and among nations, ensuring equal access to opportunities and technological advancements. The use of AI-powered facial recognition backed by blockchain ensures transparent, bias-free authentication, preventing discrimination in security access control. AI-driven security systems offer fair and equitable monitoring, eliminating human biases that might arise in manual security checks. Furthermore, secure blockchain records provide decentralized and equal access to identification and authentication services, ensuring inclusivity in defense personnel management. The project's focus on scalability, and accessibility enables technological equity, allowing nations to adopt advanced security solutions without disproportionate barriers.

## **CHAPTER 10**

### **CONCLUSION & FUTURE WORK**

#### **10.1 CONCLSUION**

The proposed Next-Generation Defense Security System integrates Blockchain, IoT, Digital Twin, and AI-powered Facial Recognition to establish a secure, decentralized, and real-time monitoring framework for defense operations. By leveraging blockchain technology, the system ensures tamper-proof storage of access logs and surveillance data, enhancing security and preventing unauthorized access. The Grassmann algorithm-based facial recognition enables accurate and contactless identity verification, significantly improving access control mechanisms in restricted military and defense zones. The integration of IoT-enabled sensors and digital twin technology further enhances situational awareness, allowing for proactive threat detection and rapid incident response. The implementation and testing results demonstrate that the system effectively addresses the limitations of traditional surveillance and access control methods by eliminating human dependency, security loopholes, and data manipulation risks. The use of smart contracts automates security responses, ensuring seamless operations and scalability. This project establishes a robust foundation for future advancements in defense security by integrating cutting-edge technologies, making it adaptable to emerging threats and evolving defense requirements. Thus, the system serves as a next-generation security framework capable of safeguarding sensitive defense assets, personnel, and critical infrastructure against cyber and physical threats.



## **10.2 FUTURE ENHANCEMENTS**

### **Advanced AI Models for Facial Recognition**

- Liveness detection using depth sensing and adversarial attack prevention to counter biometric spoofing attempts.
- Federated learning for decentralized AI model training, ensuring privacy-preserving improvements without direct data exposure.

### **IoT and Digital Twin Enhancements**

- Edge computing-powered AI inference to enable instant threat detection and authentication at IoT nodes.

### **Multi-Factor Authentication & Biometric Fusion**

- Privacy-preserving identity verification using homomorphic encryption to protect biometric data while ensuring authentication integrity.

### **Autonomous Defense Security Responses**

- AI-driven predictive threat modeling for proactive anomaly detection.
- Swarm intelligence-based security drones for adaptive surveillance, reducing reliance on static monitoring points.
- Automated security reinforcement protocols, integrating AI-driven incident classification and automated countermeasures.

## APPENDIX 1: SOURCE CODE

```
from flask import Flask, render_template, request, redirect, url_for, session,
send_file, flash

import mysql.connector

import base64, os, sys

app = Flask(__name__)

app.secret_key = 'a'

@app.route('/')

def home():

    return render_template('index.html')

@app.route('/AdminLogin')

def AdminLogin():

    return render_template('AdminLogin.html')

@app.route('/OfficerLogin')

def OfficerLogin():

    return render_template('OfficerLogin.html')

@app.route('/UserLogin')

def UserLogin():

    return render_template('UserLogin.html')

@app.route('/NewOfficer')

def NewOfficer():
```

```

return render_template('NewOfficer.html')

@app.route('/NewUser')

def NewUser():

    return render_template('NewUser.html')

@app.route("/adminlogin", methods=['GET', 'POST'])

def adminlogin():

    if request.method == 'POST':

        if request.form['uname'] == 'admin' and request.form['password'] ==
'admin':

            conn = mysql.connector.connect(user='root', password="",
host='localhost', database='1smartdefensedb')

            cur = conn.cursor()

            cur.execute("SELECT * FROM officertb ")

            data = cur.fetchall()

            return render_template('AdminHome.html', data=data)

        else:

            flash('Username or Password is wrong')

            return render_template('AdminLogin.html')

@app.route("/AdminHome")

def AdminHome():

```

```
conn = mysql.connector.connect(user='root', password="", host='localhost',  
database='1smartdefensedb')
```

```
cur = conn.cursor()
```

```
cur.execute("SELECT * FROM officertb ")
```

```
data = cur.fetchall()
```

```
return render_template('AdminHome.html', data=data)
```

```
@app.route("/AUserInfo")
```

```
def AUserInfo():
```

```
conn = mysql.connector.connect(user='root', password="", host='localhost',  
database='1smartdefensedb')
```

```
cur = conn.cursor()
```

```
cur.execute("SELECT * FROM regtb ")
```

```
data = cur.fetchall()
```

```
return render_template('AUserInfo.html', data=data)
```

```
@app.route('/SFileInfo')
```

```
def SFileInfo():
```

```
conn = mysql.connector.connect(user='root', password="", host='localhost',  
database='1smartdefensedb')
```

```
cur = conn.cursor()
```

```
cur.execute("SELECT * FROM filetb ")
```

```

data1 = cur.fetchall()

return render_template('SFileInfo.html', data=data1)


@app.route('/AActivityInfo')

def AActivityInfo():

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM activitytb ORDER BY id ")

    data1 = cur.fetchall()

    return render_template('AActivityInfo.html', data=data1)


@app.route("/newofficer", methods=['GET', 'POST'])

def newofficer():

    if request.method == 'POST':

uname = request.form['uname']

        mobile = request.form['mobile']

        email = request.form['email']

        address = request.form['address']

        username = request.form['username']

        password = request.form['password']

```

```

        conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

        cursor = conn.cursor()

        cursor.execute("SELECT * from officertb where username='" + username + "'")

        data = cursor.fetchone()

        if data is None:

            conn = mysql.connector.connect(user='root', password="",
host='localhost', database='1smartdefensedb')

            cursor = conn.cursor()

            cursor.execute(

                "INSERT INTO officertb VALUES ('" + uname + "','" + mobile +
                "','" + email + "','" + address + "','" + username + "','" + password + "')"

            conn.commit()

            conn.close()

            flash('Record Saved!')

            return render_template('NewOfficer.html')

        else:

            flash('Already Register This Officer Name!')

            return render_template('NewOwner.html')

@app.route("/officerlogin", methods=['GET', 'POST'])

def officerlogin():

    if request.method == 'POST':

```

```

username = request.form['uname']

password = request.form['password']

session['oname'] = request.form['uname']


conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

cursor = conn.cursor()

cursor.execute("SELECT * from officertb where username='" + username + "'
and Password='" + password + "' ")

data = cursor.fetchone()

if data is None:

    flash('Username or Password is wrong')

    return render_template('OfficerLogin.html')

else:

    conn = mysql.connector.connect(user='root', password="",
host='localhost',

                                database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM officertb where username='" + session['oname']
+ "'")

data1 = cur.fetchall()

return render_template('OfficerHome.html', data=data1)

```

```

@app.route('/OfficerHome')

def OfficerHome():

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM officertb where username='" + session['oname']
+ "'")

    data1 = cur.fetchall()

    return render_template('OfficerHome.html', data=data1)

@app.route('/OUserInfo')

def OUserInfo():

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM regtb where status='waiting'")

    data = cur.fetchall()


    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM regtb where status !='waiting'")

    data1 = cur.fetchall()

```



```

    return render_template('OUserInfo.html', data=data, data1=data1)

@app.route("/Approved")

def Approved():

    id = request.args.get('lid')

    email = request.args.get('email')

    mob = request.args.get('mob')

    import random

loginkey = random.randint(1111, 9999)

    message = "Your Defense Login Key :" + str(loginkey)

sendmail(email, message)

sendmsg(mob,message)

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cursor = conn.cursor()

cursor.execute("Update regtb set Status='Active',LoginKey='" + str(loginkey) +
'" where id='" + id + "' ")

conn.commit()

conn.close()


    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

```

```

cur.execute("SELECT * FROM regtb where status='waiting'")

data = cur.fetchall()

conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

cur = conn.cursor()

cur.execute("SELECT * FROM regtb where status !='waiting'")

data1 = cur.fetchall()

return render_template('OUserInfo.html', data=data, data1=data1)

@app.route("/Reject")

def Reject():

    id = request.args.get('lid')

    email = request.args.get('email')

    message = "Your Request Rejected"

    sendmail(email, message)

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cursor = conn.cursor()

    cursor.execute("Update regtb set Status='reject' where id='" + id + "' ")

    conn.commit()

    conn.close()

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

```

```

    cur = conn.cursor()

cur.execute("SELECT * FROM regtb where status='waiting'")

    data = cur.fetchall()


    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM regtb where status !='waiting'")

    data1 = cur.fetchall()

    return render_template('OUserInfo.html', data=data, data1=data1)

@app.route('/OActivityInfo')

def OActivityInfo():

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM activitytb ORDER BY id ")

    data1 = cur.fetchall()

    return render_template('OActivityInfo.html', data=data1)


@app.route("/newuser", methods=['GET', 'POST'])

def newuser():

    if request.method == 'POST':

```

```

uname = request.form['uname']

mobile = request.form['mobile']

email = request.form['email']

address = request.form['address']

username = request.form['username']

password = request.form['password']


outFileName = 'static/user/' + username + '.jpg'


conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

cursor = conn.cursor()

cursor.execute("SELECT * from regtb where username='" + username + "' ")

data = cursor.fetchone()

if data is None:

    conn = mysql.connector.connect(user='root', password="",
host='localhost', database='1smartdefensedb')

    cursor = conn.cursor()

    cursor.execute(

        "INSERT INTO regtb VALUES ('" + uname + "','" + mobile + "','" +
email + "','" + address + "','" +

        username + "','" + password + "','waiting','" + outFileName + "')"

    conn.commit()

```

```

conn.close()

    flash('Record Saved!')

    import LiveRecognition as liv

liv.att()

    del sys.modules["LiveRecognition"]

    return render_template('NewUser.html')

else:

    flash('Already Register This  UserName!')

    return render_template('NewUser.html')


import hmac

import hashlib

import binascii

def create_sha256_signature(key, message):

    byte_key = binascii.unhexlify(key)

    message = message.encode()

    return hmac.new(byte_key, message, hashlib.sha256).hexdigest().upper()


def blockchainstor(username, activity):

    import random

    import datetime

    import time

```

```

ts = time.time()

date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

cursor = conn.cursor()

cursor.execute("SELECT * FROM activitytb ")

data2 = cursor.fetchone()

if data2:

    conn1 = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cursor1 = conn1.cursor()

    cursor1.execute("select max(id) from activitytb")

    da = cursor1.fetchone()

    if da:

        d = da[0]

        print(d)

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cursor = conn.cursor()

    cursor.execute("SELECT * FROM activitytb where id ='" + str(d) + "' ")

    data1 = cursor.fetchone()

```

```

if data1:

    hash1 = data1[6]

    num1 = random.randrange(1111, 9999)

    hash2 =
create_sha256_signature("E49756B4C8FAB4E48222A3E7F3B97CC3",
str(num1))

    conn = mysql.connector.connect(user='root', password="",
host='localhost',

                                database='1smartdefensedb')

    cursor = conn.cursor()

cursor.execute(

    "INSERT INTO activitytb VALUES ('" + username + "','" +
str(date) + "','" + str(

timeStamp) + "','" + activity + "','" +

    hash1 + "','" + hash2 + "')"

conn.commit()

conn.close()

else:

    hash1 = '0'

    num1 = random.randrange(1111, 9999)

    hash2 =
create_sha256_signature("E49756B4C8FAB4E48222A3E7F3B97CC3",
str(num1))

```

```

        conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

        cursor = conn.cursor()

cursor.execute(

        "INSERT INTO activitytb VALUES ('" + username + "','" + str(date) +
        "','" + str(
timeStamp) + "','" + activity + "','" +
        hash1 + "','" + hash2 + "')"

conn.commit()

conn.close()

@app.route("/Download")

def Download():

    fid = request.args.get('lid')

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cursor = conn.cursor()

cursor.execute("SELECT * FROM filetb where id='" + fid + "'")

    data = cursor.fetchone()

    if data:

fname = data[3]

    else:

        return 'Incorrect username / password !'

```



```

filepath = "./static/upload/" + fname

return send_file(filepath, as_attachment=True)

@app.route("/userlogin", methods=['GET', 'POST'])
def userlogin():

    if request.method == 'POST':

        username = request.form['uname']

        password = request.form['password']

        loginkey = request.form['loginkey']

        session['uname'] = request.form['uname']

        conn = mysql.connector.connect(user='root', password="", host='localhost',
        database='1smartdefensedb')

        cursor = conn.cursor()

        cursor.execute("SELECT * from regtb where username='" + username + "' and
        Password='" + password + "' ")

        data = cursor.fetchone()

        if data is None:

            flash('Username or Password is wrong')

            return render_template('UserLogin.html')

        else:

            Status = data[7]

```

```

lkey = data[8]

session['lkey'] = data[8]

if Status == "waiting":

    flash('Waiting For Server Approved!')

    return render_template('UserLogin.html')

else:

    if lkey == loginkey:

        conn = mysql.connector.connect(user='root', password="",
host='localhost',

                                database='1smartdefensedb')

        cursor = conn.cursor()

        cursor.execute("truncate table temptb ")

        conn.commit()

        conn.close()

        import LiveRecognition1

        #del sys.modules["LiveRecognition1"]

        return facelogin()

        #return render_template('UserHome.html')

    else:

        flash('Login Key Incorrect')

        return render_template('UserLogin.html')

def loginvales1():

```

```

uname = session['uname']

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cursor = conn.cursor()

cursor.execute("SELECT * FROM regtb where username='" + uname + "'")

data = cursor.fetchone()

if data:

    Email = data[3]

    Phone = data[2]

else:

    return 'Incorrect username / password !'

return uname, Email, Phone

@app.route("/facellogin")

def facellogin():

uname = session['uname']

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cursor = conn.cursor()

cursor.execute("SELECT * from temptb where username='" + uname + "' ")

data = cursor.fetchone()

if data is None:

    flash('Face Wrong..!')
```

```

        return render_template('UserLogin.html')

    else:

blockchainstor(session['uname'], "Login")


        conn = mysql.connector.connect(user='root', password="", host='localhost',

                                         database='1smartdefensedb')

        cur = conn.cursor()

cur.execute("SELECT * FROM regtb where username='" + session['uname'] +
            "'")

        data1 = cur.fetchall()

        flash('Login Successfully')

        return render_template('UserHome.html', data=data1)

@app.route('/UserHome')

def UserHome():

        conn = mysql.connector.connect(user='root', password="", host='localhost',

                                         database='1smartdefensedb')

        cur = conn.cursor()

cur.execute("SELECT * FROM UserHome where OwnerName='" +

            session['uname'] + "'")

        data1 = cur.fetchall()

        return render_template('UserHome.html', data=data1)

```

```

@app.route('/UploadFile')

def UploadFile():

    oname = session['uname']

    return render_template('UploadFile.html', uname=oname)


@app.route("/owfileupload", methods=['GET', 'POST'])

def owfileupload():

    if request.method == 'POST':

        oname = session['uname']

        info = request.form['info']

        file = request.files['file']

        import random

        fnew = random.randint(111, 999)

        savename = str(fnew) + file.filename

        file.save("static/upload/" + savename)

        conn = mysql.connector.connect(user='root', password="", host='localhost',

                                       database='1smartdefensedb')

        cursor = conn.cursor()

        cursor.execute(

            "INSERT INTO filetb VALUES ('" + oname + "','" + info + "','" +

savename + "','" + oname + "')"

        )

        conn.commit()

```

```

conn.close()

    flash('File Upload Successfully ')

blockchainstor(session['uname'], "File Upload FileName:" + savename)

    return render_template('UploadFile.html', oname=oname)


@app.route('/ShareFile')

def ShareFile():

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM filetb where UserName='" + session['uname'] +
'")

    data1 = cur.fetchall()

    return render_template('ShareFile.html', data=data1)

@app.route("/sharef")

def sharef():

    lid = request.args.get('lid')

    session['fid'] = lid


    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

```

```

cur.execute("SELECT UserName FROM regtb ")

data1 = cur.fetchall()

return render_template('share.html',data=data1)

@app.route("/fshares", methods=['GET', 'POST'])

def fshares():

    if request.method == 'POST':

oname = session['uname']

suname = request.form['suname']

        conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

        cursor = conn.cursor()

cursor.execute("SELECT * FROM filetb where id='" + session['fid'] + "'")

        data = cursor.fetchone()

        if data:

            info = data[3]

ffile = data[3]


        conn = mysql.connector.connect(user='root', password="", host='localhost',

                                     database='1smartdefensedb')

        cursor = conn.cursor()

cursor.execute(

```

```

        "INSERT INTO filetb VALUES ('" + oname + "','" + info + "','" + ffile
+ "','" + surname + "')"

conn.commit()

conn.close()

flash('File Share Successfully ')

blockchainstor(session['uname'], "File Share to" + surname + "FileName:" +
ffile)

return ShareFile()

@app.route("/FileInfo")

def FileInfo():

uname = session['uname']

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM filetb where UserName='" + uname + "'")

    data = cur.fetchall()

    conn = mysql.connector.connect(user='root', password="", host='localhost',
database='1smartdefensedb')

    cur = conn.cursor()

cur.execute("SELECT * FROM filetb where shareName='" + uname + "'")

    data1 = cur.fetchall()

return render_template('UFileInfo.html', data=data, data1=data1)

```



```

@app.route('/ULogout')

def ULogout():

    blockchainstor(session['uname'], 'Logout')

    return render_template('index.html')

def sendmail(Mailid, message):

    import smtplib

    from email.mime.multipart import MIMEMultipart

    from email.mime.text import MIMEText

    from email.mime.base import MIMEBase

    from email import encoders

    fromaddr = "projectmailm@gmail.com"

    toaddr = Mailid

    # instance of MIMEMultipart

    msg = MIMEMultipart()

    # storing the senders email address

    msg['From'] = fromaddr

    # storing the receivers email address

    msg['To'] = toaddr

    # storing the subject

    msg['Subject'] = "Alert"

    # string to store the body of the mail

```

```

body = message

# attach the body with the msg instance
msg.attach(MIMEText(body, 'plain'))

# creates SMTP session
s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security
s.starttls()

# Authentication
s.login(fromaddr, "qmgnxec1bkqvmusr")

# Converts the Multipart msg into a string
text = msg.as_string()

# sending the mail
s.sendmail(fromaddr, toaddr, text)

# terminating the session
s.quit()

def sendmsg(targetno,message):

    import requests

    requests.post(

"http://sms.creativepoint.in/api/push.json?apikey=6555c521622c1&route=transs
ms&sender=FSSMSS&mobilenos=" + targetno + "&text=Dear customer your
msg is " + message + " Sent By FSMSG FSSMSS")

```

```

if __name__ == '__main__':

    # app.run(host='0.0.0.0', debug=True, port=5000)

app.run(debug=True, use_reloader=True)

from __future__ import print_function

import sys, fsdk, math, ctypes, time

from fsdk import FSDK

license_key =
"fVrFCzYC5wOtEVspKM/zfLWVcSIZA4RNqx74s+QngdvRiCC7z7MHlSf2w
3+OUyAZkTFeD4kSpfVPcRVIqAKWUZzJG975b/P4HNNzpl11edXGIyGrTO
/DImoZksDSRs6wktvgr8lnNCB5IukIPV5j/jBKlgL5aqiwSfyCR8UdC9s="

if not fsdk.windows:

    print('The program is for Microsoft Windows. '); exit(1)

import win

trackerMemoryFile = "tracker70.dat"

FONT_SIZE = 30

print("Initializing FSDK... ", end=")

FSDK.ActivateLibrary(license_key);

FSDK.Initialize()

print("OK\nLicense info:", FSDK.GetLicenseInfo())

```

```

FSDK.InitializeCapturing()

print('Looking for video cameras... ', end = ")

camList = FSDK.ListCameraNames()

if not camList: print("Please attach a camera.");

print(camList[0]) # camList[0].devicePath


camera = camList[0] # choose the first camera (0)

print("using '%s'" % camera)

formatList = FSDK.ListVideoFormats(camera)

#print(*zip(range(len(formatList)), formatList), sep='\n')

print(*formatList[0:5], sep='\n')

if len(formatList)>5: print('...', len(formatList)-5, 'more formats (skipped)...')

vfmt = formatList[0] # choose the first format: vfmt.Width, vfmt.Height,
vfmt.BPP

print('Selected camera format:', vfmt)

FSDK.SetVideoFormat(camera, vfmt)


print("Trying to open '%s'... " % camera, end = ")

camera = FSDK.OpenVideoCamera(camera)

print("OK", camera.handle)

try:

fsdkTracker = FSDK.Tracker.FromFile(trackerMemoryFile)

```

except:

```
fsdkTracker = FSDK.Tracker() # creating a FSDK Tracker
```

```
fsdkTracker.SetParameters( # set realtime face detection parameters
```

```
RecognizeFaces=True, DetectFacialFeatures=True,
```

```
HandleArbitraryRotations=True, DetermineFaceRotationAngle=False,
```

```
InternalResizeWidth=256, FaceDetectionThreshold=5
```

```
)
```

```
ExamName1 = ExamName
```

```
SubjectName1 = SubjectName
```

```
Date1 = Date
```

```
Degree1 = Degree
```

```
Department1 = Department
```

```
Year1 = Year11
```

```
print(ExamName1)
```

```
print(SubjectName1)
```

```
print(Date1)
```

```
print(Degree1)
```

```
print(Department1)
```

```
print(Year1)
```

```
def dot_center(dots): # calc geometric center of dots
```

```
    return sum(p.x for p in dots)/len(dots), sum(p.y for p in dots)/len(dots)
```

```

class LowPassFilter: # low pass filter to stabilize frame size

    def __init__(self, a = 0.35): self.a, self.y = a, None

    def __call__(self, x): self.y = self.a * x + (1-self.a)*(self.y or x); return self.y


class FaceLocator:

    def __init__(self, fid):

self.lpf = None

self.center = self.angle = self.frame = None

self.fid = fid

    def isIntersect(self, state):

        (x1,y1,x2,y2), (xx1,yy1,xx2,yy2) = self.frame, state.frame

        return not(x1 >= xx2 or x2 < xx1 or y1 >= yy2 or y2 < yy1)

    def isActive(self): return self.lpf is not None

    def is_inside(self, x, y):

        x -= self.center[0]; y -= self.center[1]

        a = self.angle * math.pi / 180

        x, y = x*math.cos(a) + y*math.sin(a), x*math.sin(a) - y*math.cos(a)

        return (x/self.frame[0])**2 + (y/self.frame[1])**2 <= 1

    def draw_shape(self, surf):

        container = surf.beginContainer()

```

```

surf.translateTransform(*self.center).rotateTransform(self.angle).ellipse(facePen,
*self.frame) # draw frame

    if activeFace == self.fid:

surf.ellipse(faceActivePen, *self.frame) # draw active frame

    if capturedFace == self.fid:

surf.ellipse(faceCapturedPen, *self.frame) # draw captured frame

surf.endContainer(container)

def draw(self, surf, path, face_id=None):

    if face_id is not None:

        ff = fsdkTracker.GetFacialFeatures(0, face_id)

        if self.lpf is None: self.lpf = LowPassFilter()

        xl, yl = dot_center([ff[k] for k in FSDK.FSDKP_LEFT_EYE_SET])

xr, yr = dot_center([ff[k] for k in FSDK.FSDKP_RIGHT_EYE_SET])

        w = self.lpf((xr - xl)*2.8)

        h = w*1.4

self.center = (xr + xl)/2, (yr + yl)/2 + w*0.05

self.angle = math.atan2(yr-yl, xr-xl)*180/math.pi

self.frame = -w/2, -h/2, w/2, h/2

self.draw_shape(surf)

    name = fsdkTracker.GetName(self.fid)

    #print(name)

```

```

surf.drawString(name, font, self.center[0]-w/2+2, self.center[1]-h/2+2,
text_shadow)

surf.drawString(name, font, self.center[0]-w/2, self.center[1]-h/2, text_color)

    else:

        if self.lpf is not None: self.lpf, self.countdown = None, 35

self.countdown -= 1

    if self.countdown<= 8:

self.frame = [v * 0.95 for v in self.frame]

    else:

self.draw_shape(surf)

    name='Unkown User!';

    #print(name)

path.ellipse(*self.frame) # frame background

    return self.lpf or self.countdown> 0

activeFace = capturedFace = None

def sendmsg(targetno,message):

    import requests

requests.post("http://smsserver9.creativepoint.in/api.php?username=fantasy&pa
ssword=596692&to=" + targetno + "&from=FSSMSS&message=Dear user
your msg is " + message + " Sent By FSMSG
FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640"
)

    for o in options:

```



```

    if o.startswith('-o'): output_file = o[2:]

elif o.startswith('-sid'): skip_image_data = True

elif o.startswith('-profileid'): face_image_id = int(o[10:])

elif o.startswith('-remove'): remove_id = int(o[7:])

elif o.startswith('-extract'): extract_id = int(o[8:])

else:

    raise FSDKTrackerDataError("Unrecognized option '%s'" % o[:2])

if not output_file and face_image_id is None:

    if len(input_files) == 1:

output_file = os.path.splitext(input_files[0])[0]+'.'

    else:

        raise FSDKTrackerDataError("Output file is not specified")

trackers = [TrackerData.from_file(f) for f in input_files]

if skip_image_data:

    for t in trackers: t.remove_image_data()

td = trackers[0]

if len(trackers) == 1:

    if output_file.endswith('.'):

output_file += 'json' if td.source_type == 'bin' else 'dat'

    else:

td.merge(*trackers[1:])

    if output_file.endswith('.'):

```

```

output_file += 'json' if td.source_type == 'json' else 'dat'

if remove_id is not None:

    if not td.remove_profile(remove_id):

        print("Faces with profile id", remove_id, "are not found.")

        exit(1)

if extract_id is not None:

    if not td.extract_profile(extract_id):

        print("Faces with profile id", extract_id, "are not found.")

        exit(1)


if output_file:

outfmt = 'json' if output_file.endswith('json') else 'binary'

    if outfmt == 'json':

td.save_to_json(output_file)

        else:

td.save_to_binary(output_file)

    print(td.statistics())

if output_file:

    if remove_id is not None:

        print("\nFaces with profile id", remove_id, "are removed.")

    if extract_id is not None:

        print("\nFaces with profile id", extract_id, "are extracted.")

```

```

print("\nFile '{}' is created in {} format.".format(output_file, outfmt))

if face_image_id is not None:

    print()

    faces = [face for face in td.faces if face.id == face_image_id]

    if not faces:

        print("Faces with profile id", face_image_id, "are not found.")

    else:

        faces = [face for face in faces if face.image]

        if not faces:

            print("Faces with profile id", face_image_id, "do not have images.")

        else:

            from PIL import Image

            for face in faces:

                im = Image.frombytes('L', (face.image.width, face.image.height),
                face.image.data)

                fname = "face%s_%s.png"%(face_image_id, face.face_id)

                im.save(fname)

                print("Image file", fname, "is created")

# winapi&gdi+ constants, structs, classes and functions used in FSDK examples

import ctypes

from ctypes import windll, wintypes, byref

```

```

from ctypes import c_int, c_uint, c_void_p, c_long, c_bool, c_float as cF

from ctypes.wintypes import UINT, LPCWSTR, HWND, WPARAM,
LPARAM, HINSTANCE, HICON, HBRUSH, HANDLE, POINT

HCURSOR = HANDLE

LRESULT = LPARAM

import sys

if sys.version_info.major == 2:

    L = lambda x: ctypes.c_wchar_p(x) # ctypes.create_unicode_buffer(x)

else:

    L = lambda x: x # python 3 does not need to transform str to wchar *

# used dll

gdi32 = windll.gdi32

user32 = windll.user32

gdip = windll.gdiplus

ImageFileType_JPEG = "image/jpeg";

ImageFileType_BMP = "image/bmp";

ImageFileType_GIF = "image/gif";

ImageFileType_TIFF = "image/tiff";

ImageFileType_PNG = "image/png";

```

```

class HBITMAP(wintypes.HBITMAP):
    def __del__(self): gdi32.DeleteObject(self)

    user32.GetCursorPos(ctypes.byref(p))

    return p

def ScreenToClient(hwnd, p):
    user32.ScreenToClient(hwnd, ctypes.byref(p))

    return p

def GetWindowText(hwnd):
    char_buffer = (ctypes.c_wchar*256)()

    user32.GetWindowTextW(hwnd, char_buffer, 255)

    return char_buffer.value

SetWindowText = user32.SetWindowTextW

DrawState = user32.DrawStateW

Rectangle = gdi32.Rectangle

Ellipse = gdi32.Ellipse

PeekMessage = user32.PeekMessageW

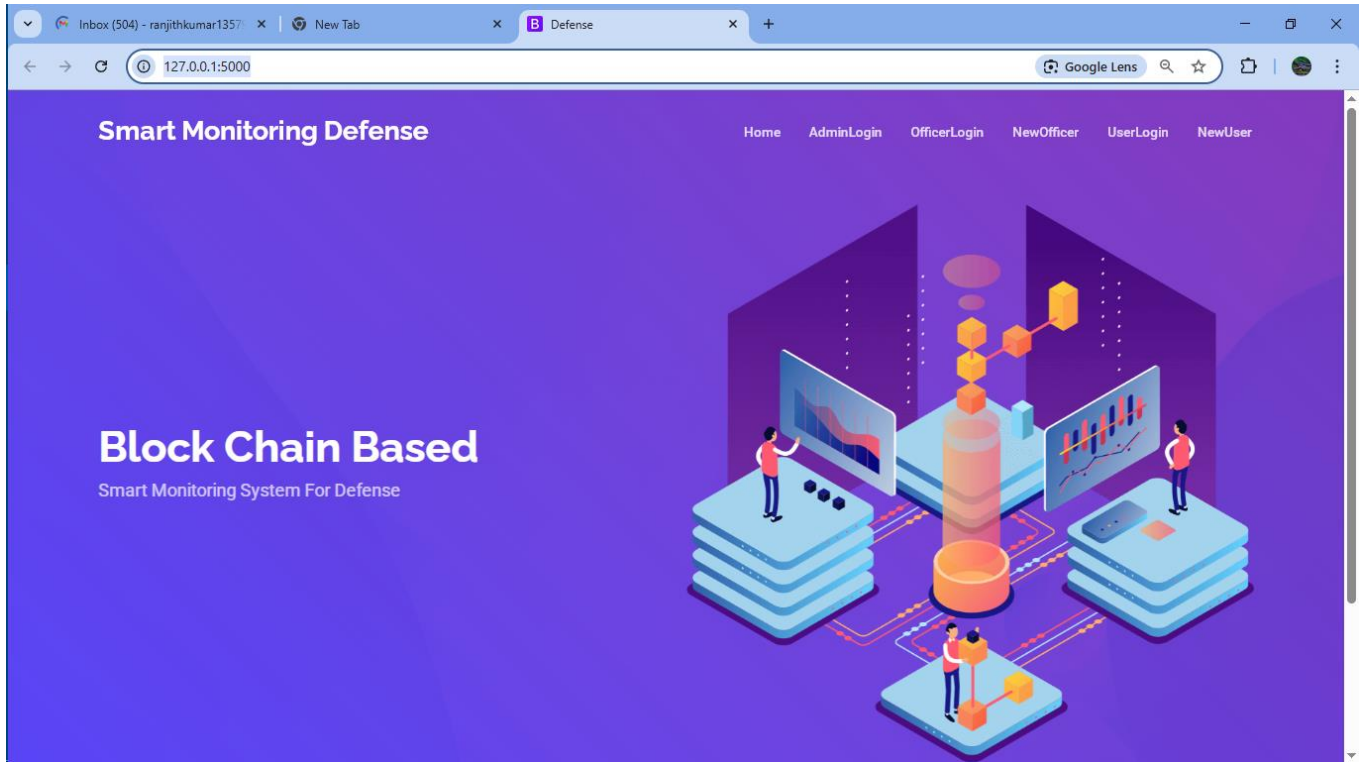
TranslateMessage = user32.TranslateMessage

DispatchMessage = user32.DispatchMessageW

SendMessage = user32.SendMessageW

```

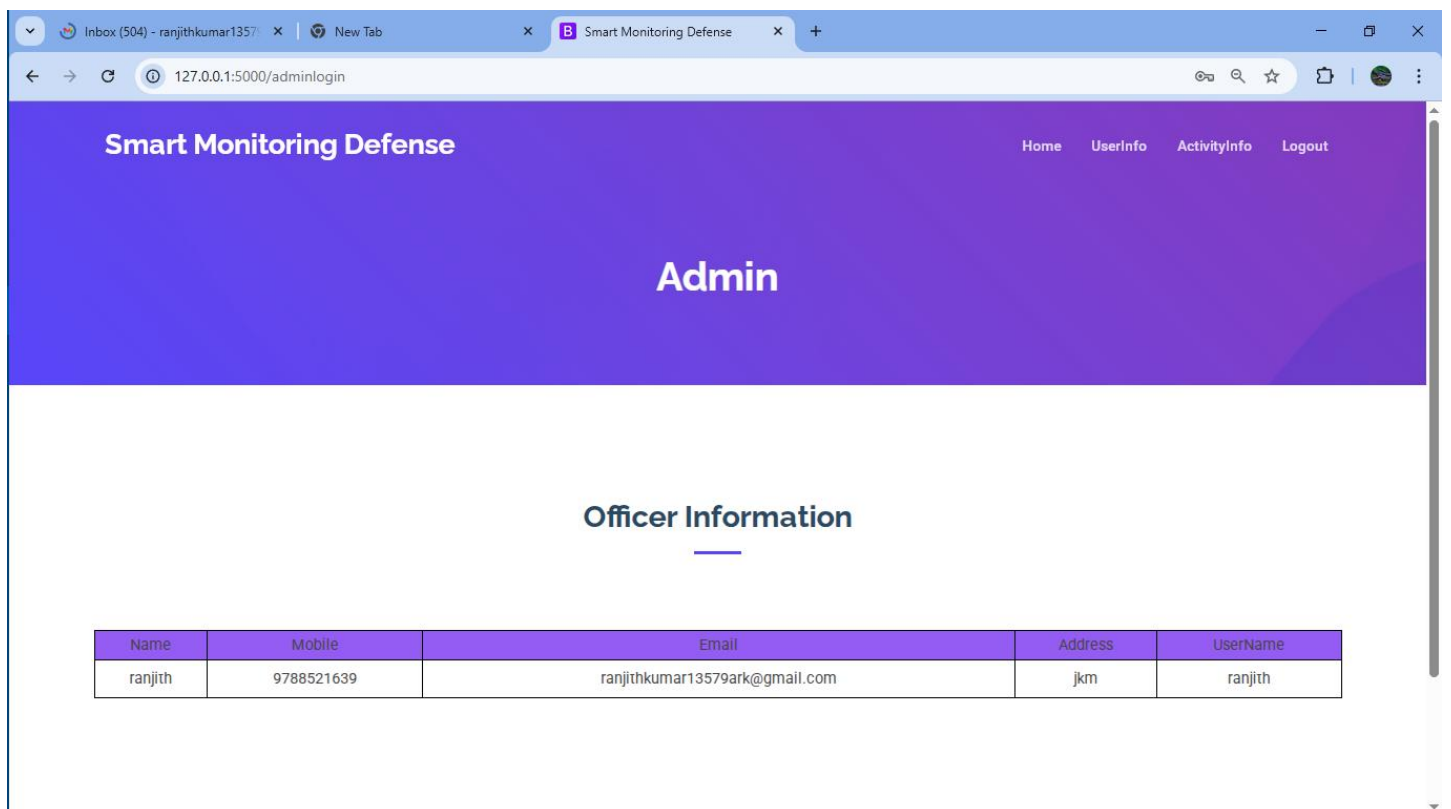
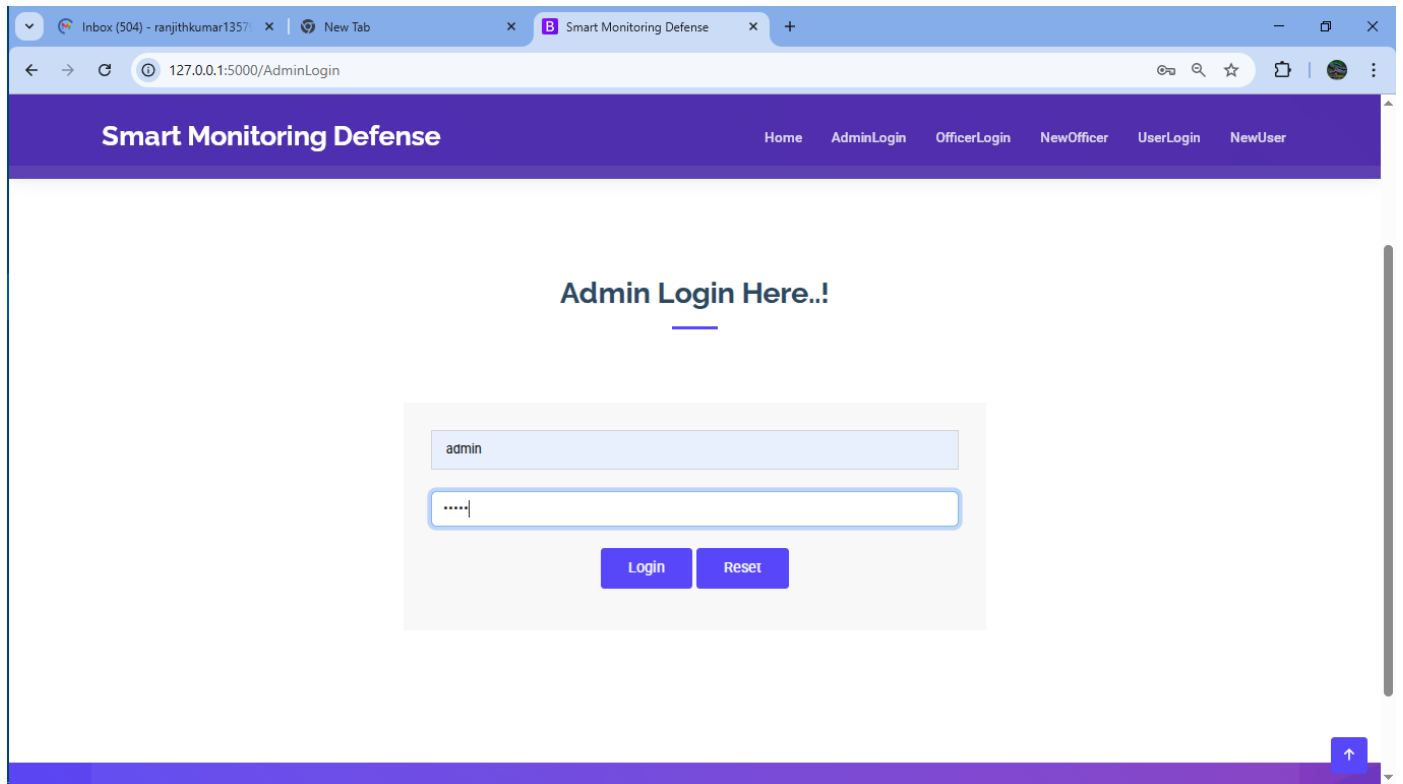
## APPENDIX 2: SCREEN SHOTS



The screenshot shows the 'New Officer Register Here..!' registration page. The browser's address bar displays '127.0.0.1:5000/NewOfficer'. The page has a dark purple header with the title 'Smart Monitoring Defense' on the left and a navigation menu on the right containing links: 'Home', 'AdminLogin', 'OfficerLogin', 'NewOfficer', 'UserLogin', and 'NewUser'. The main content area is white and features a registration form with the following fields and buttons:

- First Name:
- Phone Number:
- Email:
- Address:
- Username:
- Password:
- Buttons:  and

A small blue button with an upward arrow is located at the bottom right of the page.



Smart Monitoring Defense

HomeAdminLoginOfficerLoginNewOfficerUserLoginNewUser

NewUser Register Here..!

kumar

9788521639

ranjithkumar13579ark@gmail.com

jayakondam

kumar

....

Submit

Reset

Smart Monitoring Defense

OfficerLoginNewOfficerUserLoginNewUser

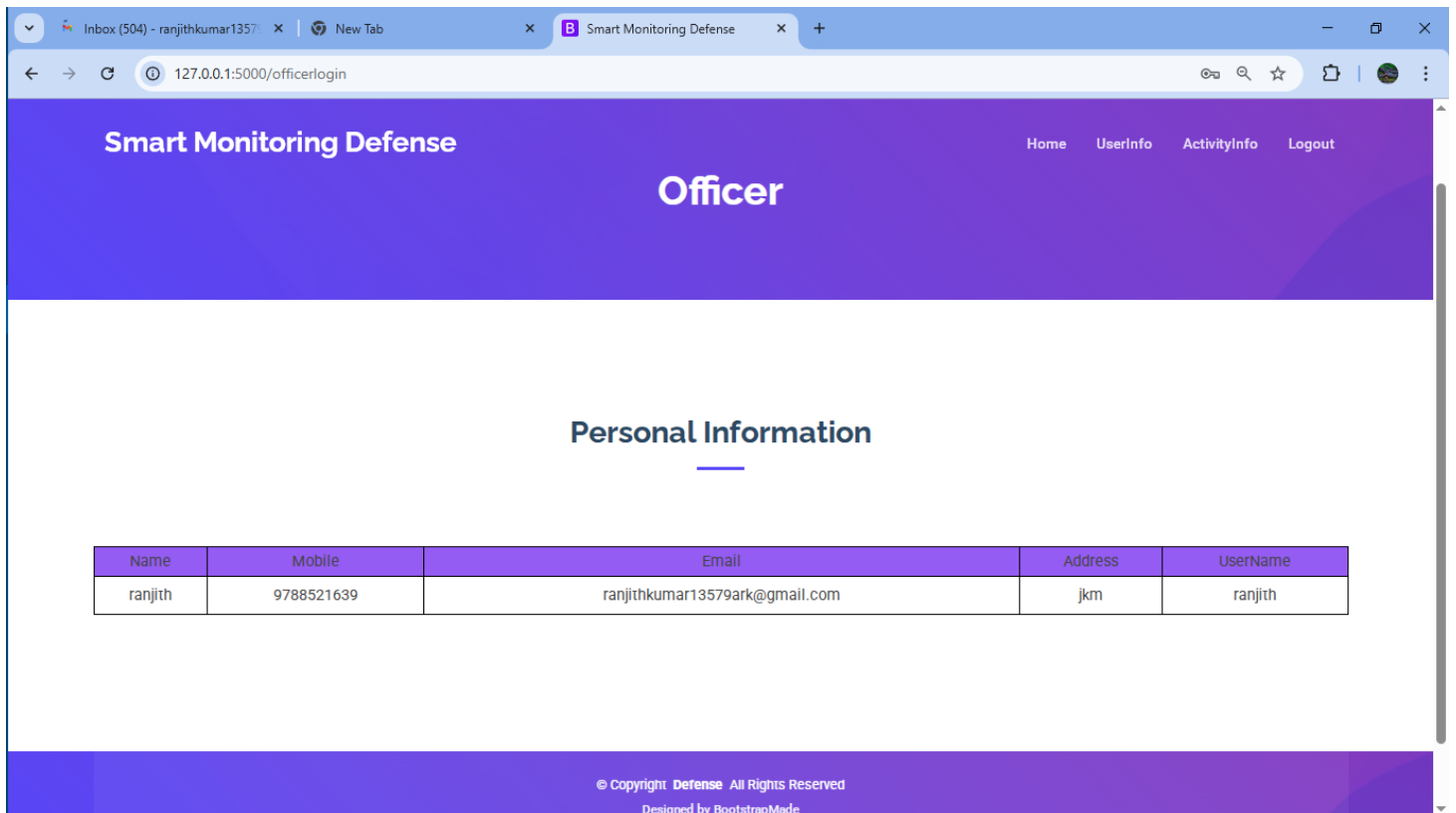
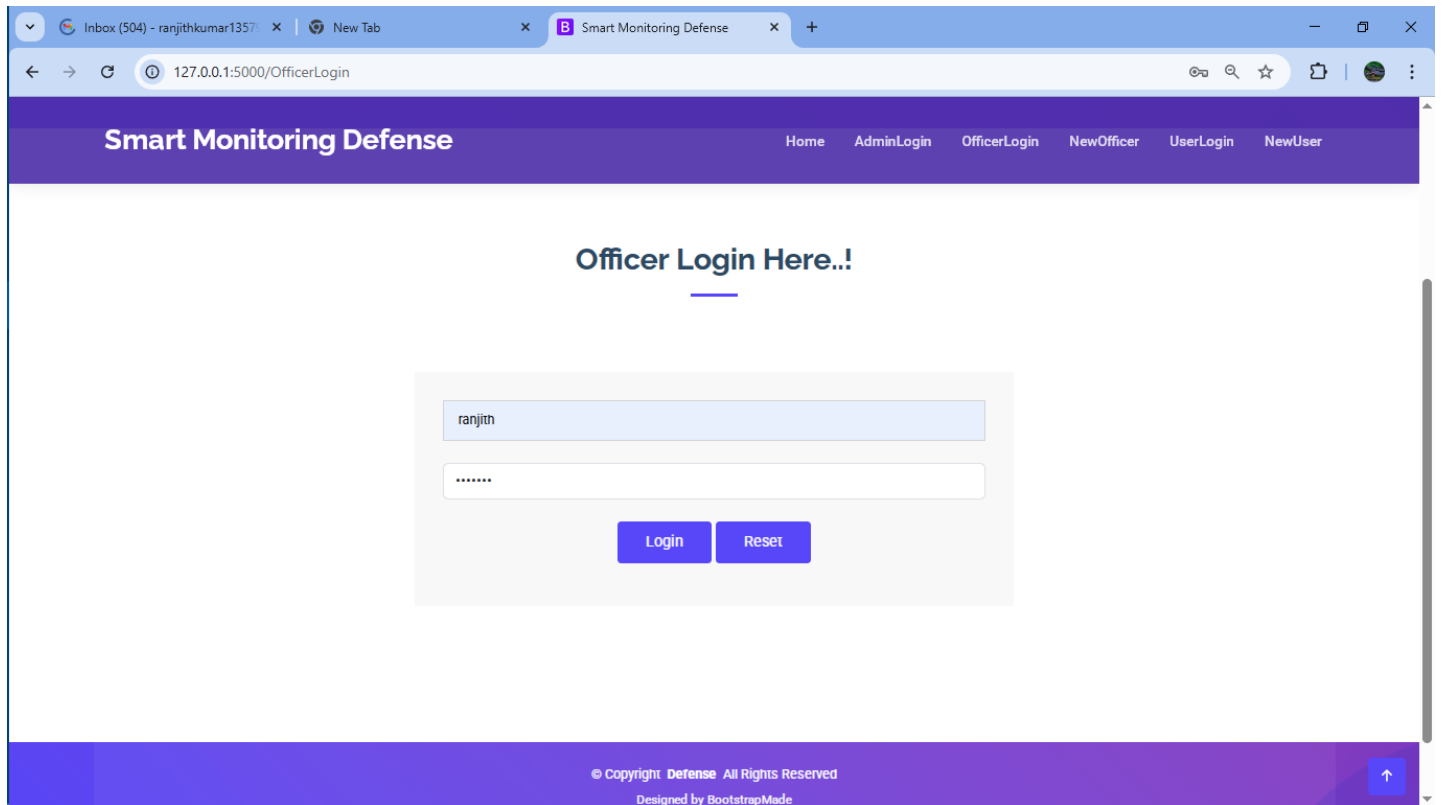
127.0.0.1:5000 says  
Record Saved!

OK

Home

70





Inbox (504) - ranjithkumar1357

New Tab



Smart Monitoring Defense

127.0.0.1:5000/OUUserInfo

Smart Monitoring Defense

Approved Waiting User Information

[Home](#)
[UserInfo](#)
[ActivityInfo](#)
[Logout](#)

Name	Mobile	Address	UserName	Status	Image	Action
sakara	8489096269	ariyalur	sakara	waiting		<a href="#">Approved//Reject</a>
kumar	9788521639	jayakondam	kumar	waiting		<a href="#">Approved//Reject</a>

Approved/Reject User Information

WhatsApp Web

Inbox (506) - ranjithkumar1357

Smart Monitoring Defense




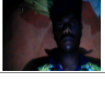
127.0.0.1:5000/OUUserInfo

Smart Monitoring Defense

ariyalur sakara waiting

[Home](#)
[UserInfo](#)
[ActivityInfo](#)
[Logout](#)

Approved/Reject User Information

Name	Mobile	Email	Address	UserName	Status	Image
sound	9788521639	ranjithkumar13579ark@gmail.com	jkm	sound	Active	
mark	8489096269	v7200512@gmail.com	jmjh	mark	Active	
ANBARASI	9952321289	anbarasi1990@gmail.com	PERAMBALUR	ANBARASI	Active	
kumar	9788521639	ranjithkumar13579ark@gmail.com	jayakondam	kumar	Active	

Inbox (505) - ranjithkumar13579ark@...Smart Monitoring Defense

127.0.0.1:5000/AActivityInfo

Smart Monitoring Defense

HomeUserInfoActivityInfoLogout

2025-05-0912:34:43

ActivityInfoLogout

Hash176FA18150756A8D7D7BA8E290836EC5EDC295491C130AC0994A5231FEDFF20A4

Hash2F5B3B707C36473AD9649A1DCEB03367CCBC8DB4BC96DF2DBF00A75642F97D7D5

DownloadDownload

UserNamekumar

Date2025-05-09

Time22:02:34

ActivityInfoLogin

Hash1F5B3B707C36473AD9649A1DCEB03367CCBC8DB4BC96DF2DBF00A75642F97D7D5

Hash260CFDF7597D3A0557288BC2AB5B202611EBADFDA47023C924F3E0CB7877E9E5D

DownloadDownload

WhatsApp WebAlert - ranjithkumar13579ark@...Smart Monitoring Defense

mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzQbfBtlXrSZdmJVZKbMcQvgFpdK

Gmail

Search mail

19 of 678

Alert

projectmail@gmail.com

Your Request Rejected

Thu, May 8, 10:34 AM (10 days ago)

☆

3

projectmail@gmail.com

Your Defense Login Key :7736

Thu, May 8, 12:12 PM (10 days ago)

☆

projectmail@gmail.com

to me

Your Defense Login Key :4711

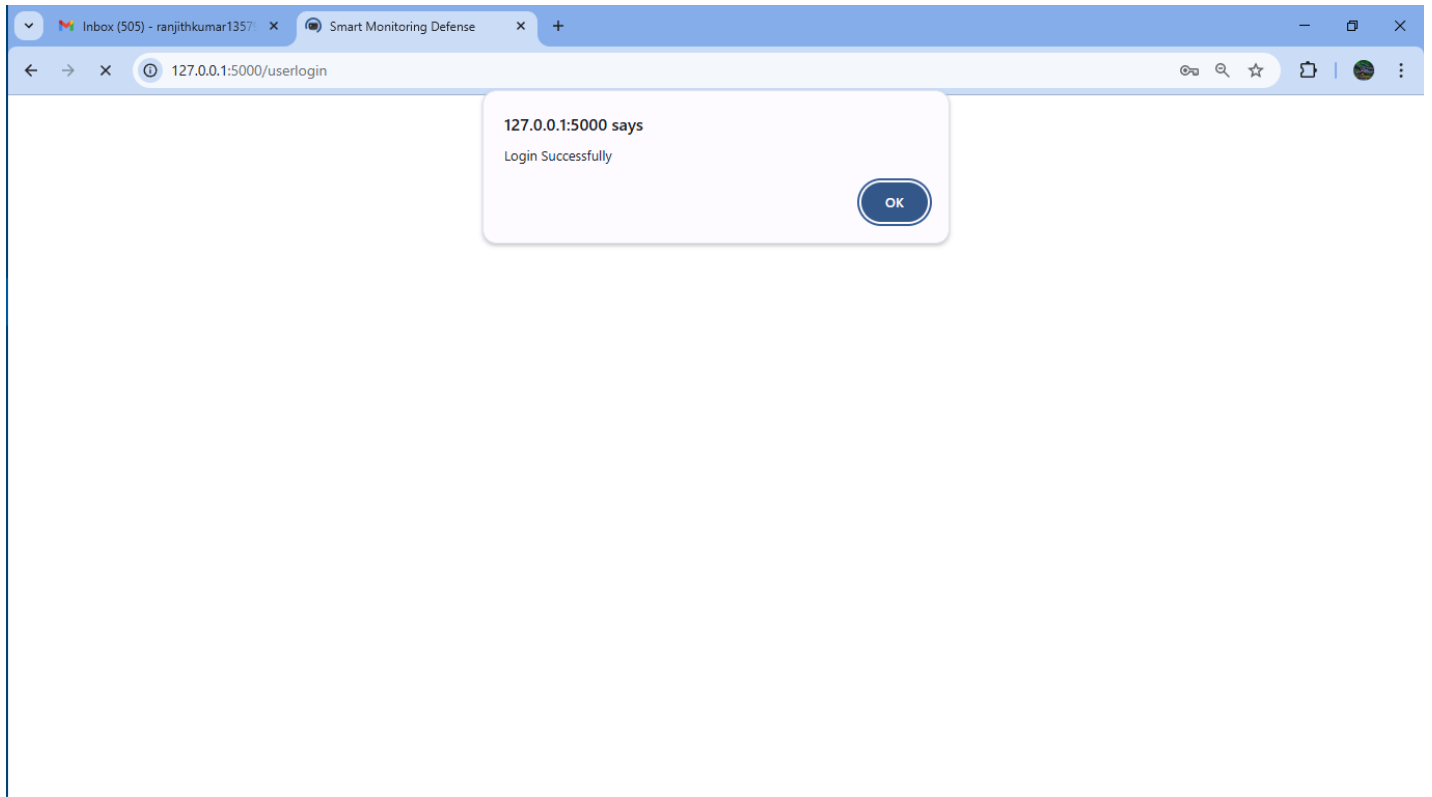
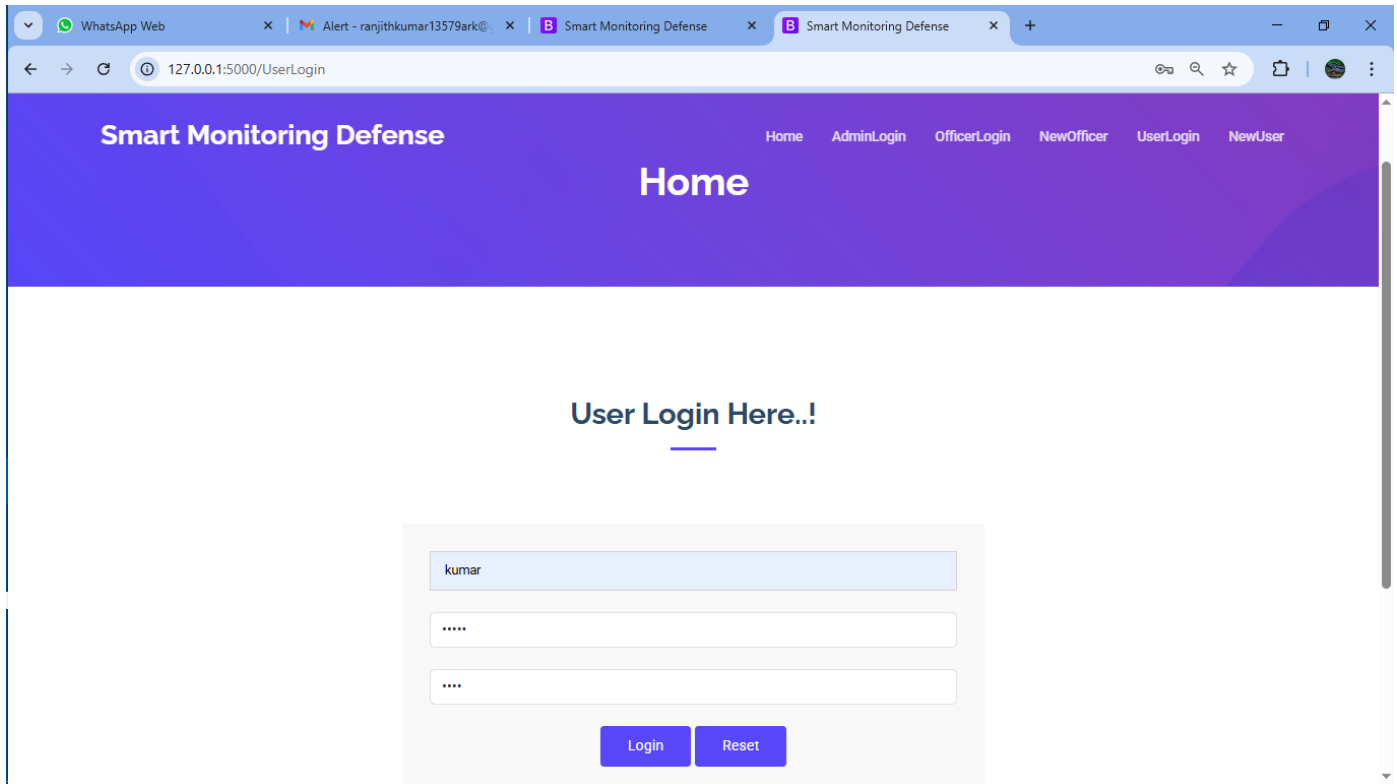
Thu, May 8, 12:21 PM (10 days ago)

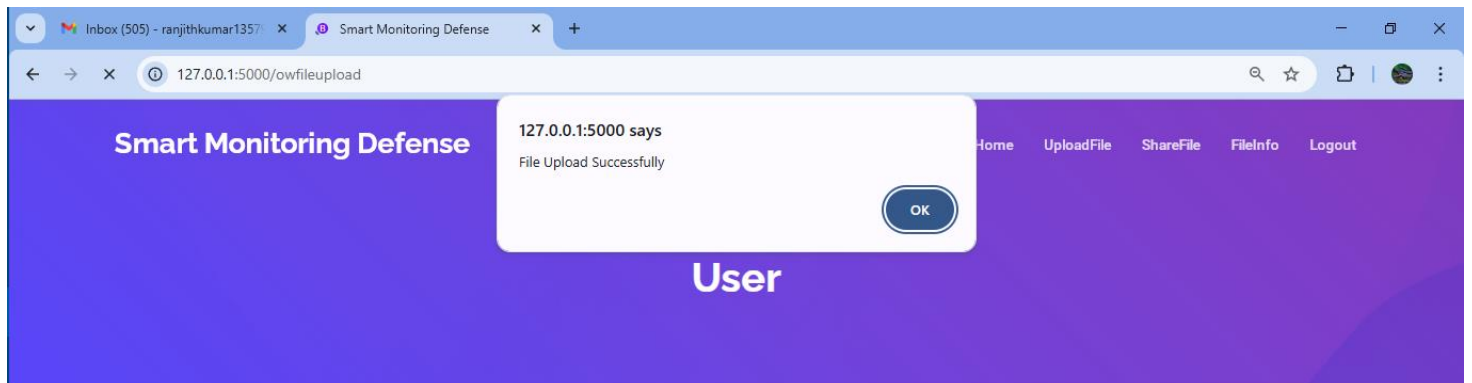
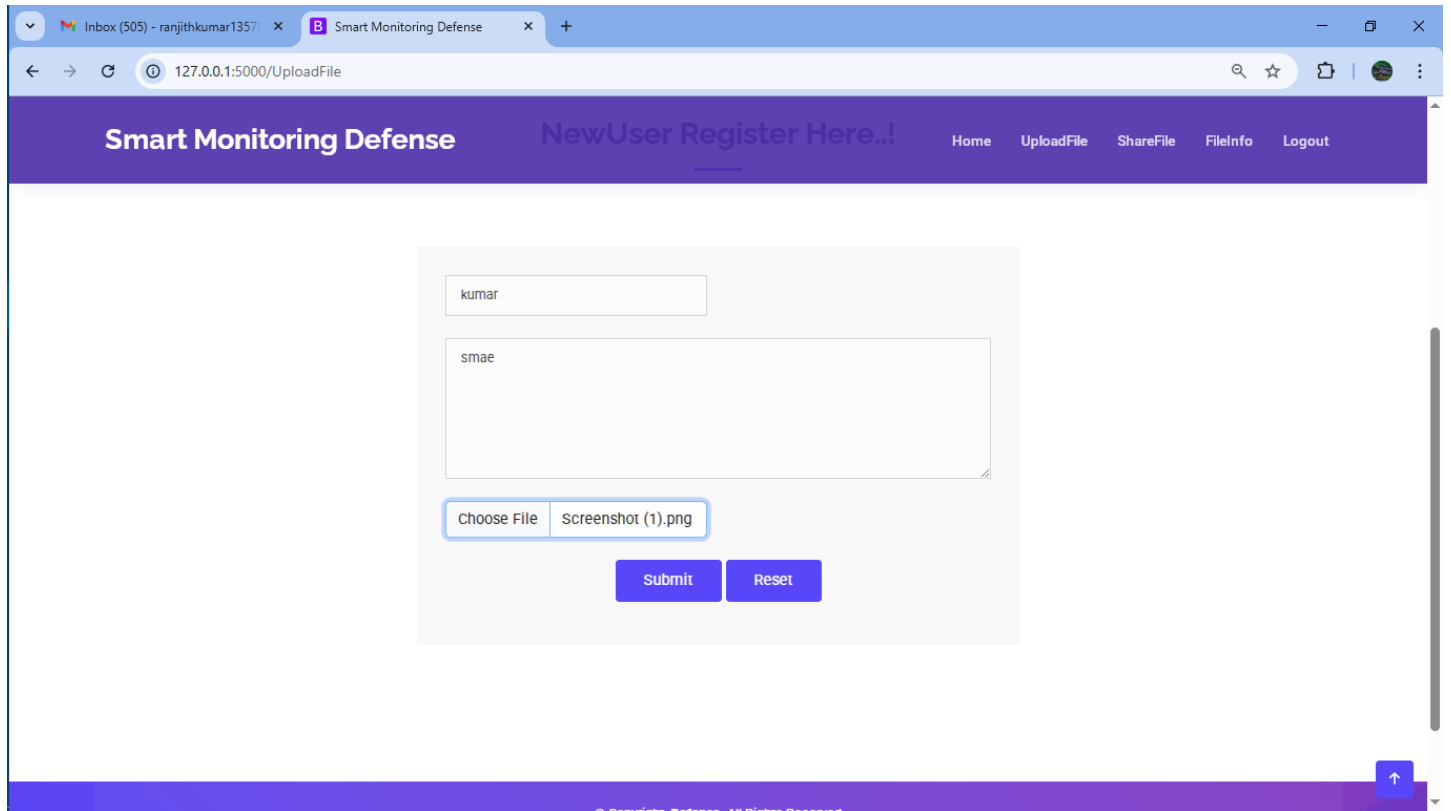
★😊↩⋮

↩ Reply

➦ Forward

😊





Inbox (505) - ranjithkumar1357

Smart Monitoring Defense

127.0.0.1:5000/fshares

Smart Monitoring Defense

HomeUploadFileShareFileFileInfoLogout

User

Upload File Information

UserName	Fileinfo	FileName	Action
kumar	smae	542Screenshot (1).png	Share
kumar	542Screenshot (1).png	542Screenshot (1).png	Share

Inbox (505) - ranjithkumar1357

Smart Monitoring Defense

127.0.0.1:5000/sharef?lid=4

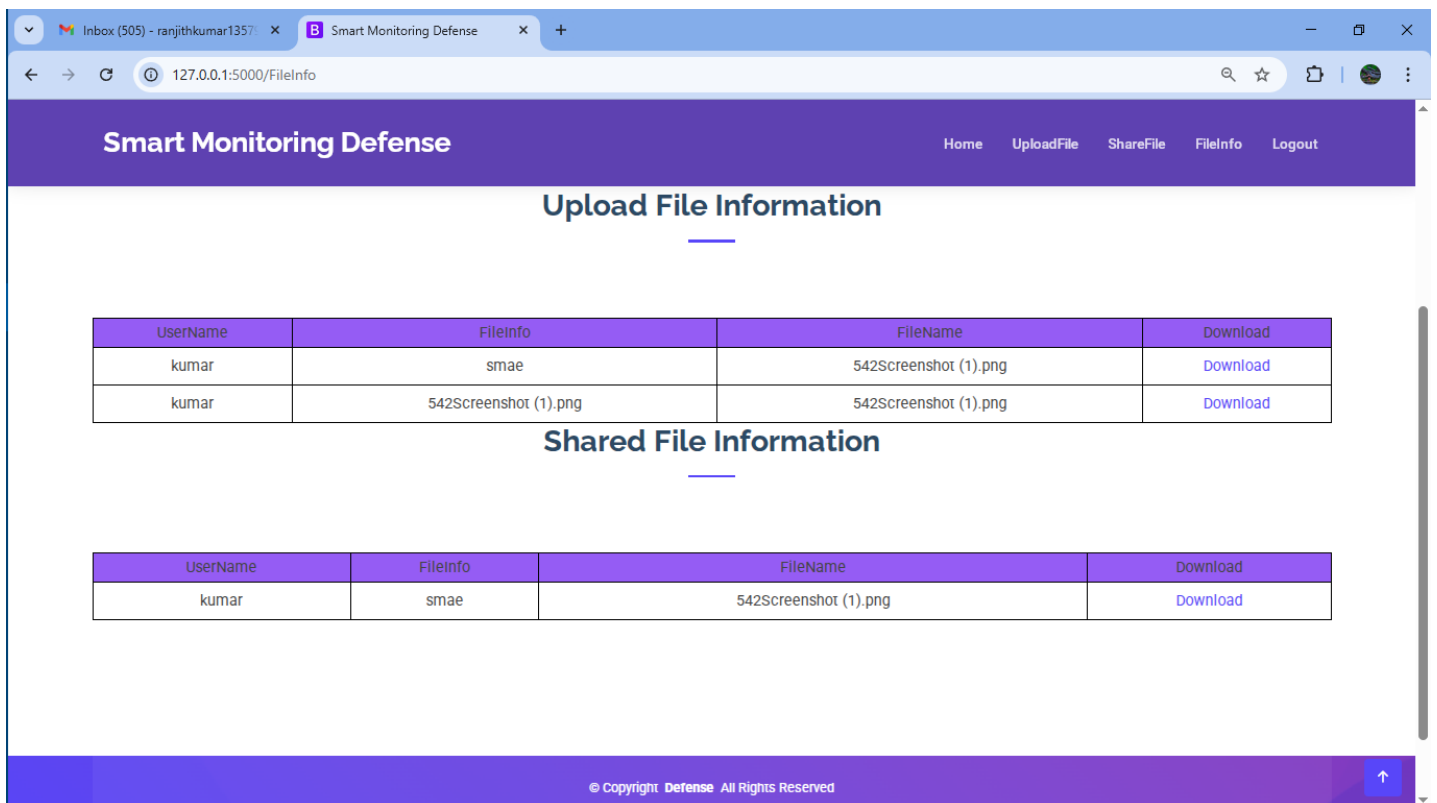
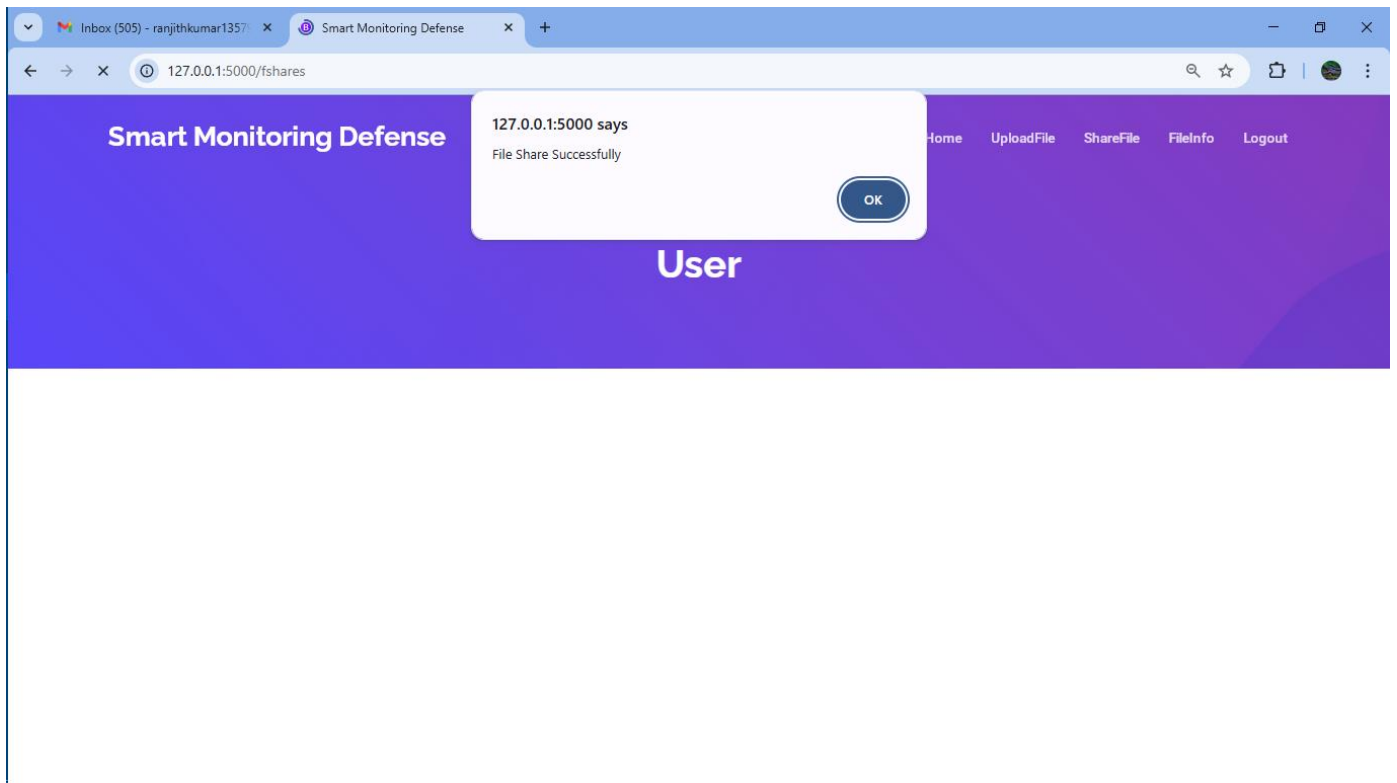
Smart Monitoring Defense

HomeUploadFileShareFileFileInfoLogout

User

Share File

SubmitReset



Inbox (505) - ranjithkumar1357

Smart Monitoring Defense

127.0.0.1:5000/AActivityInfo

Smart Monitoring Defense

HomeUserInfoActivityInfoLogout

2025-05-09

22:02:34

Logout

76FA18150756A8D7D7BA8E290836EC5EDC295491C130AC0994A5231FEDFF20A4

F5B3B707C36473AD9649A1DCEB03367CCBC8DB4BC96DF2DBF00A75642F97D7D5

Download

kumar

2025-05-09

22:02:34

Login

F5B3B707C36473AD9649A1DCEB03367CCBC8DB4BC96DF2DBF00A75642F97D7D5

60CFDF7597D3A0557288BC2AB5B202611EBADFDA47023C924F3E0CB7877E9E5D

Download

kumar

2025-05-09

22:04:22

File Upload FileName:542Screenshot (1).png

60CFDF7597D3A0557288BC2AB5B202611EBADFDA47023C924F3E0CB7877E9E5D

E59B34E34AE2B08A8721512D7272B62CC442CF1FAF3726BC9285BAAD09631119

Download

kumar

2025-05-09

22:04:43

File Share tosoundFileName:542Screenshot (1).png

E59B34E34AE2B08A8721512D7272B62CC442CF1FAF3726BC9285BAAD09631119

103E1203D578B716D782A292764685601B1C4589B60F260AF7E8BF3441E152B9

Download



## REFERENCES

- [1] Bhatia, Munish, and Ankush Manocha. "Cognitive framework of food quality assessment in IoT-inspired smart restaurants." *IEEE Internet of Things Journal* 9.9 (2020): 6350-6358.
- [2] Bhatia, Munish, et al. "Internet of things-inspired healthcare system for urine-based diabetes prediction." *Artificial Intelligence in Medicine* 107 (2020): 101913.
- [3] Bhatia, Munish, Sandeep K. Sood, and Simranpreet Kaur. "Quantumized approach of load scheduling in fog computing environment for IoT applications." *Computing* 102.5 (2020): 1097-1115.
- [4] Fukawa, Nobuyuki, and Aric Rindfleisch. "Enhancing innovation via the digital twin." *Journal of Product Innovation Management* 40.4 (2023): 391-406.
- [5] Javed, Safdar Hussain, et al. "APT adversarial defence mechanism for industrial IoT enabled cyber-physical system." *IEEE Access* 11 (2023): 74000-74020.
- [6] Li, Luning, et al. "Digital twin in aerospace industry: A gentle introduction." *IEEE Access* 10 (2021): 9543-9562.
- [7]. Pătrașcu, Petrișor. "Emerging technologies and National Security: The impact of IoT in critical infrastructures protection and defence sector." *Land Forces Academy Review* 26.4 (2021): 423-429.
- [8] Sharma, Pradip Kumar, et al. "Wearable computing for defence automation: Opportunities and challenges in 5G network." *IEEE Access* 8 (2020): 65993-66002.
- [9] Sirait, Jonathan, Hazen Alrasyid, and Nadia Aurora Soraya. "Strengthening The Defense Industry's Independence Through The Internet Of Things In The Manufacturing Sector: A Review." *International Journal of Science, Technology & Management* 4.2 (2023): 335-340.
- [10] Zhou, Xiaokang, et al. "Deep-learning-enhanced multitarget detection for end-edge-cloud surveillance in smart IoT." *IEEE Internet of Things Journal* 8.16 (2021): 12588-12596.