Physical Security Management System

BAKLIWAL FOUNDATION
College of Arts, Commerce & Science

A

PROJECT REPORT

On

Physical Security Management System

Submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Computer Applications (BCA)
of
KaviKulaguru Kalidas Sanskrit Vishwavidyalaya

Submitted by

Dhanraj Rajesh Tungar

Harsh Dinesh Sathe

Afzan Arshad Shaikh

Shaikh Mohammed Asif Feroz

Under the guidance of

Prof. Kalyani Akshay Kulkarni

KaviKulaguru Kalidas Sanskrit University

Bakliwal Foundation Collage of Arts, Commerce & Science

Vashi Navi Mumbai

BATCH:2022-2025

KaviKulaguru Kalidas Sanskrit University

Bakliwal Foundation Collage of Arts, Commerce & Science

Vashi

CERTIFICATE

This is to certify that the project entitled **Physical Security Management System** is undertaken at the Bakliwal Foundation College, Vashi Navi Mumbai by **Dhanraj Rajesh Tungar** holding **PRN No. 2022018100094671** studying in **Bachelor of Computer Application** Semester – VI has been satisfactorily completed as prescribed by the University of KKSU, during the year 2024 – 2025

Project In-Charge **Coordinator**

External Examiner

Internal Examiner **Principal**

Declaration

I hereby declare that the project "**Physical Security Management System (PSMS)"** is the result of my own efforts and has been developed as a part of my academic curriculum. This project has been undertaken with the objective of creating a robust, automated security platform to streamline physical security management through real-time monitoring, access control, and incident reporting.

The contents of this report reflect the conceptual framework and core functionalities of the system, including CCTV integration, alarm management, and role-based access control. While every effort has been made to ensure the accuracy and effectiveness of the system, it may still be subject to further refinement and enhancement based on future technological advancements.

This project has been completed under the guidance of **Prof. Kalyani Kulkarni,** whose expertise and support have been invaluable throughout the development process. I take full responsibility for the content of this report and affirm the originality of the work presented.

Dhanraj Rajesh Tungar

Acknowledgement

I would like to express my sincere gratitude to all those who have supported me throughout the completion of my project, **Physical Security Management System** . Without their guidance and encouragement, this project would not have been possible.

First and foremost, I would like to thank my supervisor, **Prof.Kalyani Akshay Kulkarni**, for his constant support and valuable guidance. His insights and expertise have been instrumental in shaping the direction of this project. I deeply appreciate his patience and constructive feedback throughout the process.

I would also like to extend my heartfelt thanks to the faculty **members** of the **BCA Program Dr. Sharadkumar Shah, H.O.D Prof. Sneha Shashikant Lokhande, Prof. Shaikh Mohammed Umar, Prof. Divya Patil, Prof. Kalyani Kulkarni, Prof. Ankit Srivastava** at **Bakliwal Foundation collage, Vashi Navi Mumbai** for their dedication and knowledge, which provided me with the foundation to successfully carry out this project. Their lessons and encouragement have been crucial in my academic journey.

I am grateful to my friends and peers for their continuous support and collaboration. Their ideas, discussions, and feedback have significantly contributed to the development and improvement of this project.

A special thanks to my family for their unwavering support, love, and understanding during the course of this project. Their belief in my abilities and their encouragement have been a great source of motivation.

Lastly, I would like to acknowledge the valuable online resources and research materials that aided in my understanding and completion of this project. Their accessibility and wealth of information have been invaluable to my work.

AIM: To create a Physical Security Management for secure user authentication and access management.

ABSTRACT

The **Physical Security Management** is a web-based application designed to streamline and automate the management of physical security components such as **CCTV cameras, access cards, Email alerts and alarms**. This system provides a centralized platform for real-time monitoring, access control, and incident management, ensuring a secure environment for organizations.
The system features **role-based access control**, allowing administrators, security personnel, and regular users to perform specific tasks based on their roles. Key functionalities include:

- **User Authentication**: Secure login and registration.
- **Access Control**: Logging and tracking access attempts.
- **CCTV Monitoring**: Simulating real-time video feeds.
- **Alarm Management**: Triggering and resolving alarms.
- **Reporting**: Generating reports for access logs and incidents.
- **Email Alert**: Automated email alert

Built using **HTML, CSS, JavaScript, PHP, and MySQL**, **Python** the system is designed to be user-friendly, scalable, and efficient. The project aims to address the limitations of manual security management by providing a robust, automated solution that enhances security and operational efficiency.

This project demonstrates the practical application of web development and database management techniques to solve real-world problems. It serves as a foundation for future enhancements, such as IoT integration and advanced analytics.

Table of Contents

Table of Figure

Chapter 1

Chapter 1:

Introduction

1.1 Background

In today's world, security is a top priority for organizations and individuals. Traditional security systems often rely on manual processes, which are inefficient and prone to errors. The Physical Security Management addresses these challenges by providing a centralized, automated platform for managing physical security components like CCTV cameras, access control systems, and alarms.

This system integrates modern technologies to offer real-time monitoring, access control, and incident management. It ensures a secure environment by automating processes, providing instant alerts, and generating detailed reports. The project is highly relevant for organizations, institutions, and residential complexes looking to enhance their security measures.

1.2 Objective

1) Centralized Security Management:
   a) Provide a unified platform to manage CCTV feeds, access control, and alarms.

2) Real-Time Monitoring:
   a) Enable live monitoring of CCTV feeds and instant alerts for suspicious activities.

3) Access Control:
   a) Manage access cards and log access attempts (successful/failed).

4) Incident Management:
   a) Track and resolve security incidents (e.g., alarms) efficiently.
   b) When suspicious activity like unauthorized face detection occurs, an email is sent.

5) Reporting and Analytics:
   a) Generate detailed reports and visualizations for better decision-making.

6) Scalability:
   a) Design a system that can be easily scaled to accommodate future needs.

1.3. Purpose, Scope, and Applicability

1) **Purpose**
   a) Centralize security management.
   b) Automate monitoring and access control.
   c) Provide real-time alerts and detailed reports.

2) **Scope**
   a) CCTV integration and access control.
   b) Alarm management and reporting.

3) **Applicability**
   a) Organizations (asset protection).
   b) Educational institutions (campus safety).
   c) Residential complexes (property security).

d) Achievements

i) The **Physical Security Management** has achieved:
   (1) Centralized management of CCTV, access control, and alarms.
   (2) Real-time monitoring and instant alerts.
   (3) incident management.
   (4) Secure.
   (5) Scalable design for future needs.

e) Organization of Report

i) The report is organized as follows:
   (1) **Introduction**: Overview, objectives, and achievements.
   (2) **Survey of Technologies**: Features and advantages of technologies used.
   (3) **Requirements and Analysis**: Problem statement and system requirements.
   (4) **System Design**: Diagrams like activity, data flow, and use case diagrams.
   (5) **Implementation and Testing**: Code details and testing strategies.
   (6) **Results and Discussions**: Project outcomes and their significance.
   (7) **Cost and Benefit Analysis**: Cost of implementation and system benefits.

Chapter 2

Chapter 2
SURVEY OF TECHNOLOGIES

2.1 FEATURES OF BACKEND

1. **Database Management** : Stores and retrieves data (users, access logs, alarms) using MySQL.
2. **API Integration** : Provides APIs for front-end communication (e.g., fetch access logs, alarms).
3. **Real-Time Alerts**: Triggers and logs alarms for suspicious activities.
4. **Access Control**: Manages access cards and logs access attempts.
5. **Data Processing**: Processes data for reporting and analytics.
6. **Scalability**: Designed to handle growing data and user demands.

2.2 FEATURES OF FRONTEND

The front-end of the **Physical Security Management** includes the following features:

1) **User Interface**:
    a) Intuitive and user-friendly design for easy navigation.
2) **Dashboard**:
    a) Displays real-time CCTV feeds, access logs, and alarms.
3) **Responsive Design**:
    a) Works seamlessly on desktops, tablets, and mobile devices.
4) **Interactive Elements**:
    a) Buttons, forms, and tables for user interaction (e.g., login, view reports).
5) **Real-Time Updates**:
    a) Automatically updates data (e.g., alarms, Email and access logs) without refreshing the page.
6) **Visualizations**:
    a) Charts and graphs for displaying reports and analytics.

2.3 COMPARATIVE STUDY

1) **Traditional Security Systems**

   a) **Manual Monitoring**:

      i. Requires constant human intervention to monitor CCTV feeds and access logs.

   b) **Limited Integration**:

      i.Systems like CCTV, access control, and alarms often operate independently.

   c) **Delayed Response**:

      i. Lack of real-time alerts leads to slower response times.

   d) **Basic Reporting**:

      i.Limited or no reporting capabilities for analysing security incidents.

2) **Physical Security Management**

   a) **Automated Monitoring**:

      i) Real-time monitoring of CCTV feeds and instant alerts for suspicious activities.

   b) **Integrated Platform**:

      i) Combines CCTV, access control, and alarms into a single system.

   c) **Faster Response**:

      i) Real-time alerts enable quicker response to security incidents.

   d) **Advanced Reporting**:

      i) Detailed reports and analytics for better decision-making.

   e) **Scalability**:

      i) Designed to handle growing data and user demands.

2.4 ADVANTAGES

1) **The system offers:**
   a) Centralized management of CCTV, access control, email , face detection and alarms

   b) Real-time monitoring and instant alerts.

   c) Efficient access control and improved response time.

d) Detailed reporting and user-friendly interface.

e) Scalable and cost-effective design.

2.5 DISADVANTAGES

a) High initial setup cost.

b) Requires technical expertise.

c) Depends on power and internet.

d) Potential data security risks.

e) Scalability challenges.

f) Learning curve for users.

Chapter 3

Chapter 3:
Requirements & Analysis

3.1 Problem Definition

1) Traditional security systems face:
   a) Manual monitoring and lack of integration.
   b) Delayed response and limited reporting.
   c) Scalability issues.

The **Physical Security Management** solves these by providing a centralized, automated platform for real-time monitoring and incident management.

3.2 Proposed System

1) The system offers:
   a) Centralized platform for CCTV, access control, and alarms.
   b) Real-time monitoring and automated access control.
   c) Efficient incident management and detailed reporting.
   d) User-friendly interface and scalable design.

3.4 Planning and Scheduling

1) Timeline

Project Timeline: July 2024 – April 2025

Phase 1: Planning & Analysis

- **July 2024**
  - **Week 1–2**: Requirement analysis and survey

Phase 2: System Design

- **August 2024**
  - **Week 3–4**: System design and database schema

Phase 3: Backend Development

- **September 2024**
  - **Week 5–6**: Backend – Access control system
- **October 2024**
  - **Week 7–8**: Backend – API development & integration

Phase 4: Frontend Development

- **November 2024**
  - **Week 9–10**: UI Design and frontend integration

Phase 5: Testing & Deployment

- **December 2024**
  - **Week 11**: Testing and bug fixing
  - **Week 12:** Deployment and basic documentation

Phase 6: Enhancements & Reports

- **January – February 2025**
  - Add-ons: Reports module, logs, face detection, alarm features
  - Documentation expansion

Phase 7: Review & Final Submission Prep

- **March 2025**

- Code review, UI polishing, final testing

Phase 8: Submission

- **April 2025**
  - Final project documentation
  - Project submission and viva preparation

2) Milestones

a) Milestone 1: System design completed.
b) Milestone 2: Backend completed.
c) Milestone 3: Frontend completed.
d) Milestone 4: Testing and deployment done.

3.5 Software and Hardware Requirements

3.5.1 Software Requirements

1) Operating **System**:
   a) Windows
   b) macOS
   c) Linux.

2) Development **Tools**:
   a) PHP (version 7.4 or higher)
   b) MySQL Database Server
   c) Web Server (e.g., Apache or Nginx)
   d) IDE/Text Editor (e.g., Visual Studio Code, Sublime Text)

2) **Languages**:
   a) HTML, CSS, JavaScript (for front-end)

   b) PHP, Python (for back-end)

3) **Database**:
   i) MySQL for storing user data and access logs

      (a) Platform independent.

      (b) Supports multimedia like images, audio, and video.

      (c) HTML5 is used to build modern websites.

3.5.2 Hardware Requirements

1) **Processor**: Intel Core i3 or equivalent

2) **RAM**: Minimum 4GB (8GB recommended for smoother performance)

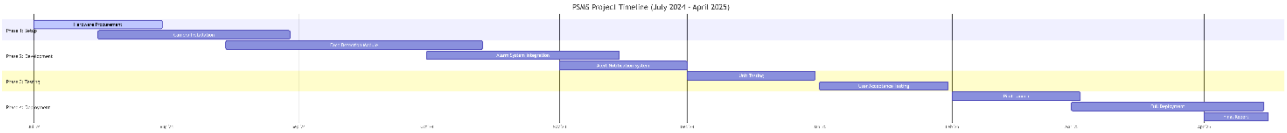
Storage: 500MB of free disk space for the development environment and database

Chapter 4

Chapter 4
SYSTEM DESIGN

1) Gannt chart

Fig 4.1



Key Features:

1. Timeline: 10-month schedule from July 2024 to April 2025
2. Critical Path:
    - Camera Installation → Face Detection → Alarm System

2 a) Activity Diagram :

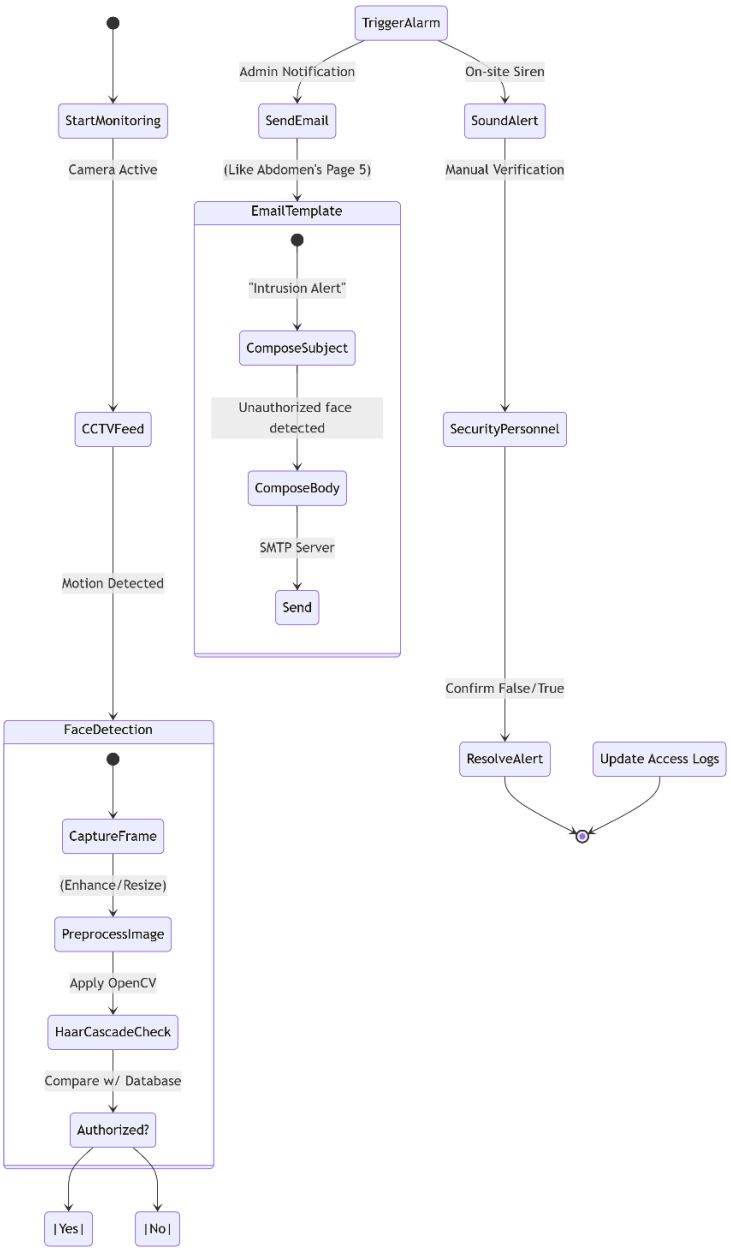


Fig 4.2

1. Key Elements Explained:

    a) Start Monitoring: Initial state (like Abdomen's "Input Data" in Fig 4.1).

1. Face Detection Subprocess:

    i. Preprocess Image:
    a. Resize/Clean frame (similar to Abdomen's image enhancement on Page 25).

ii. Haar Cascade Check:
   a. OpenCV-based detection (analogous to Abdomen's CNN classification).

iii. Decision Node (Authorized?):
   a. Like Abdomen's "Guard Conditions" (Page 22).

iv. Parallel Actions:
   a. Alarm triggers both Sound Alert and Send Email (like Abdomen's synchronous alerts).

v. Email Template:
   a. Reuses the structure from Abdomen's email (Page 5) but for security alerts.

2 b) Active for Security Diagram:

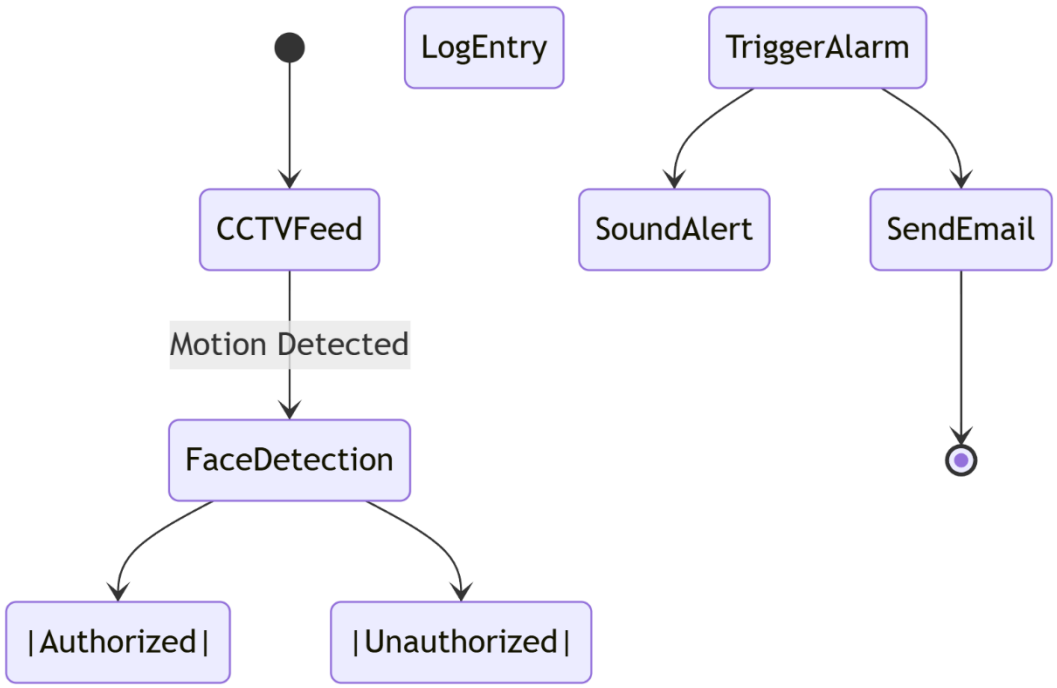


Fig 4.3

Elements:

Swim lanes:

Add actors (Admin, System) if needed (ref. Abdomen Page 24).

Loops:

Add Retry Detection on failure (like Abdomen's Page 31).

3.a) ER Diagram :

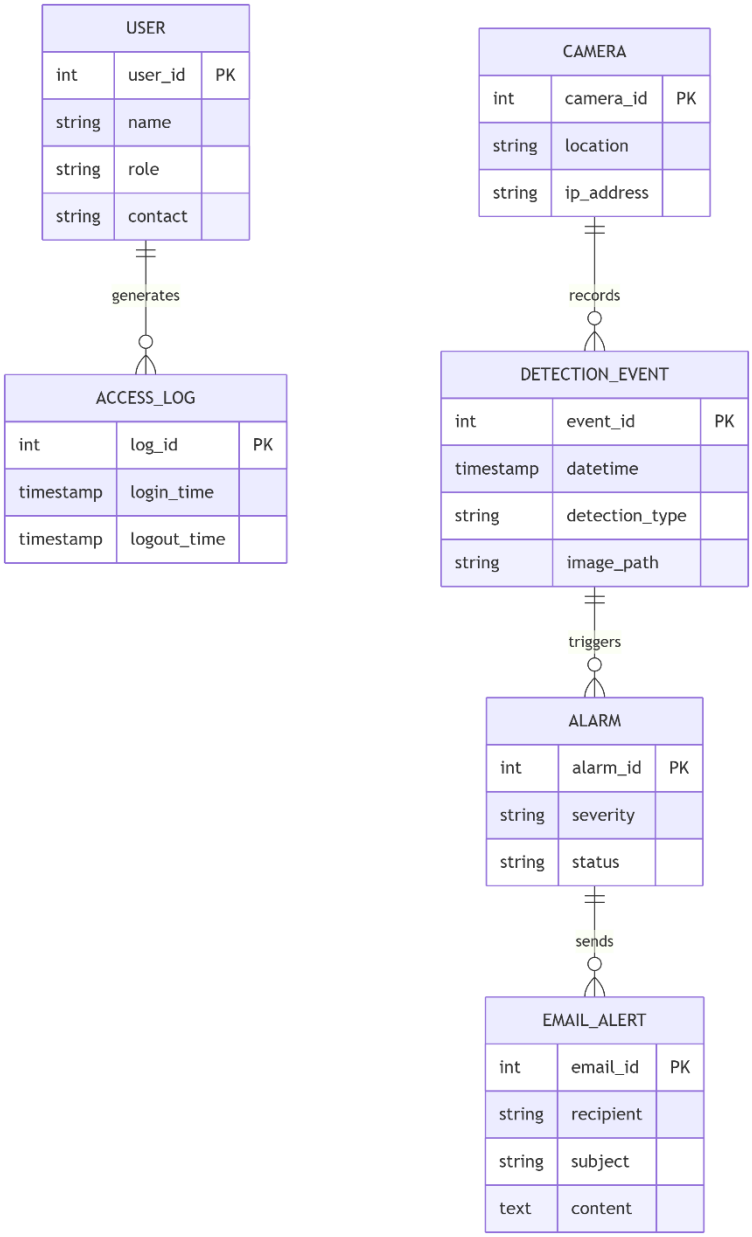


Fig 4.4

Key Components:

Entities (Inspired by Abdomen's ERD on Page 26):

USER:

Stores admin/security personnel details (like Abdomen's "Patient" entity).

CAMERA:

IP cameras with locations (analogous to Abdomen's "CT Scanner" entity).

DETECTION_EVENT:

Records face detection triggers (similar to Abdomen's "Diagnosis" entity).

ALARM & EMAIL_ALERT:

Mirror Abdomen's "Alert" system (Page 5) but with security context.

Relationships:

USER generates ACCESS_LOG (1-to-many).

CAMERA records DETECTION_EVENT (1-to-many).

DETECTION_EVENT triggers ALARM (1-to-1).

Attributes:

Primary keys (PK) for all entities (like Abdomen's ERD conventions).

Timestamps for audit trails (critical in both medical and security systems).

3.b)ER for Database Structure

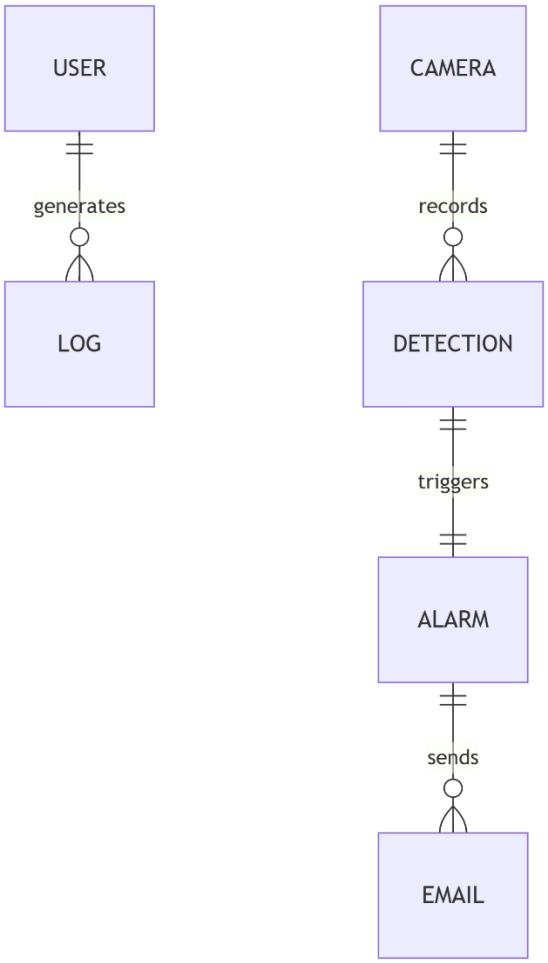


Fig 4.5

Tables:

USER:
user_id (PK), name, role

DETECTION: event_id (PK), timestamp, camera_id (FK)

4) <u>Data Flow Diagram :</u>

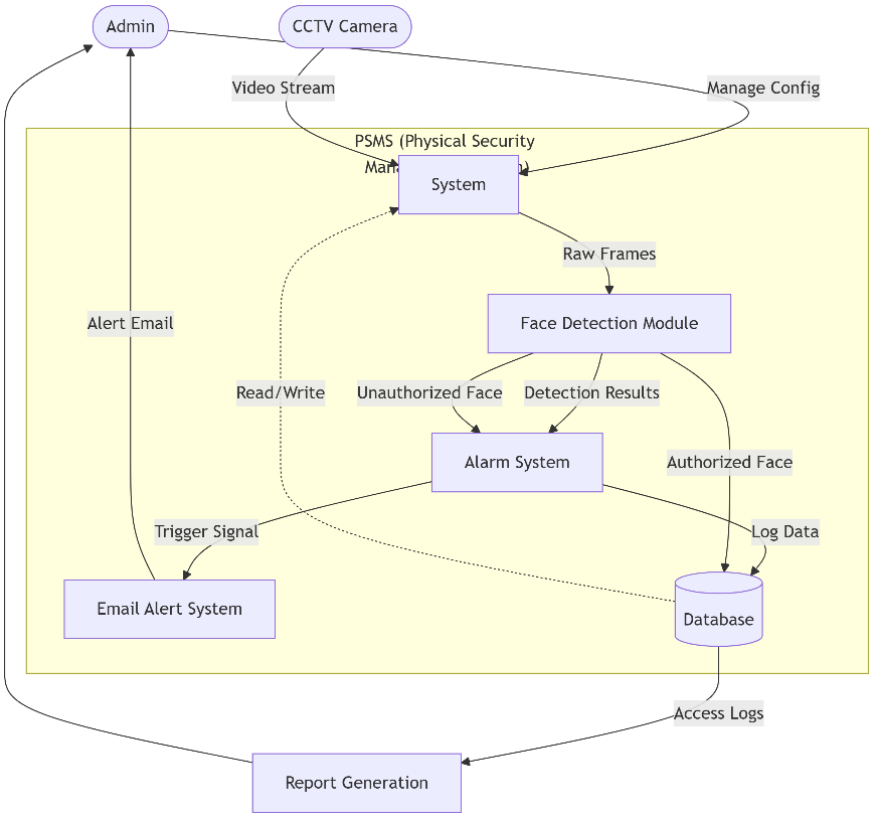


Fig 4.6

Key Components:

External Entities (Rectangles):

Admin:

Manages system configuration.

CCTV:

Provides video input.

Processes (Rounded Rectangles):

Face Detection:

Processes frames using OpenCV (like Abdomen's image processing).

Alarm System:

Triggers alerts (mirrors Abdomen's alert logic).

Email System:

Sends notifications (ref. Abdomen's email template).

Data Stores (Open Rectangles):

Database:

Stores logs (similar to Abdomen's data handling).

Data Flows (Arrows):

Video → Detection → Alarms → Email (linear flow).

Parallel logging to database.

5) Sequence Diagram :

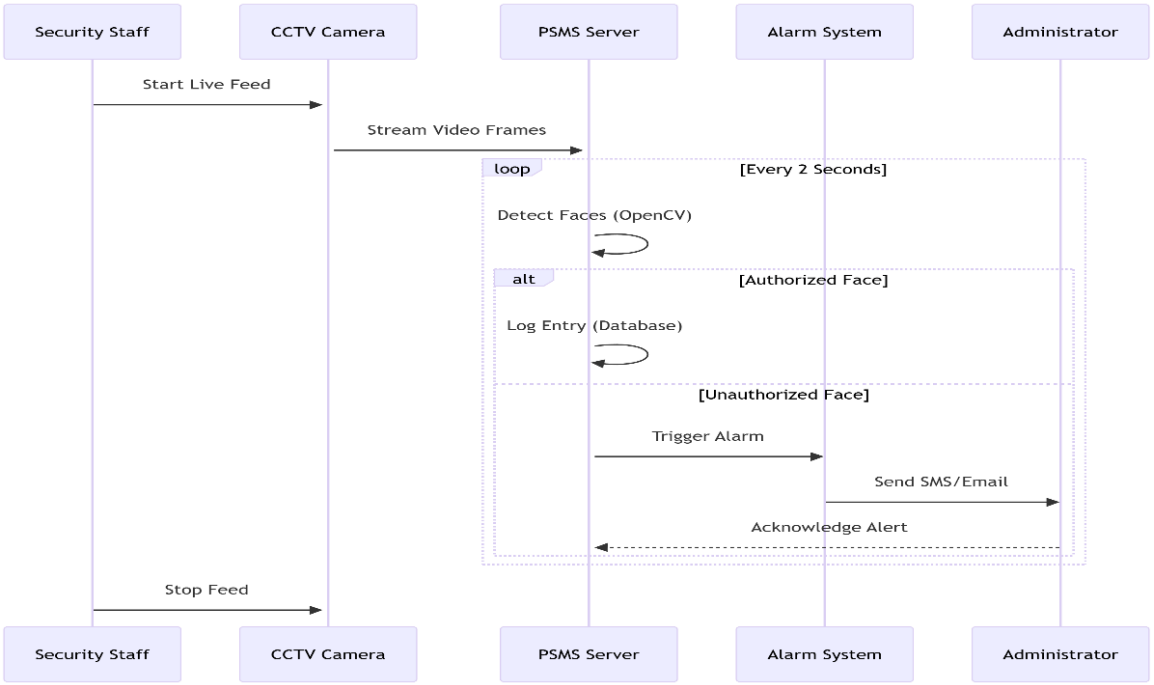


Fig 4.7

Key Interactions:

Staff starts camera feed

System checks frames for faces

If intruder found → Alarm triggers → Admin gets alert

4 Use Case Diagram:

1. User Auth & Dashboard-
   a) Secure login/registration with centralized monitoring (CCTV, logs).

2. Real-Time Security-
   a. Camera access, automated alarms, and access control (door/gate triggers).

3. Data & Reporting-
   a. Logs all events (timestamps, alerts) and generates exportable reports.

4. Scalable Design-
   a. Modular backend (PHP/Python) and responsive UI for future upgrades.

6) Use Case Diagram:

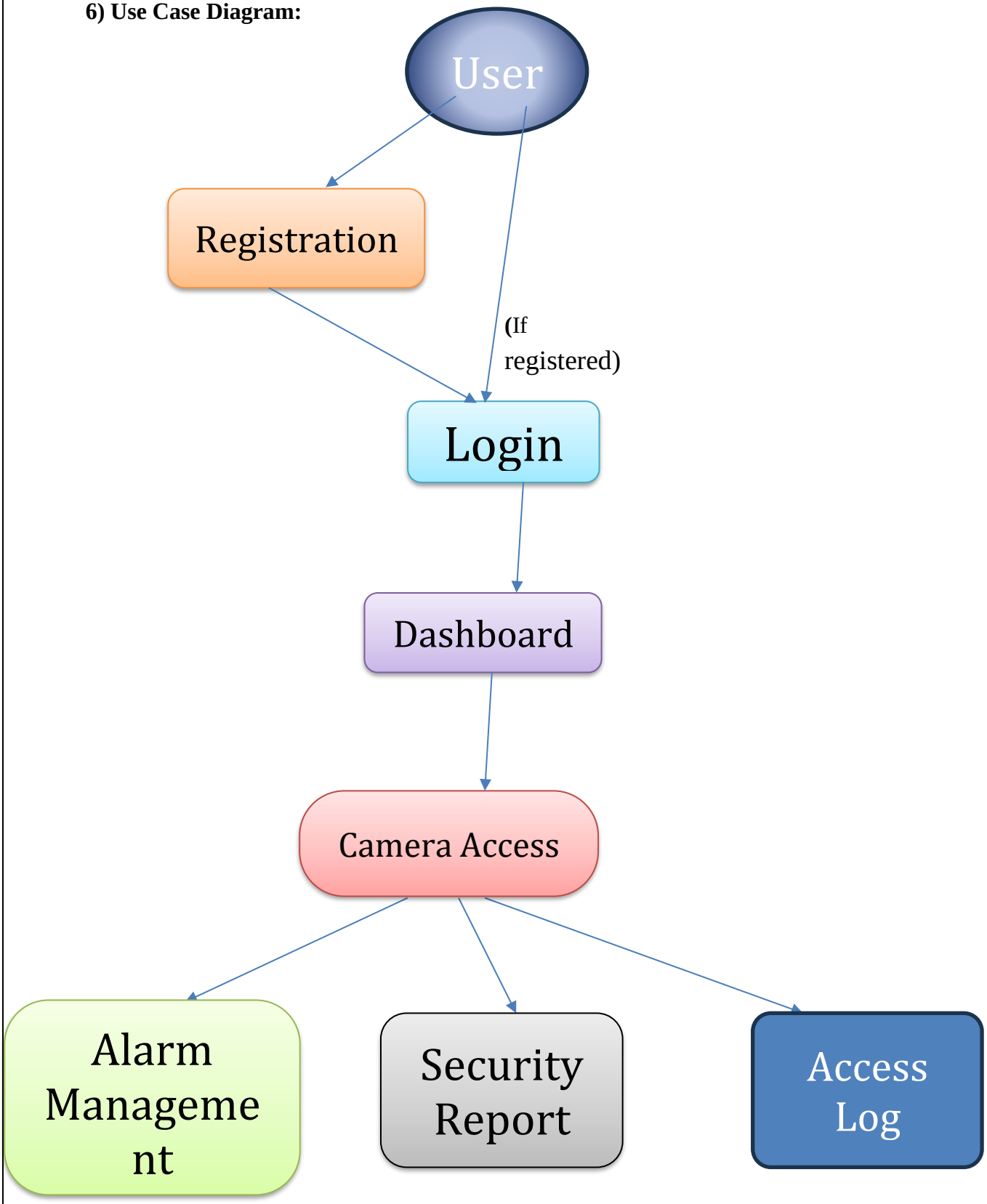


Fig 4.8

Chapter 5

Chapter 5:
Implementation and Testing

4.1 Implementation Approaches

The Physical Security Management was implemented using a full-stack approach combining frontend and backend technologies, supported by database connectivity and scripting. The system follows modular development principles and was built using HTML, CSS, JavaScript, PHP, Python, and MySQL.
The major implementation components are described below:

1) Frontend Design (HTML, CSS, JavaScript)

a) HTML was used to create the layout of the user interface, including login, registration, and dashboard pages.

b) CSS provided styling to enhance the user experience, including a background image, form styling, and responsive design.

c) JavaScript was used for basic client-side form validation and user feedback without reloading the page.

2) Backend Logic (PHP & Python)

a) PHP was used to handle core backend operations such as user registration, login authentication, session management, and role-based redirection.

b) Python was used for auxiliary backend scripting, such as:
   i) Managing user access logs
   ii) Performing data formatting or exporting logs
   iii) Running background scripts for admin analysis (e.g., generating reports or summary statistics)

c) PHP and Python worked in harmony, with PHP handling web interactions and Python executing logic-heavy background operations when needed.

3) Database Layer (MySQL)

a) MySQL was used as the database system to store all critical information including:
   i) User credentials (username, hashed password, role)
   ii) Login and access attempt logs

b) The database was connected using MySQLi in PHP and MySQL Connector in Python for advanced data operations.

4) Authentication and Authorization

a) The system supports secure user login with hashed passwords using password_hash() and password_verify().

b) Based on the user's role (admin or user), they are redirected to the appropriate section of the system.

c) Unauthorized access to admin areas is restricted using session checks.

5) Session Management

a) PHP sessions are used to track user login status and restrict access to internal pages.

b) Sessions are automatically destroyed on logout, improving security.

6) Security Implementation

a) Passwords are hashed before storage, preventing exposure even if the database is compromised.

b) Input sanitization and prepared statements are used to avoid SQL injection attacks.

c) Access to admin functionality is protected by both frontend and backend checks.

7) Error Handling and User Feedback

a) JavaScript provides real-time form validation.

b) PHP and Python scripts handle backend exceptions and return meaningful error messages to users.

c) Alerts are used to inform users about success, failure, or invalid input.

8) UI/UX and Responsiveness

a) Forms and pages are designed to be clean and responsive using CSS.

b) The layout adjusts to different screen sizes for mobile, tablet, and desktop users.

c) Visual enhancements like shadows, rounded corners, and color schemes improve usability.

5.2 Code details and code efficiency

The Physical Security Management has been developed using clean, structured, and modular code practices. The system consists of multiple components built with HTML, CSS, JavaScript, PHP, Python, and MySQL. Each module is logically separated, reusable where possible, and designed for simplicity and maintainability.

1) Code Structure

a) Frontend:

i) HTML and CSS are used to structure and style the user interface. The forms (registration, login) are responsive and mobile-friendly.

b) Backend:
   i) PHP handles user interactions, form submissions, authentication, session management, and role-based redirection.

   ii) Python is used to process backend scripts such as log formatting, data processing, and (optionally) exporting admin reports.

c) Database:
   i) MySQL is used to store user data, roles, and access logs.

2) Important Code Files

a) register.php
   i) Accepts user data from the registration form.
   ii) Uses password_hash() to securely store passwords.
   iii) Stores the user role as 'user' or 'admin'.
b) login.php
   i) Verifies credentials using password_verify().
   ii) Uses prepared statements to prevent SQL injection.
   iii) Logs login attempts to the access_logs table.
   iv) Redirects users based on their role (admin.php or dashboard.php).
c) Dashboard.php
   i) Stores database connection settings.
   ii) Included in all PHP files for consistent access.
d) admin.php
   i) Displays access logs to admin users.
   ii) Protected via session-based access control.
   iii) Shows data in a table format using PHP-MySQL queries.
e) logout.php
   i) Destroys the session and redirects to the login page.
f) background_log.py (optional)
   i) Python script used to fetch, clean, or export login records.
   ii) Helps generate insights or summary reports for the admin.

3) Code Efficiency

a) Reusability:
   i) config.php is used throughout for centralized DB connection.
   ii) CSS is separated and reused across all pages for consistent design.
b) Performance:
   i) Lightweight HTML and CSS enhance page load speed.
   ii) Login attempts are handled with optimized queries.
   iii) Only necessary data is fetched from the database to reduce overhead.
c) Security:
   i) Uses password hashing and prepared statements.
   ii) Ensures only logged-in users can access protected pages.
d) Readability:
   i) Code is indented and commented clearly.
   ii) Functions are logically grouped for easy debugging.

4) Technologies & Functions Used

a) HTML5, CSS3 for layout and design
b) JavaScript for form validation
c) PHP functions: password_hash(), password_verify(), mysqli_connect(), session_start()
d) Python libraries: mysql.connector (optional)
e) SQL operations: INSERT, SELECT, UPDATE with WHERE conditions and prepared statements

5.3 Testing Approach

The testing phase is critical to ensure that all components of the Security Management System work as expected and meet the intended requirements. A systematic testing approach was followed to identify and fix bugs, validate workflows, and verify system security and performance.

The testing strategies applied are as follows:

1) Unit Testing

a) Each individual module was tested in isolation to ensure correct functionality:

i) register.php

(1) Tested user registration with valid and invalid inputs (e.g., empty fields, duplicate usernames).

(2) Verified that data is saved correctly in the database with hashed passwords.

ii) login.php

(1) Checked login success with correct credentials and failure with incorrect inputs.

iii) logout.php

(a) Confirmed session destruction and redirection to the login page.

iv) admin.php

(a) Tested access restriction for non-admin users and verified log display for admin users.

v)email.py

(a) When suspicious activity like unauthorized face detection occurs, an email.

2) Integration Testing

a) Modules were tested together to ensure proper communication between them:
b) Registration followed by login and redirection to appropriate dashboard.
c) Login attempts recorded and displayed in the admin panel.
d) Session maintained across different pages.

3) Validation Testing

a) All input fields were tested for required validation using both JavaScript (client-side) and PHP (server-side).

b) Examples tested include empty fields, invalid characters, and incorrect passwords.

4) **Security Testing**
   a) SQL Injection: Attempted to submit malicious SQL in input fields. Prepared statements successfully prevented injection.
   b) Session Hijacking: Verified that protected pages (like admin.php and dashboard.php) cannot be accessed without logging in.
   c) Password Security: Confirmed that passwords are stored using PHP's password_hash() and verified using password_verify().

5) **Compatibility Testing**
   a) Cross-browser testing was done on major browsers including Google Chrome, Mozilla Firefox, and Microsoft Edge to ensure consistent layout and functionality.

6) **Performance Testing**
   a) System was tested under multiple consecutive logins and form submissions to ensure stable performance.
   b) Page load time remained optimal due to lightweight frontend and efficient backend queries.

7) **User Acceptance Testing (UAT)**
   a) A sample group of users (students and peers) tested the application to check user experience, ease of navigation, and feedback alerts.
   b) All users successfully completed registration and login, confirming usability.

🧪 **Summary of Test Results**

Test Type	Status
Registration	Passed
Login & Logout	Passed
Admin Access Logs	Passed
Input Validation	Passed
Role-Based Redirect	Passed
SQL Injection Prevention	Passed
Session Protection	Passed

5.4 Modifications and Improvement

During the development and testing of the Security Management System, several enhancements and additional features were implemented to improve functionality, security, and user interaction. These modifications were based on feedback, practical testing, and ideas to increase real-world relevance.

The following key improvements were added:

1) **Alarm System Integration**
   a) An audible alarm mechanism was added to trigger when an unauthorized access attempt or detection event occurs. This enhances the physical security layer by providing instant local feedback in case of suspicious activity.
      i) Implemented using JavaScript and an embedded media alert.

ii) Automatically plays an alarm sound when a predefined detection trigger is activated.
iii) Integrated with backend logic to only activate in genuine intrusion scenarios.

2) Email Notification System

a) An automated email alert feature was developed to notify the system admin immediately when suspicious activity is detected.
b) Configured using PHP's mail() function or SMTP libraries.
c) The system sends real-time alerts to a predefined admin email address when:
   i) Multiple failed login attempts occur.
   ii) An unknown user tries to access the admin panel.
   iii) A scheduled security check is triggered.

3) Schedule Detection Module

a) A schedule-based detection system was implemented, allowing the system to automatically start surveillance or detection scripts during specific time periods.
b) Developed using Python to run background scripts (e.g., for motion detection, face recognition, or file scanning).
c) Admin can configure the start time and duration of the detection module.
d) Useful in securing premises during after-hours or specific periods.

4) Improved Role Management

a) The system initially supported basic user/admin roles. It was enhanced with better session control and admin-only access to log reports and alert settings.

5) Enhanced UI Feedback

a) JavaScript-based alert boxes and styled messages were added for better user experience.
b) Visual indicators for success/failure, and real-time form feedback, were implemented.

6) Code Optimization

a) Redundant code was removed for efficiency.
b) Database queries were optimized with indexing and better filtering.
c) Security logic was centralized for easier maintenance.

- **Future Suggestions:**
  1. Expand the alarm system to include hardware-based triggers.
  2. Integrate SMS alerts or mobile push notifications for faster response.

Chapter 6

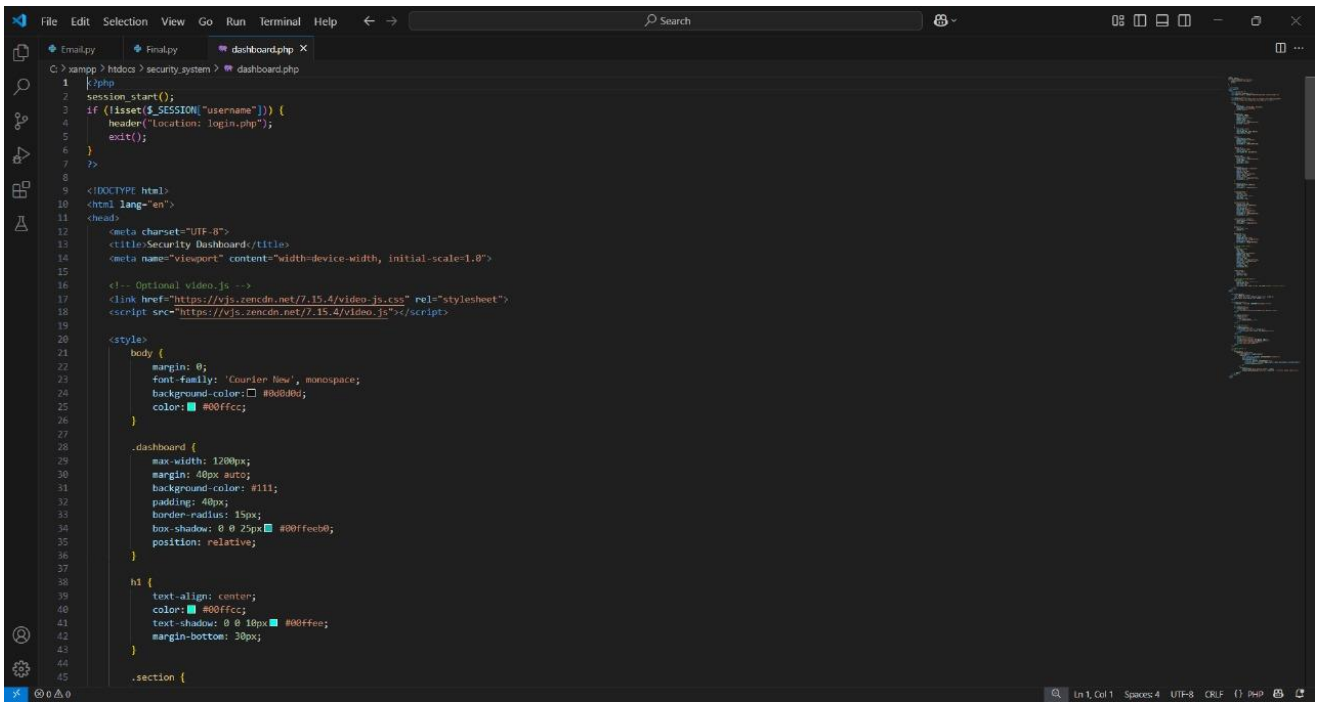
Chapter 6:
Results and Discussions

register.php

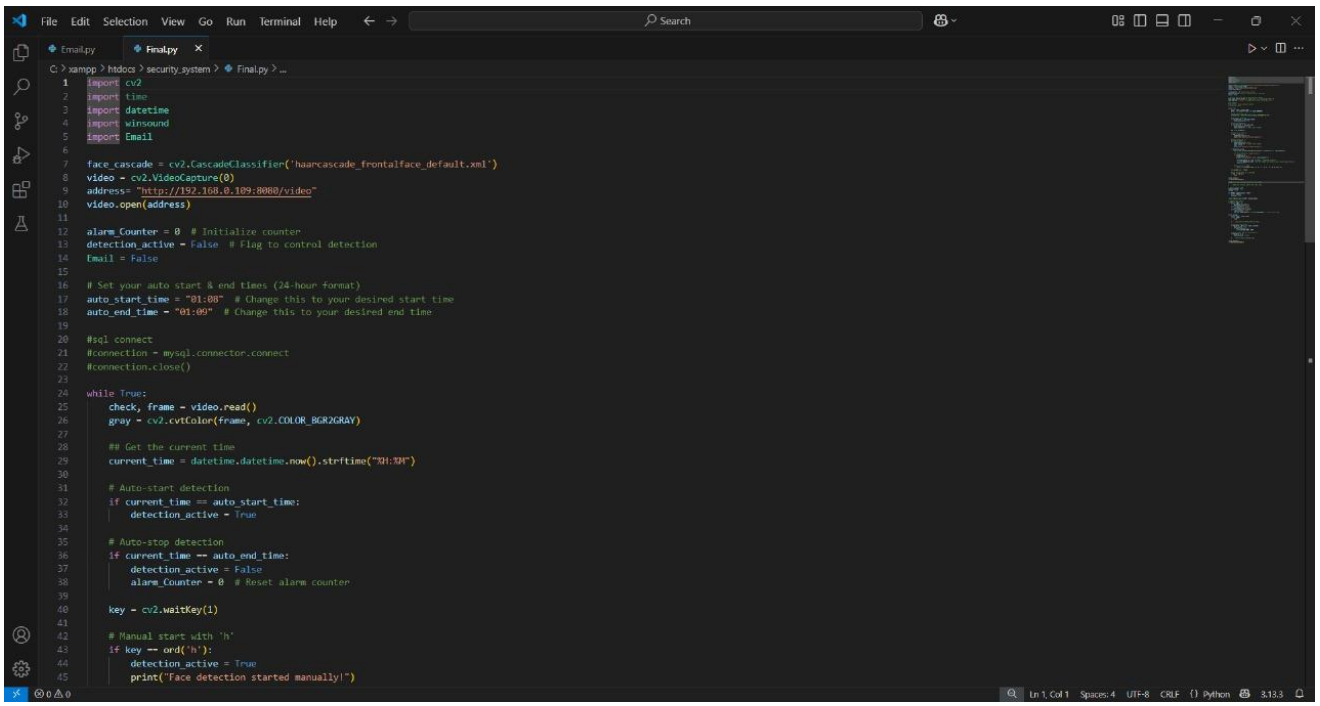
```php
<?php
include 'db.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST["username"]);
    $password = trim($_POST["password"]);

    // Validate inputs
    if (empty($username) || empty($password)) {
        die("Username and password are required");
    }

    // Hash the password
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

    // Insert using prepared statement
    $stmt = $conn->prepare("INSERT INTO users (username, password) VALUES (?, ?)");
    $stmt->bind_param("ss", $username, $hashed_password);

    if ($stmt->execute()) {
        header("Location: login.php?registration=success");
        exit();
    } else {
        die("Registration failed: " . $conn->error);
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <style>
        /* General Body Styling */
        body {
            font-family: 'Courier New', Courier, monospace;
            background: #1e1e1e;
            color: #00FF41; /* Terminal Green */
            margin: 0;
            padding: 0;
            height: 100vh;
            display: flex;
```

login.php

```php
<?php
session_start();
include 'db.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST["username"]);
    $password = trim($_POST["password"]);

    // Validate inputs
    if (empty($username) || empty($password)) {
        die("Username and password are required");
    }

    // Get user from database
    $stmt = $conn->prepare("SELECT id, username, password FROM users WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows === 1) {
        $user = $result->fetch_assoc();

        // Verify password
        if (password_verify($password, $user['password'])) {
            // Password is correct
            $_SESSION["user_id"] = $user['id'];
            $_SESSION["username"] = $user['username'];
            header("Location: dashboard.php");
            exit();
        } else {
            // Password is incorrect
            $error = "Invalid password";
        }
    } else {
        // User doesn't exist
        $error = "User not found";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

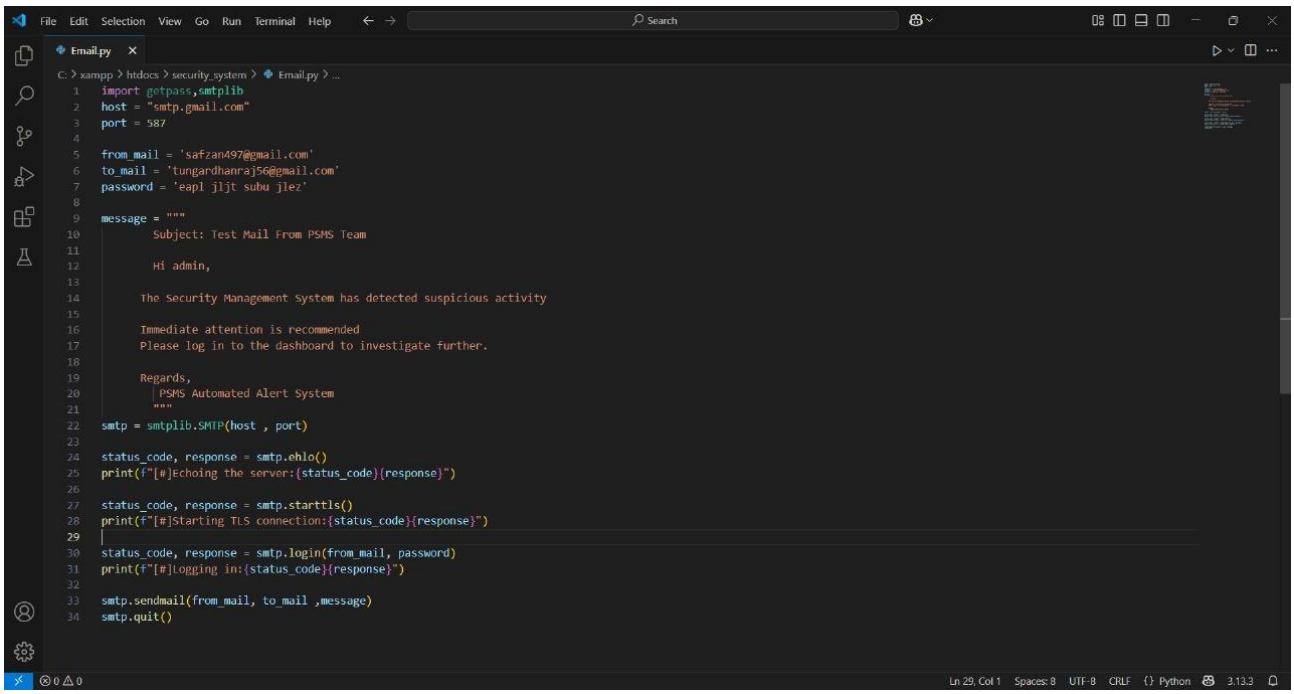


```php
<?php
session_start();
if (!isset($_SESSION["username"])) {
    header("Location: login.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Security Dashboard</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Optional video.js -->
    <link href="https://vjs.zencdn.net/7.15.4/video-js.css" rel="stylesheet">
    <script src="https://vjs.zencdn.net/7.15.4/video.js"></script>

    <style>
        body {
            margin: 0;
            font-family: 'Courier New', monospace;
            background-color: #0d0d0d;
            color: #00ffcc;
        }

        .dashboard {
            max-width: 1200px;
            margin: 40px auto;
            background-color: #111;
            padding: 40px;
            border-radius: 15px;
            box-shadow: 0 0 25px #00ffee80;
            position: relative;
        }

        h1 {
            text-align: center;
            color: #00ffcc;
            text-shadow: 0 0 10px #00ffee;
            margin-bottom: 30px;
        }

        .section {
```

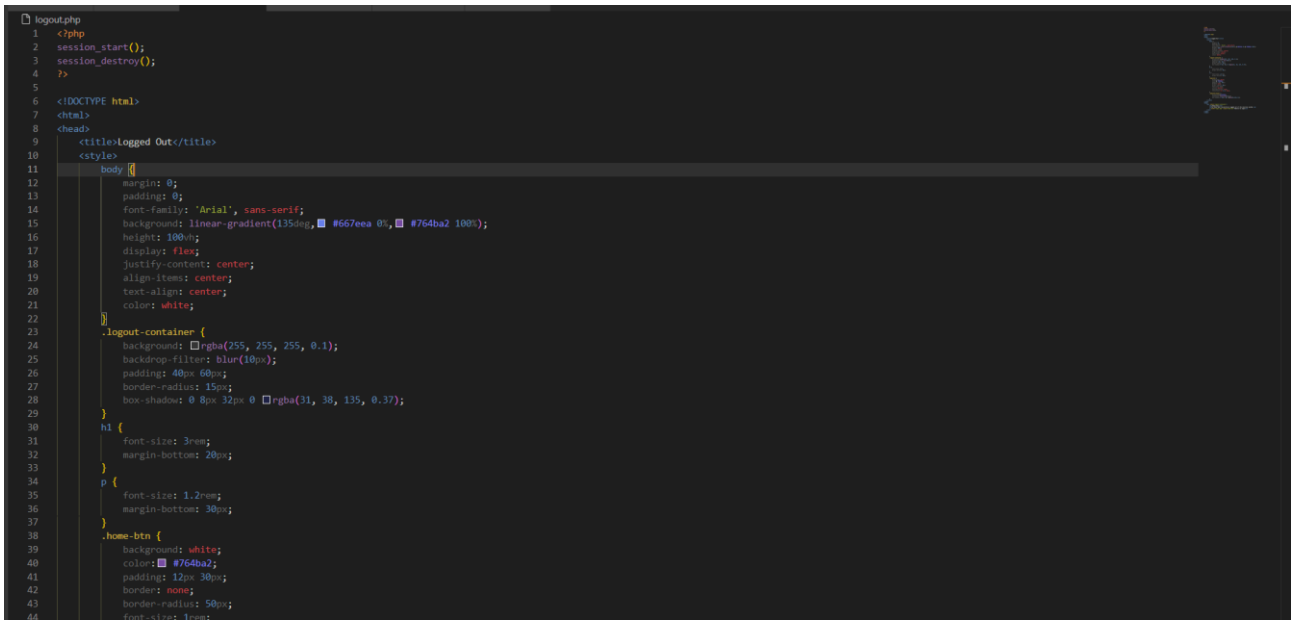


```python
import cv2
import time
import datetime
import winsound
import Email

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
video = cv2.VideoCapture(0)
address= "http://192.168.0.109:8080/video"
video.open(address)

alarm_Counter = 0  # Initialize counter
detection_active = False  # Flag to control detection
Email = False

# Set your auto start & end times (24-hour format)
auto_start_time = "01:08"  # Change this to your desired start time
auto_end_time = "01:09"  # Change this to your desired end time

#sql connect
#connection = mysql.connector.connect
#connection.close()

while True:
    check, frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    ## Get the current time
    current_time = datetime.datetime.now().strftime("%H:%M")

    # Auto-start detection
    if current_time == auto_start_time:
        detection_active = True

    # Auto-stop detection
    if current_time == auto_end_time:
        detection_active = False
        alarm_Counter = 0  # Reset alarm counter

    key = cv2.waitKey(1)

    # Manual start with 'h'
    if key == ord('h'):
        detection_active = True
        print("Face detection started manually!")
```

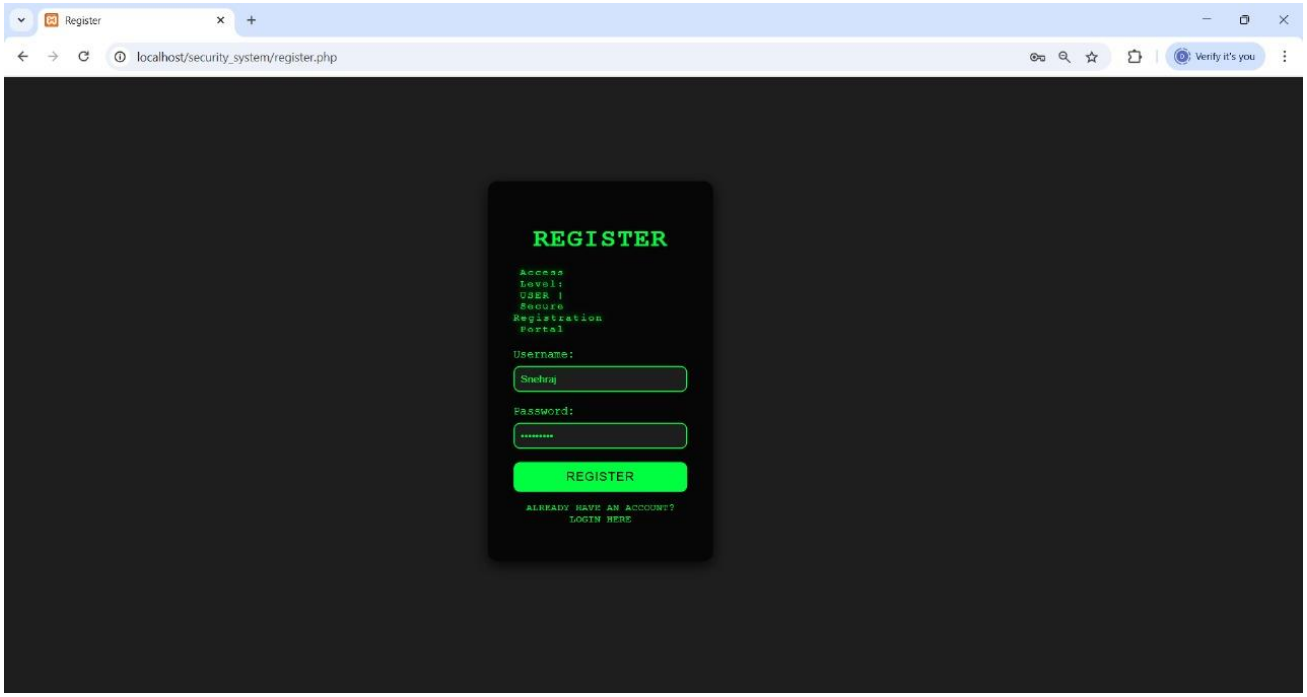
=

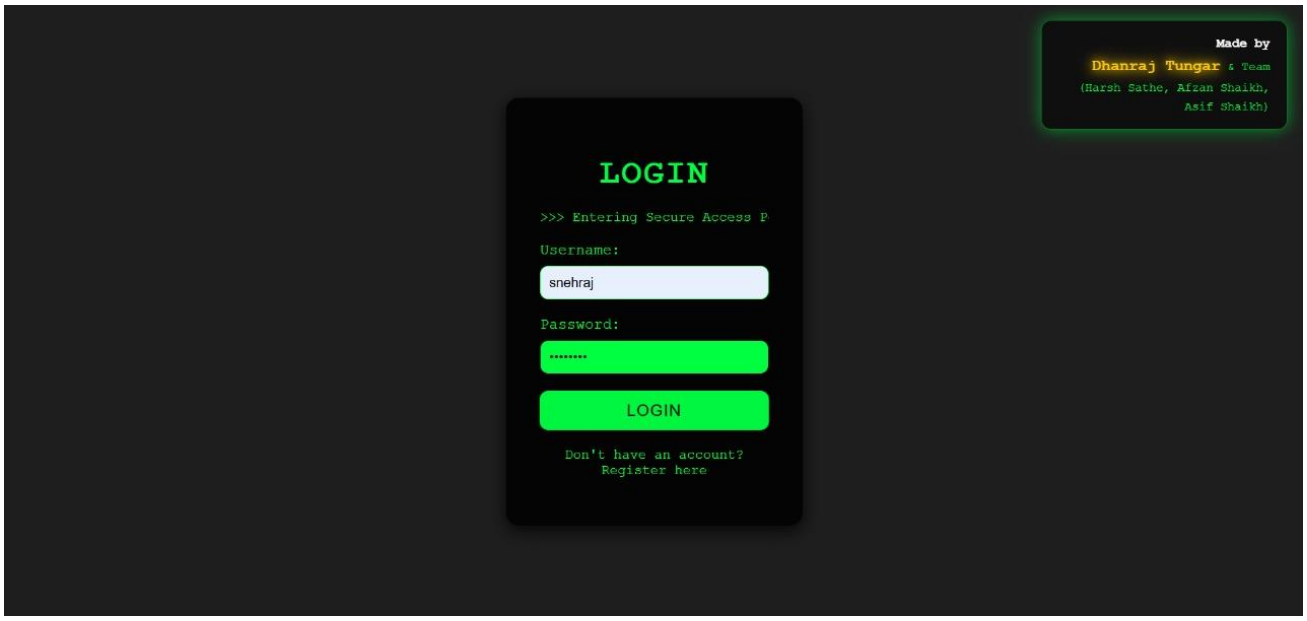


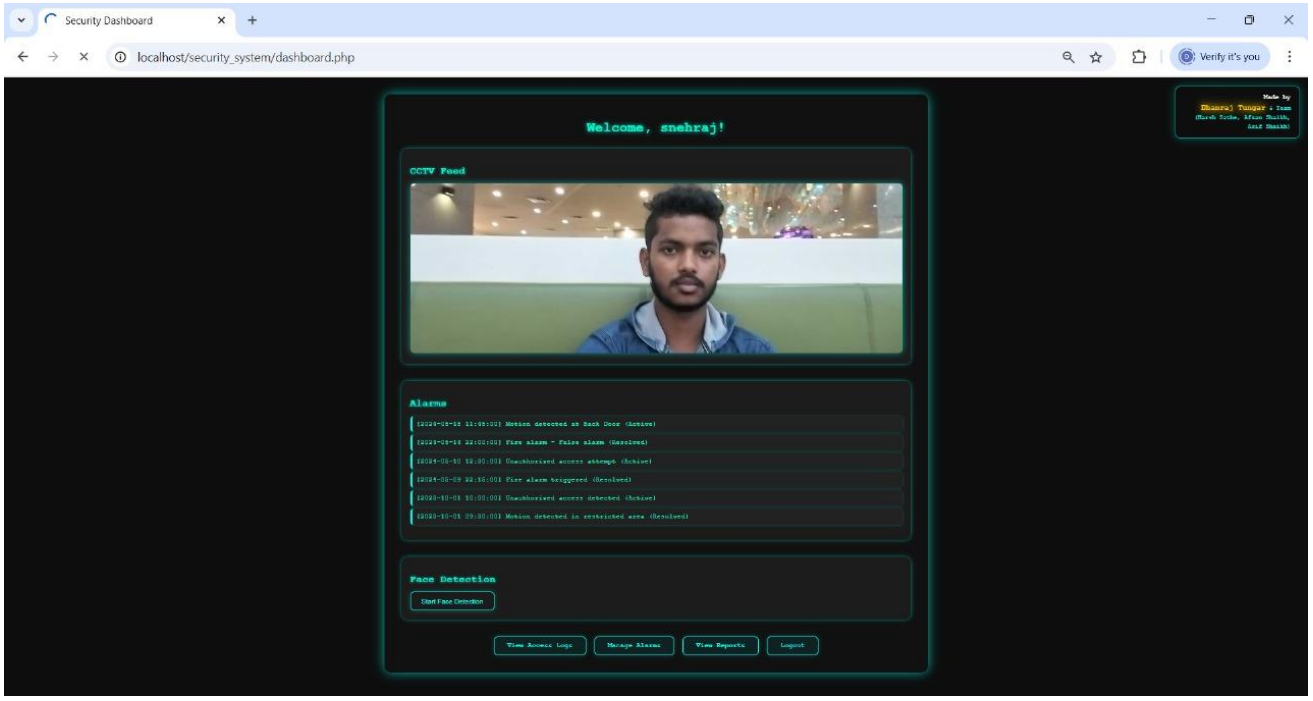
```python
import getpass,smtplib
host = "smtp.gmail.com"
port = 587

from_mail = 'safzan497@gmail.com'
to_mail = 'tungardhanraj56@gmail.com'
password = 'eapl jljt subu jlez'

message = """
        Subject: Test Mail From PSMS Team

        Hi admin,

      The Security Management System has detected suspicious activity

      Immediate attention is recommended
      Please log in to the dashboard to investigate further.

      Regards,
        PSMS Automated Alert System
        """
smtp = smtplib.SMTP(host , port)

status_code, response = smtp.ehlo()
print(f"[#]Echoing the server:{status_code}{response}")

status_code, response = smtp.starttls()
print(f"[#]Starting TLS connection:{status_code}{response}")

status_code, response = smtp.login(from_mail, password)
print(f"[#]Logging in:{status_code}{response}")

smtp.sendmail(from_mail, to_mail ,message)
smtp.quit()
```

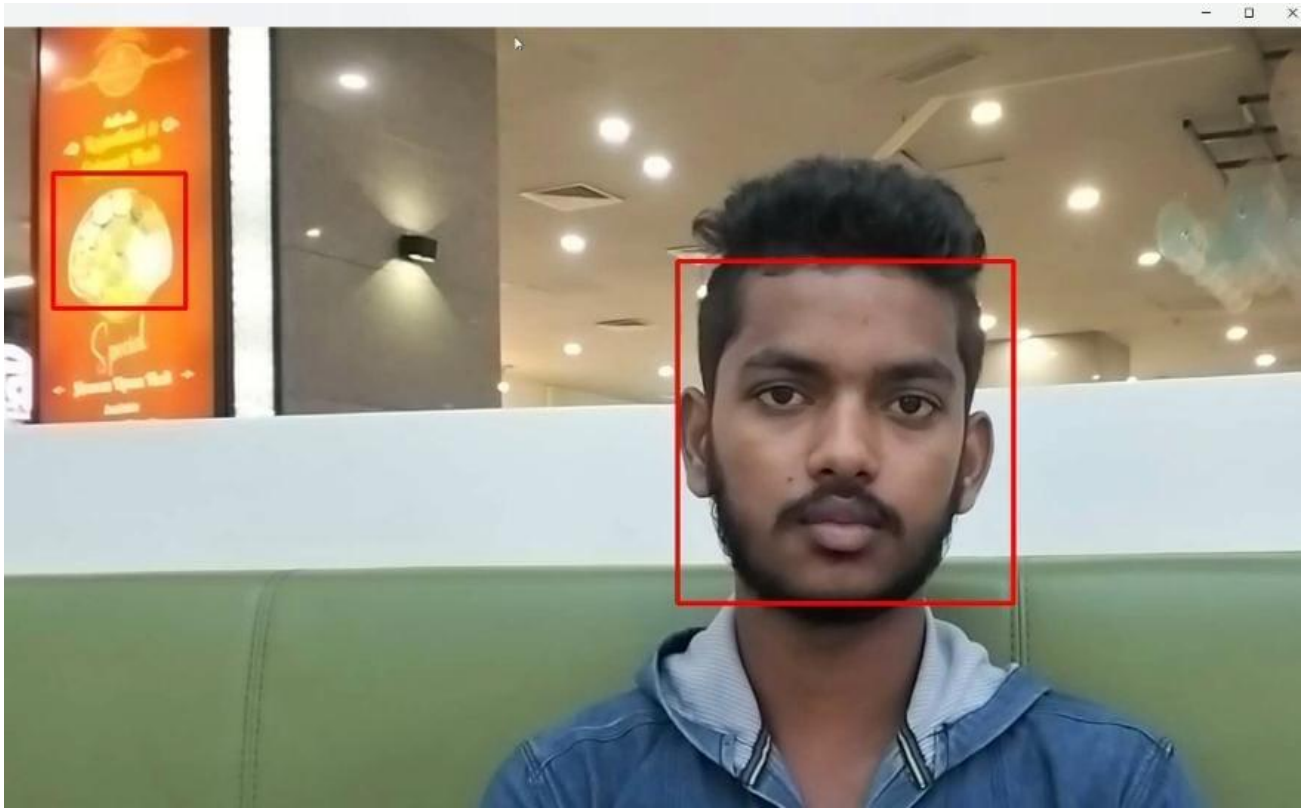


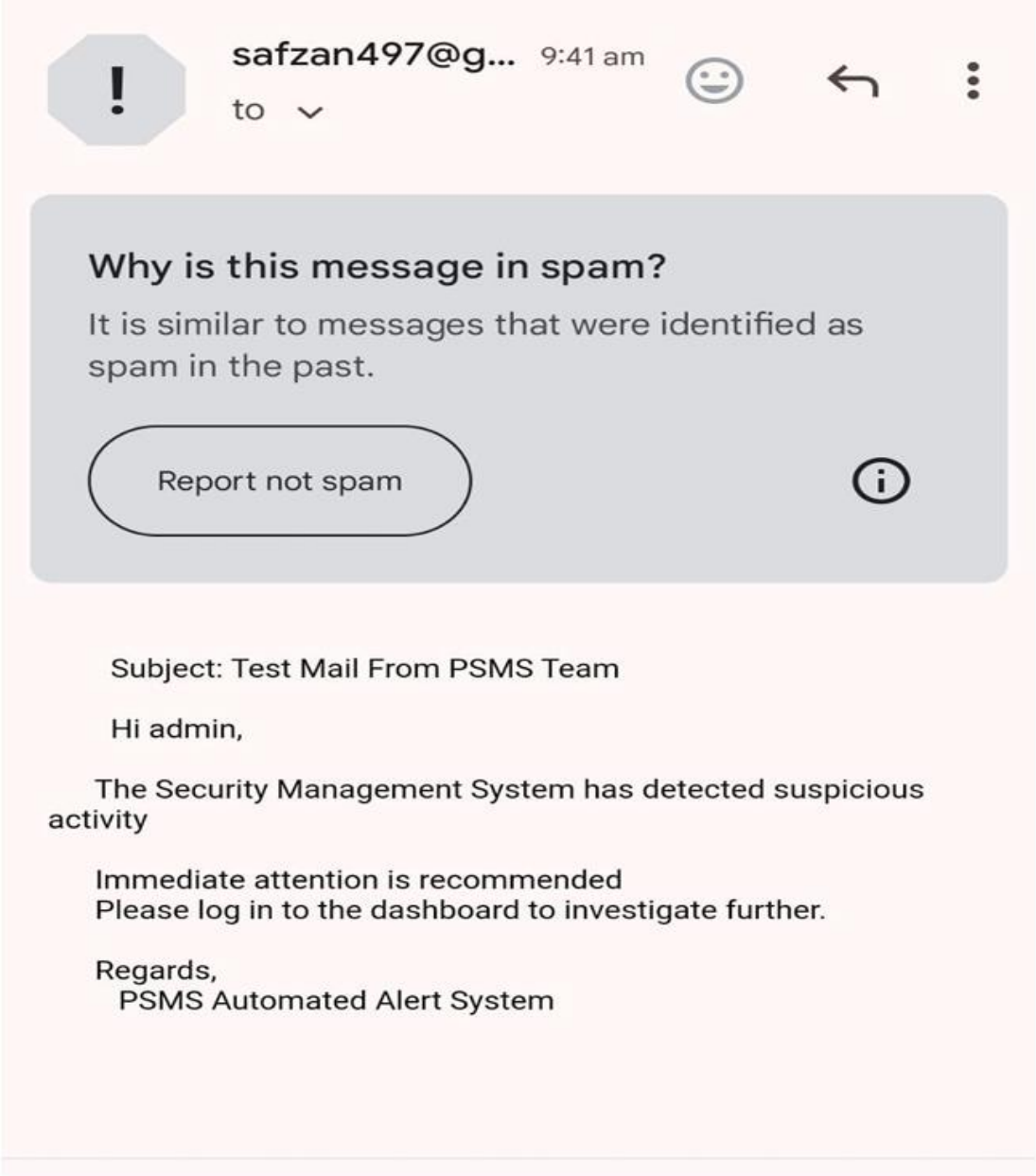
```php
<?php
session_start();
session_destroy();
?>

<!DOCTYPE html>
<html>
<head>
    <title>Logged Out</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: 'Arial', sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            text-align: center;
            color: white;
        }
        .logout-container {
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            padding: 40px 60px;
            border-radius: 15px;
            box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
        }
        h1 {
            font-size: 3rem;
            margin-bottom: 20px;
        }
        p {
            font-size: 1.2rem;
            margin-bottom: 30px;
        }
        .home-btn {
            background: white;
            color: #764ba2;
            padding: 12px 30px;
            border: none;
            border-radius: 50px;
```


Register
localhost/security_system/register.php
Verify it's you
REGISTER
Access Level: USER | Secure Registration Portal
Username:
Snehraj
Password:
REGISTER
ALREADY HAVE AN ACCOUNT? LOGIN HERE


Made by
Dhanraj Tungar & Team
(Harsh Sathe, Afzan Shaikh, Asif Shaikh)
LOGIN
>>> Entering Secure Access P
Username:
snehraj
Password:
LOGIN
Don't have an account? Register here


Security Dashboard
localhost/security_system/dashboard.php
Verify it's you
Welcome, snehraj!
CCTV Feed
Alarms
Face Detection
Start Face Detection
View Access Logs
Manage Alarms
View Reports
Logout





safzan497@g... 9:41 am

to

Why is this message in spam?

It is similar to messages that were identified as spam in the past.

Report not spam

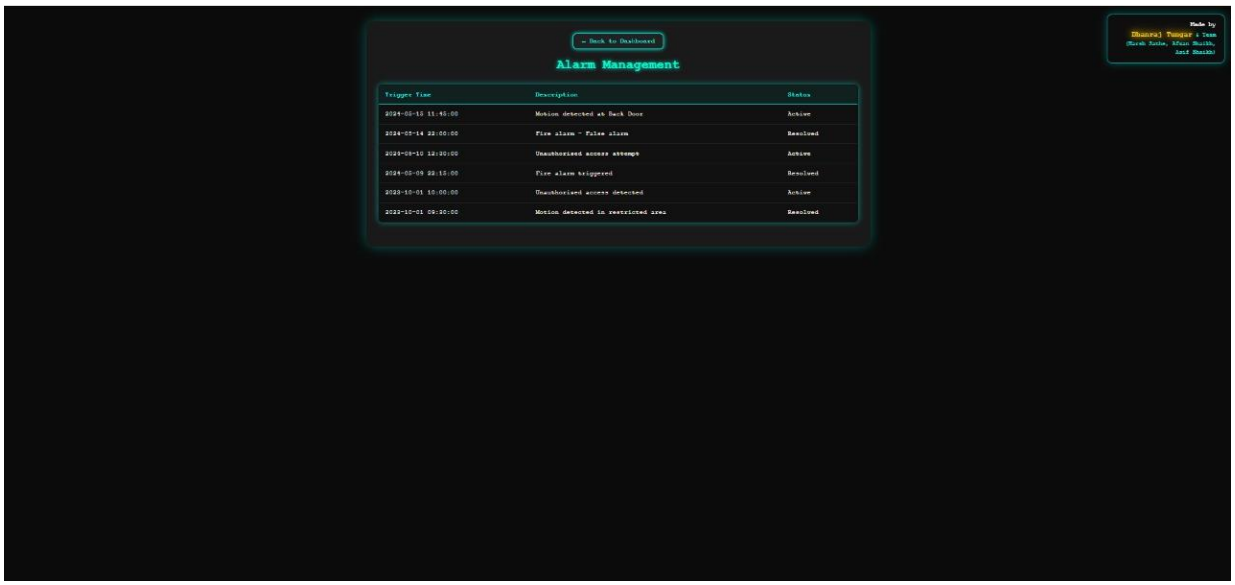
Subject: Test Mail From PSMS Team

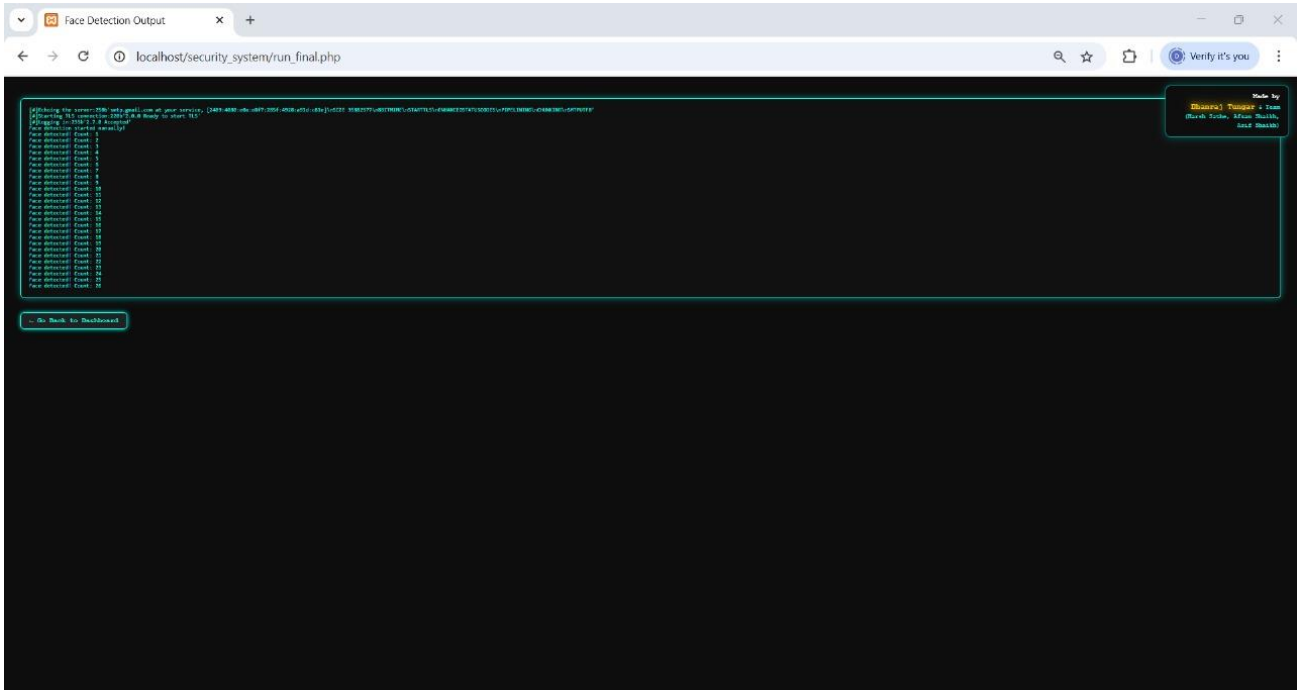
Hi admin,

The Security Management System has detected suspicious activity

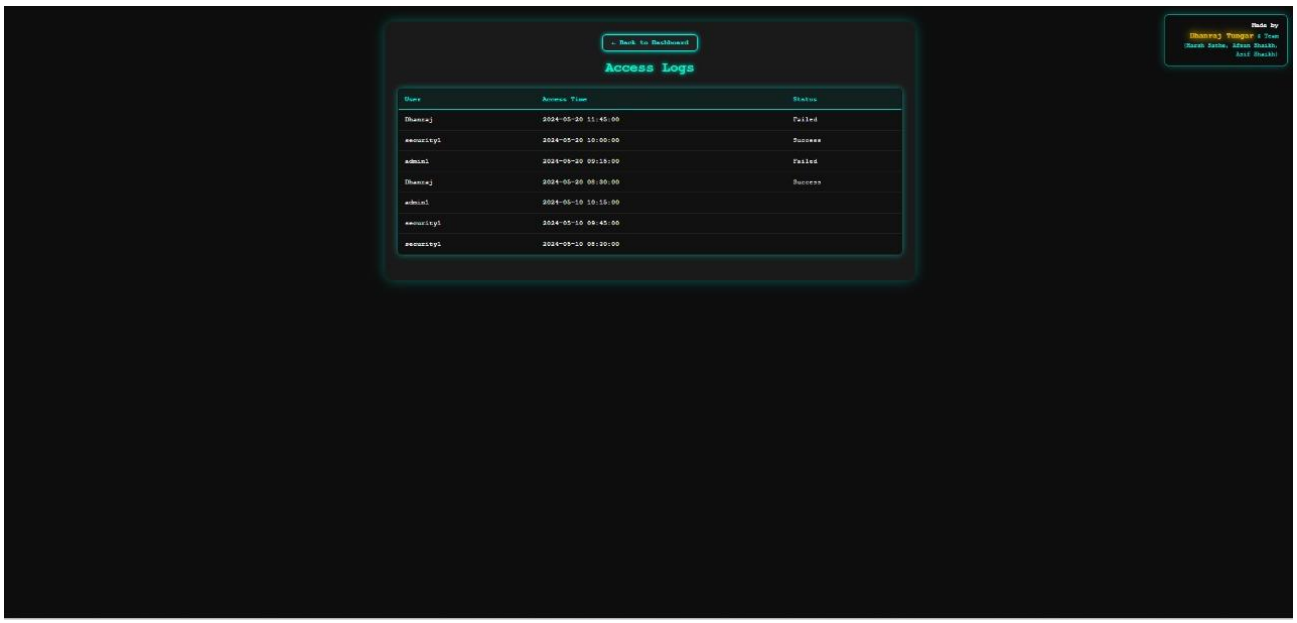
Immediate attention is recommended
Please log in to the dashboard to investigate further.

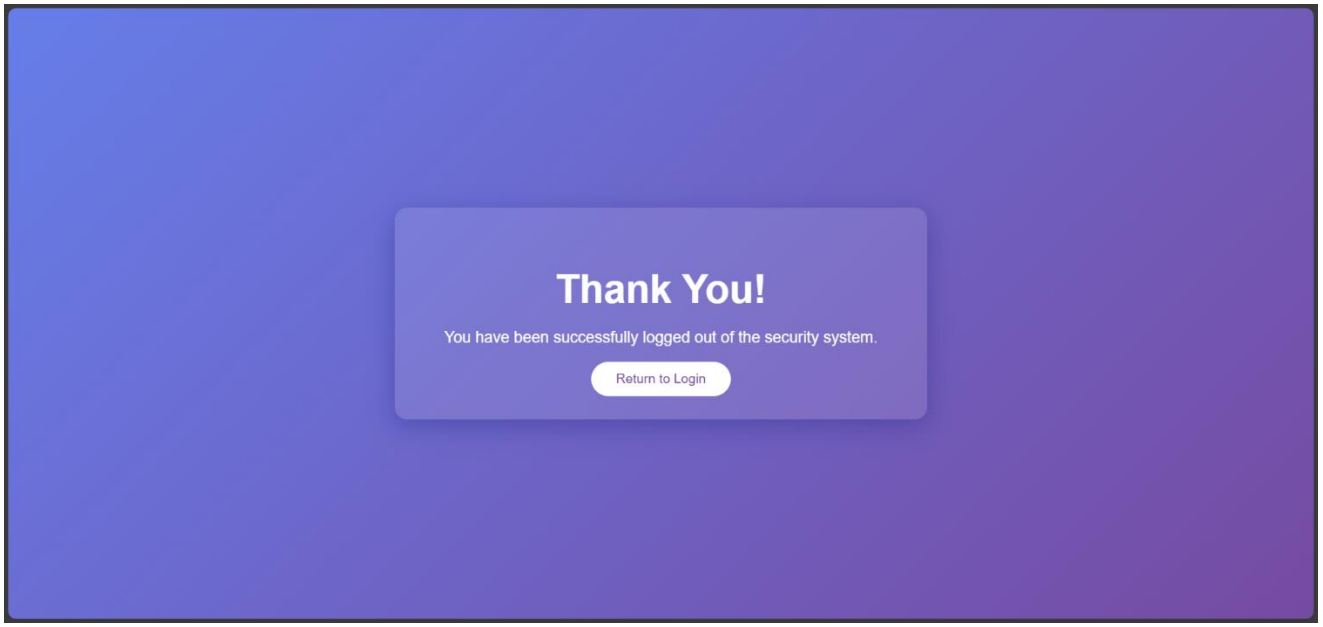
Regards,
PSMS Automated Alert System




Face Detection Output
localhost/security_system/run_final.php
Verify it's you
Made by
Dhamra] Tungar + Team
(Harsh Jatke, Afsan Shaikh,
Asif Shaikh)
← Go Back to Dashboard


← Back to Dashboard
Security Reports
Access Logs
User ID
Access Time
Status
1
2024-05-20 11:45:00
Failed
3
2024-05-20 10:00:00
Success
2
2024-05-20 09:15:00
Failed
1
2024-05-20 08:30:00
Success
2
2024-05-10 10:15:00
3
2024-05-10 09:45:00
3
2024-05-10 08:30:00
Alarm Logs
Trigger Time
Description
Status
2024-05-15 11:45:00
Motion detected at Back Door
Active
2024-05-14 22:00:00
Fire alarm - False alarm
Resolved
2024-05-10 12:30:00
Unauthorized access attempt
Active
2024-05-09 22:15:00
Fire alarm triggered
Resolved
2023-10-01 10:00:00
Unauthorized access detected
Active
2023-10-01 09:30:00
Motion detected in restricted area
Resolved
Made by
Dhamra] Tungar + Team
(Harsh Jatke, Afsan Shaikh,
Asif Shaikh)


← Back to Dashboard
Access Logs
User
Access Time
Status
Dhanraj
2024-05-20 11:45:00
Failed
security1
2024-05-20 10:00:00
Success
admin1
2024-05-20 09:15:00
Failed
Dhanraj
2024-05-20 08:30:00
Success
admin1
2024-05-19 10:15:00
security1
2024-05-19 09:45:00
security1
2024-05-19 08:30:00


Thank You!
You have been successfully logged out of the security system.
Return to Login

Chapter 7

Chapter 7
Cost and Benefit Analysis

6.1 Cost Breakdown

1) **Development Costs**
    a) **Software:** Free (PHP, MySQL,Python) or up to ₹5,000 for premium tools.
    b) **Hardware:** ₹5,000–10,000 for basic servers and testing PCs.
    c) **Labor:** ₹3,000–5,000 (student project; commercial would be higher).
    d) **Testing & Deployment:** ₹3,000–5,000 for debugging and hosting.
    - **Total Setup Cost:** ~₹15,000–25,000 (lower for academic use).
2) **Operational Costs**
    a) **Maintenance:** ₹5,000–7,000/year for updates.
    b) **Hosting:** ₹3,000–5,000/year (cloud or local server).
    c) **Security Checks:** ₹3,000–5,000/year for audits.
    - **Yearly Cost:** ~₹10,000–17,000.

6.2 Key Benefits

1) **Saves Labor Costs**
    a) Replaces 60–70% of manual monitoring (saving ~₹2 lakh/year per guard).
2) **Faster Alerts**
    a) Real-time notifications cut response time from minutes to seconds.
3) **All-in-One Management**
    a) Single dashboard for CCTV, access logs, and alarms.
4) **Scalable & Secure**
    a) Easily adds users/devices; encrypted data meets compliance.
5) **Long-Term Savings**
    a) Pays for itself in 1–2 years via efficiency gains.

Chapter 8

Chapter 8
Conclusions

The **Physical Security Management (PSMS)** successfully demonstrates how technology can improve security operations. By combining CCTV monitoring, access control, and alarms in one system, it solves key problems of traditional security methods.

1) **Key Outcomes:**

   a) Automated security monitoring

   b) Cost savings over manual systems

   c) Ready for future upgrades

2) **Future Improvements:**

   a) Mobile app for remote access

   b) AI-powered threat detection.

   c) Biometric login options

This project shows how smart technology can make security systems more efficient and affordable. While developed for academic purposes, PSMS has real-world potential for schools, offices, and other organizations.

Reference:

1. ChatGPT
2. https://chat.deepseek.com
3 https://youtube.com/@wscubetech
4. https://www.geeksforgeeks.org/