PYTHON INTRUDER ALERT SYSTEM

An Application Development-1 Report Submitted
In partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
In
Computer Science and Engineering - Artificial Intelligence and Machine Learning

By

DEEPIKA CHINDAM (22N31A6646)
D. LOKESH (22N31A6647)
DURISHETTY ANIRUDH (22N31A6648)

Under the Guidance of
Ms. B. Neelima
Professor
Department of Computational Intelligence
MRCET

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
(Affiliated to JNTU, Hyderabad)
ACCREDITED by AICTE-NBA
Maisammaguda, Dhulapally post, Secunderabad-500014.

(2021-2025)

DECLARATION

We hereby declare that the project entitled "PYTHON INTRUDER ALERT SYSTEM" submitted to Malla Reddy College of Engineering and Technology, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering - Artificial Intelligence and Machine Learning is a result of original research work done by us. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

DEEPIKA CHINDAM (22N31A6646)

D. LOKESH (22N31A6647)

DURISHETTY ANIRUDH (22N31A6648)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)
Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

CERTIFICATE

This is to certify that this is the bonafide record of the project titled **"Python Intruder Alert System"** submitted by **Deepika Chindam (22N31A6646), D. Lokesh (22N31A6647) and Durishetty Anirudh (22N31A6648)** of B.Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering - Artificial Intelligence and Machine Learning**, Dept. of CI during the year 2023-2024. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

INTERNAL GUIDE
Ms. Neelima
(Professor)

HEAD OF THE DEPARTMENT
Dr. D. Sujatha
(Professor, HOD)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director
Dr. VSK Reddy who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal, ***Dr. S. Srinivasa Rao*** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department, ***Dr. D. Sujatha*** for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our application development guide as well as our internal guide ***D. Chandra Sekhar, Assoc. Professor,*** for her structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout application development work.

We sincerely thank all the ***teaching and non-teaching staff*** of the Department of Computational Intelligence, for their timely suggestions, healthy criticism and motivation during the course of our work.

We would also like to thank our ***friends*** for providing help and moral support at the right timing. With great respect and obedience, we thank our ***parents*** who were the backbone behind our deeds.

Deepika Chindam (22N31A6646)
D. Lokesh (22N31A6647)
Durishetty Anirudh (22N31A6648)

Abstract:

Upon computer startup, a Python application vigilantly monitors security logs for unauthorized access attempts. Upon detecting an incorrect password, it promptly triggers the camera to capture a photo of the intruder. This image, along with a danger alert message, is immediately emailed to the user, ensuring swift notification of security breaches. Engineered for efficiency, the application operates seamlessly in the background, minimizing system resource consumption. Continuously scanning logs for anomalies, it ensures proactive threat detection and mitigation. By integrating this Python application into the startup routine, users establish a robust security measure, responding promptly to potential breaches. This proactive approach empowers users to protect their assets effectively, enhancing overall system security and providing peace of mind knowing that their systems are actively monitored and protected against unauthorized access.

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document outlines the specifications for the development of the "Python Intruder Alert System," version 1.0. It delineates the scope of the product, detailing the functionalities and features to be included in this iteration. The purpose of this document is to provide a comprehensive understanding of the software requirements, serving as a guide for the development team throughout the project lifecycle.

1.2 Document Conventions

This document adheres to standard typographical conventions, including font styles and highlighting for clarity and emphasis. Priority levels for requirements are indicated to guide the development process, ensuring that higher-level objectives are inherited by detailed requirements.

1.3 Intended Audience and Reading Suggestions

This document is intended for various stakeholders involved in the project, including developers, project managers, testers, and documentation writers. Developers will find detailed technical specifications, while project managers can use it to track progress and allocate resources. Testers can derive test cases from the requirements outlined herein, and documentation writers can use it as a reference for user manuals.

The document is organized into sections covering different aspects of the software requirements, starting with an overview followed by detailed specifications. Readers are encouraged to begin with the overview sections to grasp the project's scope and objectives before delving into specific requirements relevant to their roles.

1.4 Product Scope

The software specified in this document is the "Python Intruder Alert System," designed to enhance system security by monitoring security logs, capturing photos upon detecting suspicious activities, and notifying users via email. It aims to provide proactive security measures, real-time detection, and automated notification, aligning with corporate goals or business strategies. For a more comprehensive understanding, readers are referred to the separate vision and scope document, which provides additional insights into the product's overarching vision and objectives.

1.5 References

This SRS document references various documents and web addresses for additional context and information. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or the vision and scope document. Complete details of each reference, including title, author, version number, date, and source or location, are provided for easy access by readers.

2. Overall Description

2.1 Product Perspective

The "Python Intruder Alert System" is a standalone security application developed to enhance system security by monitoring security logs, detecting suspicious activities, and providing real-time alerts to users. It is designed as a new, self-contained product, independent of any existing systems or product families. The system interacts with the underlying operating system to access security logs and hardware components such as the camera for capturing images. A simple diagram illustrating the major components of the system, including the security log monitor, image capture module, and email notification system, would provide clarity on the product's perspective.

2.2 Product Functions

- Monitor security logs for unauthorized access attempts.
- Capture photos using the device's camera upon detecting suspicious activities.
- Send email alerts to users containing captured images and danger notifications.

2.3 User Classes and Characteristics

User classes for the "Python Intruder Alert System" may include:

- System administrators: Users responsible for monitoring system security and responding to security incidents.
- General users: Individuals who receive email alerts and notifications regarding security breaches.

Characteristics of each user class may vary based on their technical expertise, familiarity with security protocols, and role within the organization.

2.4 Operating Environment

The software operates in the following environment:

- Hardware platform: Any system capable of running Python scripts with access to a camera device.
- Operating system: Compatible with various operating systems supporting Python, such as Windows, Linux, and macOS.
- Software components: Requires access to security logs and the ability to send emails.

2.5 Design and Implementation Constraints

Constraints for the "Python Intruder Alert System" include:

- Hardware limitations: The system's functionality may be impacted by the performance and capabilities of the underlying hardware, particularly the camera device.

-Operating system compatibility: The application must be compatible with different operating systems and versions to ensure broad accessibility.

-Security considerations: The system must adhere to security protocols to prevent unauthorized access to sensitive information.

2.6 User Documentation

User documentation components for the "Python Intruder Alert System" may include:

- User manuals: Detailed guides on installing, configuring, and using the application.
- Online help: Interactive assistance accessible within the application interface.
- Tutorials: Step-by-step instructions for performing specific tasks or troubleshooting common issues.

The documentation will be delivered in digital formats, following standard documentation delivery standards and formats.

3. External Interface Requirements

3.1 User Interfaces

The "Python Intruder Alert System" primarily interacts with users through a command-line interface (CLI). Users can execute the application from the command prompt or terminal and interact with it by entering commands and viewing output messages. The CLI provides a simple and straightforward interface for users to monitor security logs, configure settings, and receive alerts.

3.2 Hardware Interfaces

The software interacts with the following hardware components:

- Camera device: The system accesses the camera hardware to capture images upon detecting suspicious activities. It communicates with the camera device to trigger image capture and retrieve captured images for processing.
- Storage device: Captured images may be stored temporarily or permanently on a storage device, depending on the system configuration. The software interacts with the storage device to save and retrieve image files.

3.3 Software Interfaces

The "Python Intruder Alert System" may interface with the following software components:

- Operating system: The application interacts with the underlying operating system to access security logs, manage system resources, and execute system commands.
- Email client: The system communicates with an email client to send email alerts to users. It utilizes the email client's interface or API to compose and send email messages containing captured images and danger notifications.

3.4 Communications Interfaces

The software utilizes standard email protocols, such as SMTP (Simple Mail Transfer Protocol), for sending email alerts to users. It communicates with the email server using the designated SMTP port and adheres to authentication and encryption standards specified by the email service provider. Message formatting follows standard email conventions, including text-based message bodies and image attachments for captured photos. Data transfer rates and synchronization mechanisms depend on the email server's capabilities and network conditions.

4. System Features
4.1 Intruder Detection
4.1.1 Description and Priority
This feature enables the system to detect and respond to potential intrusions by monitoring security logs and capturing photos of suspicious activities. Priority: High.

4.1.2 Stimulus/Response Sequences
- User enters incorrect password.
- System detects invalid login attempt.
- System triggers camera to capture photo.
- Photo and danger alert message are sent to user via email.

4.1.3 Functional Requirements
- REQ-1: Monitor security logs for unauthorized access attempts.
- REQ-2: Detect incorrect password attempts in real-time.
- REQ-3: Activate camera to capture photo upon detection of suspicious activity.
- REQ-4: Send email containing captured photo and danger alert message to user.
- REQ-5: Ensure seamless integration into system startup routine.
- REQ-6: Maintain efficient system resource utilization.
- REQ-7: Provide configurable settings for email notifications and security monitoring.
- REQ-8: Implement error handling for invalid inputs or system failures.

4.2 Email Notification
4.2.1 Description and Priority
This feature enables the system to send email notifications to users containing captured photos and danger alerts. Priority: High.

4.2.2 Stimulus/Response Sequences
- User triggers system startup.
- System detects security breach.
- System captures photo and prepares email.
- Email is sent to user's designated email address.

4.2.3 Functional Requirements
- REQ-1: Compose email containing captured photo and danger alert message.
- REQ-2: Include user-configurable settings for email content and recipients.
- REQ-3: Implement email sending functionality using SMTP protocol.
- REQ-4: Ensure secure transmission of email content.
- REQ-5: Provide error handling for email transmission failures.
- REQ-6: Support customization of email templates and formatting.
- REQ-7: Allow users to specify preferred email notification frequency.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system should respond to security breaches within 5 seconds of detection to ensure timely action and notification.
- The application should have minimal impact on system performance, consuming no more than 5% of CPU resources during normal operation.

5.2 Safety Requirements

- The system should not pose any physical risks or hazards to users or hardware components.
- Safeguards should be in place to prevent accidental triggering of the camera or email notifications.

5.3 Security Requirements

- User identity authentication is required to access the system settings and configure email notification preferences.
- All captured photos and email communications must be encrypted to ensure data privacy and security.
- The system must comply with relevant privacy regulations, such as GDPR or HIPAA, depending on the nature of the data being processed.

5.4 Software Quality Attributes

- Reliability: The system should operate continuously without unexpected crashes or downtime, with a target uptime of 99.9%.
- Maintainability: The codebase should be well-organized and documented to facilitate future updates and maintenance tasks.
- Usability: The user interface should be intuitive and user-friendly, allowing users to configure settings and interpret alerts easily.

5.5 Business Rules

- Only authorized users with administrative privileges can access and modify system settings.
- Email notifications should be sent only to designated recipients specified by the user.
- Any changes to system configurations or settings should be logged for auditing purposes.

6. Other Requirements

6.1 Database Requirements
- The system shall utilize a database to store user preferences, system configurations, and log data.
- The database should support efficient retrieval and storage of data related to security events and captured images.
- Requirements for database design, schema, and interaction will be detailed in the Database Design Document.

6.2 Internationalization Requirements
- The system shall support multiple languages to accommodate users from diverse linguistic backgrounds.
- User interface elements, error messages, and notifications should be translatable into different languages.
- Internationalization (i18n) and localization (l10n) standards will be followed to ensure language support.

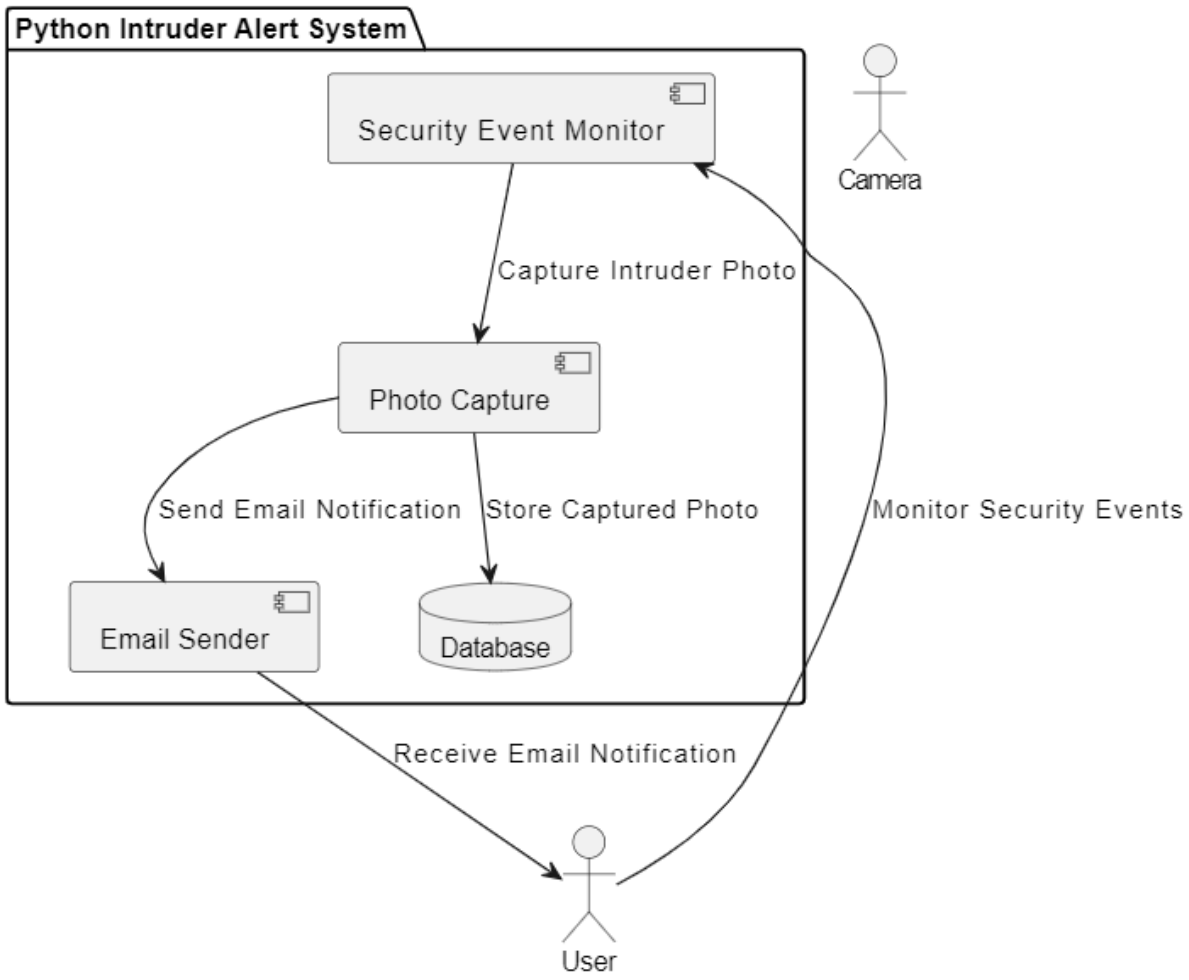
6.3 Legal Requirements
- The system must comply with relevant laws and regulations regarding data privacy and security, such as GDPR, HIPAA, etc.
- Any data collected or processed by the system must adhere to legal guidelines regarding its storage, usage, and protection.
- Legal compliance considerations will be documented and incorporated into the system design and development process.

6.4 Reuse Objectives
- The system shall incorporate reusable components and modules to facilitate future development and maintenance.
- Existing libraries, frameworks, and open-source resources should be leveraged wherever applicable to promote code reuse and efficiency.
- Reuse objectives will be documented in the Reusability Plan, outlining strategies for maximizing code reusability.

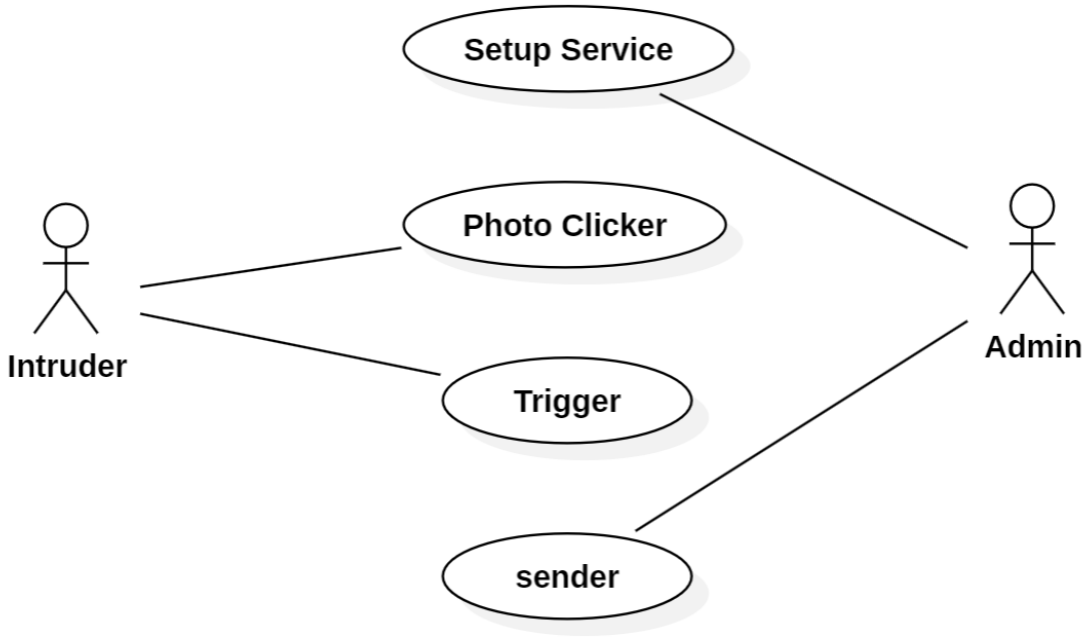
Appendix A: Glossary
1. SMTP: Simple Mail Transfer Protocol - A protocol used for sending email messages between servers.
2. SSL: Secure Sockets Layer - A standard security protocol that establishes encrypted links between a web server and a browser.
3. GDPR: General Data Protection Regulation - A regulation in EU law on data protection and privacy for all individuals within the European Union and the European Economic Area.
4. HIPAA: Health Insurance Portability and Accountability Act - A US law designed to provide privacy standards to protect patients' medical records and other health information provided to health plans, doctors, hospitals, and other health care providers.
5. UML: Unified Modeling Language - A standardized general-purpose modeling language in the field of software engineering.
6. TBD: To Be Determined - An abbreviation used to indicate that a particular detail or requirement has not yet been decided or finalized.

Appendix B: Analysis Models:

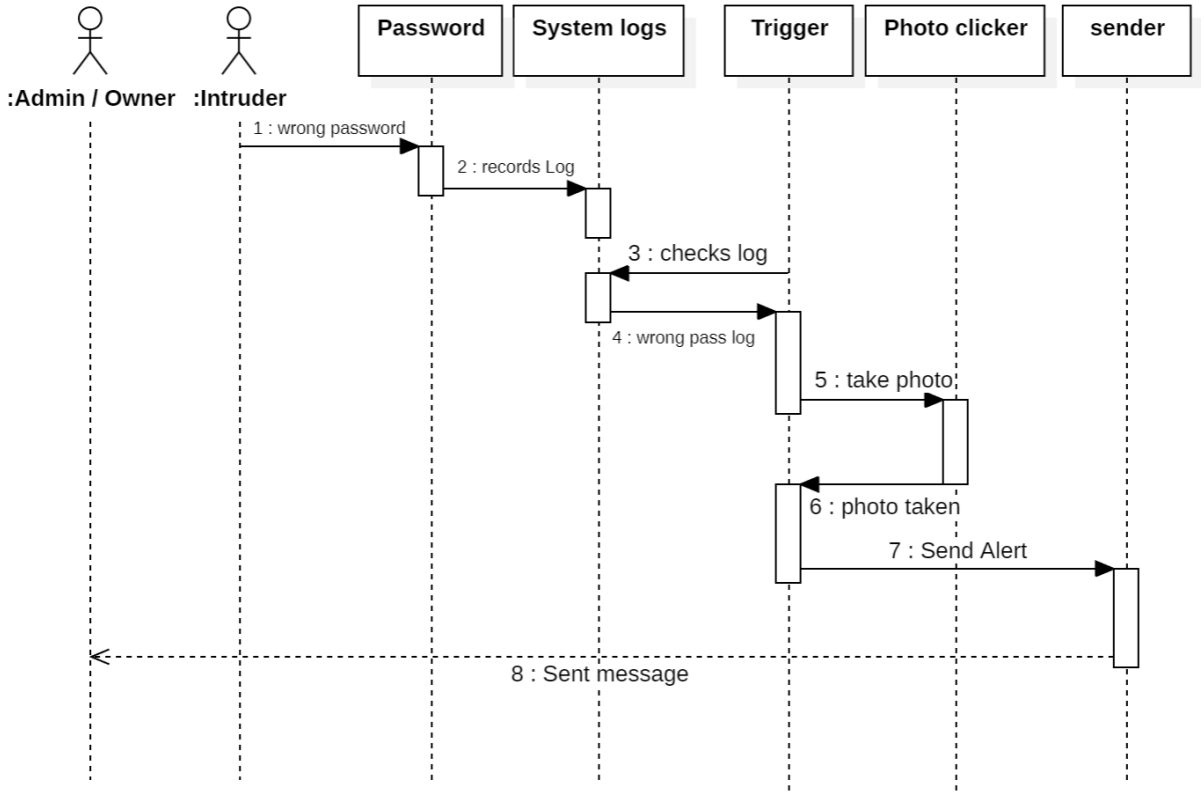
System Architecture:



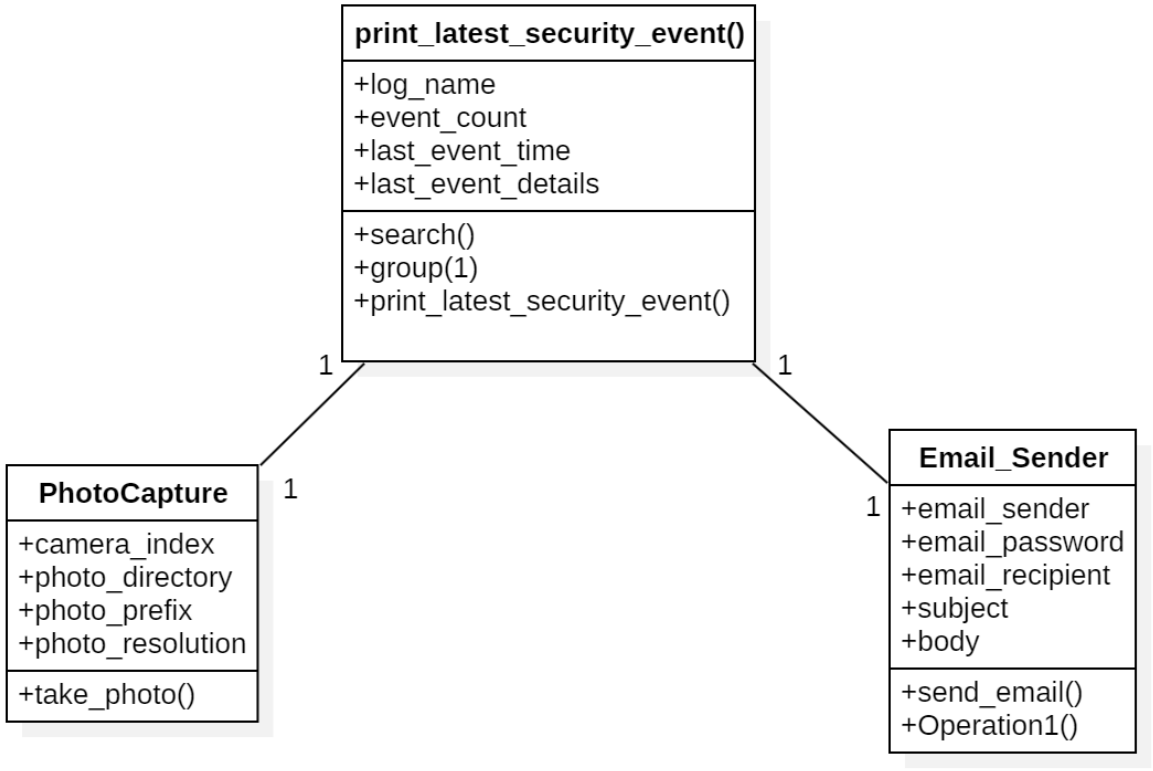
1. Use Case Diagram:



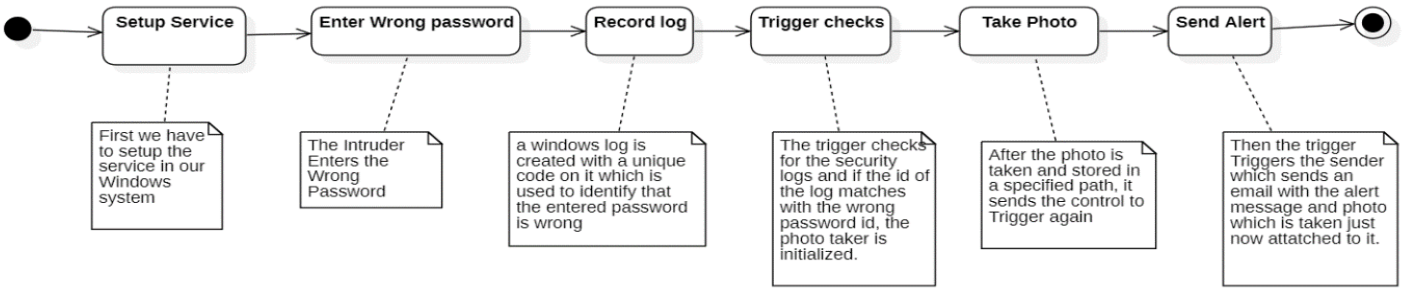
2. Sequence Diagram:



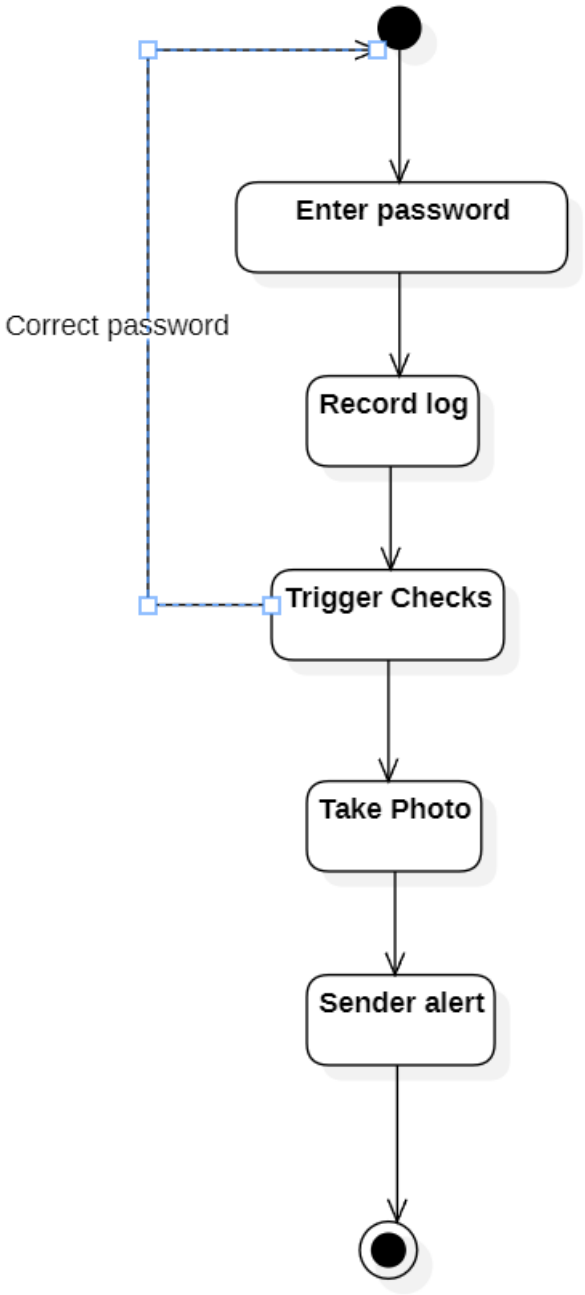
3. Class Diagram:



4. State Chart Diagram



5. Activity Diagram:



Appendix C: To Be Determined List:

1. Geolocation to the Alert
2. More triggers
3. More information required for tracing the alert
4. Live Preview of the Intruder

Source Code:

trigger.py:

```
import subprocess
import pic
import re
import time
import os
import sender
import setup
count = 0


file_path1 = "mail.txt"
file_path2 = "password.txt"

with open(file_path2, "r") as f:
    passkey = f.read()
    if passkey == "password":
        setup.setupFiles()

with open(file_path1, "r") as f:
    mailkey = f.read()
    if mailkey == "mail":
        setup.setupFiles()

def print_latest_security_event():
    global count
    command = 'powershell "Get-WinEvent -LogName Security | Select-Object -First 1 |
Format-List -Property * | Out-String"'
    try:
output=subprocess.check_output(command,shell=True, stderr=subprocess.STDOUT,
universal_newlines=True)
        f.write(output+"\n\n\n")

        # Extract event ID from the output
        event_id_match = re.search(r"Id\s+:\s+(\d+)", output)
        if count == 2:
            print("Exiting monitoring")
            exit(0)

        else:
            if event_id_match:
                event_id = int(event_id_match.group(1))
                if event_id == 4634:
                    print("Correct")
                    if count == 1:
                        count += 1
                elif event_id == 4625:
```

```
                pic.takephoto()
                sender.sendMessage()
                count+=1
            else:
                print("Unknown event ID:", event_id)
        else:
            print("Failed to extract event ID from event details.")

    except subprocess.CalledProcessError as e:
        print("Error executing PowerShell command:", e)
with open("Logs.txt", "a") as f:
    while True:
        print_latest_security_event()
```

pic.py:

```
import cv2
import subprocess
def takephoto():
    # Initialize the camera
    cam = cv2.VideoCapture(0)

    # Capture a single frame
    ret, frame = cam.read()

    # Release the camera
    cam.release()

    # Check if the frame was captured successfully
    if not ret:
        print("Frame not captured")
    else:
        # Save the captured frame as an image
        img_name = "Culprit.jpg"
        cv2.imwrite(img_name, frame)
        print("Image captured")
```

sender.py:

```
from email.message import EmailMessage
from password import user_mail, user_password
import ssl
import smtplib
import os
file_path1 = "mail.txt"
file_path2 = "password.txt"
if not(os.path.exists(file_path1) and os.path.exists(file_path2)):
    with open(file_path1, "w") as f:
        f.write("mail")

    with open(file_path2, "w") as f:
        f.write("password")
# declare sende password and reciever
with open("mail.txt","r") as em: email_sender = em.read()
with open("password.txt","r") as em: email_password = em.read()
with open("mail.txt","r") as em: email_reciever = em.read()
# declare subject body
subject = "Danger alert! Unsuccessful attempt to login"
body = """
If you watch this,
then successfully alert has been sent and its working
"""
# frame email
em = EmailMessage()
em['from'] = email_sender
em['to'] = email_reciever
em['subject'] = subject
em.set_content('image attatched')


def sendMessage():
    # attatch the image
    with open ('Culprit.jpg', 'rb') as f:
        file_data = f.read()
        file_type = 'jpeg'
        file_name = f.name
    em.add_attachment(file_data, maintype='image', subtype=file_type, filename=file_name)
    context= ssl.create_default_context()
   # send the image using smtp service
    with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
        smtp.login(email_sender, email_password)
        smtp.sendmail(email_sender, email_reciever, em.as_string())
```

Execution Screenshots:

Intruder entering wrong password:

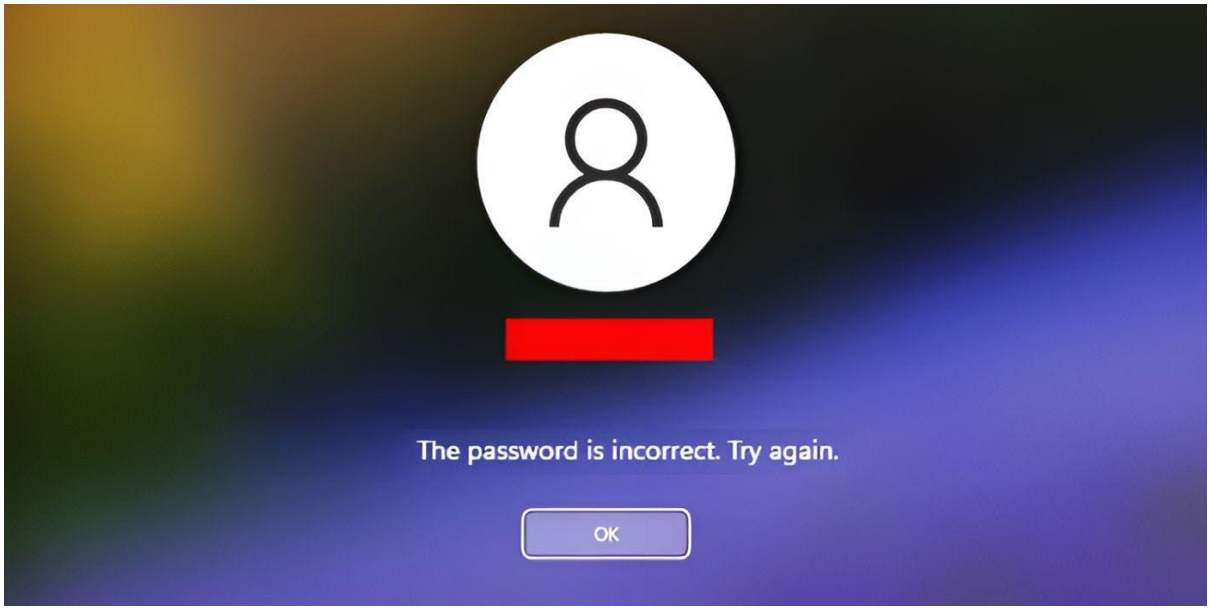


Image getting captured:

Alert notification and mail on user mobile: