PROJECT REPORT ON

USB PHYSICAL SECURITY

Submitted by

Tadi Mohana Pavani

Table of Contents

1. Introduction :

In today's digital world, USB (Universal Serial Bus) devices are used for data transfer and for storage purposes . However, this also leads to security vulnerabilities. Unauthorized USB access can lead to data breaches, malware infections, and system exploitation. Traditional software-based security such as antivirus programs fails in providing security and preventing unauthorized access or tampering.

To overcome this vulnerability and provide security, we designed USB Physical Security System to provide solutions to this problem. It monitors and control USB port usage. This system overcomes the problem of traditional system by providing access control to administrator and providing real-time intruder detection using webcam, user authentication mechanisms, and automated USB port disabling via the Windows Registry.

In the event of an unauthorized access attempt, the system captures images or videos, sends instant alerts via email to the admin, and maintains a detailed log in a structured HTML format. The entire solution is developed using Python and Tkinter for GUI, with OpenCV for webcam access, ensuring it is cost-effective, user-friendly, and platform-compatible with Windows systems.

This project enhances security by providing physical-level protection against unauthorized USB access, making it ideal for use in personal computers, academic labs, and corporate environments where data security is important.

2. Existing System

In the current scenario, computers rely mainly on **software-based security measures** like antivirus programs, firewalls, or basic operating system permissions to protect against threats from USB devices.

- USB ports are **always active** and accessible unless manually disabled.
- Antivirus software detects known malware but may **fail to stop any threats** introduced through USB drives.
- There is **no physical-level access control** — any user can plug in a USB drive and potentially transfer or steal data.
- No **real-time monitoring or alert system** for unauthorized physical access.
- No mechanism for **capturing evidence** (like images or videos) if unauthorized use occurs.

As a result, sensitive environments like corporate offices, academic labs, or personal PCs remain **vulnerable to data theft, malware infections, and hardware misuse**.

---

3. Proposed System

The **USB Physical Security System** addresses the limitations of the traditional system by combining **hardware monitoring**, **access control**, and **real-time intrusion detection** in an integrated solution.

Key features of the proposed system:

- **Administrator-controlled access:** Only authorized users can enable or disable USB ports.
- **Real-time monitoring:** Uses a webcam (via OpenCV) to capture photos or video if someone tries to access a USB port without permission.
- **User authentication:** The system verifies users before allowing USB port access.
- **Automatic USB disabling:** Controls USB ports programmatically through Windows Registry modifications.
- **Instant alerts:** Sends email notifications to the administrator when unauthorized access is detected.
- **Activity logging:** Keeps detailed logs in a clear HTML format for audits and evidence.
- **User-friendly GUI:** Built with Python's Tkinter for easy interaction and management.
- **Cost-effective and platform-compatible:** Uses open-source tools and works on standard Windows machines.

Benefits:
The proposed system provides **physical-level security**, reducing the risk of unauthorized access, data theft, and malware introduction. It is suitable for:

- Personal desktops and laptops.

- School or college computer labs.
- Corporate offices with sensitive data.

4. System Design

4.1 Feasibility Study

4.1.1 Economical Feasibility

The system uses open-source technologies like Python, Tkinter for GUI, and OpenCV for webcam monitoring, which eliminates licensing costs. The only hardware required is a webcam, which is cost-effective and often already available in most PCs. This makes the project feasible for schools, offices, and individuals with minimal investment.

4.1.2 Technical Feasibility

The system can be easily implemented on Windows platforms using Python scripts and Windows Registry modifications. No specialized hardware is needed beyond a webcam. The required skills (Python programming, Tkinter, basic registry handling) are well within reach of IT staff or advanced students.

4.1.3 Social Feasibility

The system promotes better security practices by raising awareness about the risks of unauthorized USB usage. It is user-friendly and requires minimal training for administrators. It does not disrupt regular workflows and provides peace of mind by adding an additional layer of physical security.

4.2 Input and Output Design

4.2.1 Input Design

- **Admin Credentials:** Used to authenticate and authorize USB access control.
- **User Access Requests:** Users must request access to USB ports through the GUI.
- **Webcam Feed:** Monitored continuously to detect physical access attempts.

4.2.2 Objectives

- Prevent unauthorized data transfer via USB ports.
- Detect and record any unauthorized attempts.
- Alert the admin immediately upon suspicious activity.
- Maintain detailed activity logs for auditing.

4.2.3 Output Design

- **Alerts:** Real-time email notifications to the administrator.
- **Captured Evidence:** Images or videos saved when an unauthorized attempt is detected.
- **Logs:** Structured HTML log files recording user activities, access attempts, and alerts.
- **Status Reports:** Summaries for system health and security status.

---

5. Implementation

5.1 Module Description

5.1.1 Admin Authentication Module

- Provides secure login for the administrator.
- Verifies username and password before granting access to system controls.

5.1.2 USB Port Control Module

- Controls the enabling and disabling of USB ports by modifying Windows Registry keys.
- Ensures that only authenticated users can enable USB ports.

5.1.3 Intruder Detection Module

- Continuously monitors the webcam feed using OpenCV.
- Captures and saves images or video if unauthorized access is detected.
- Triggers alerts when physical access is attempted without permission.

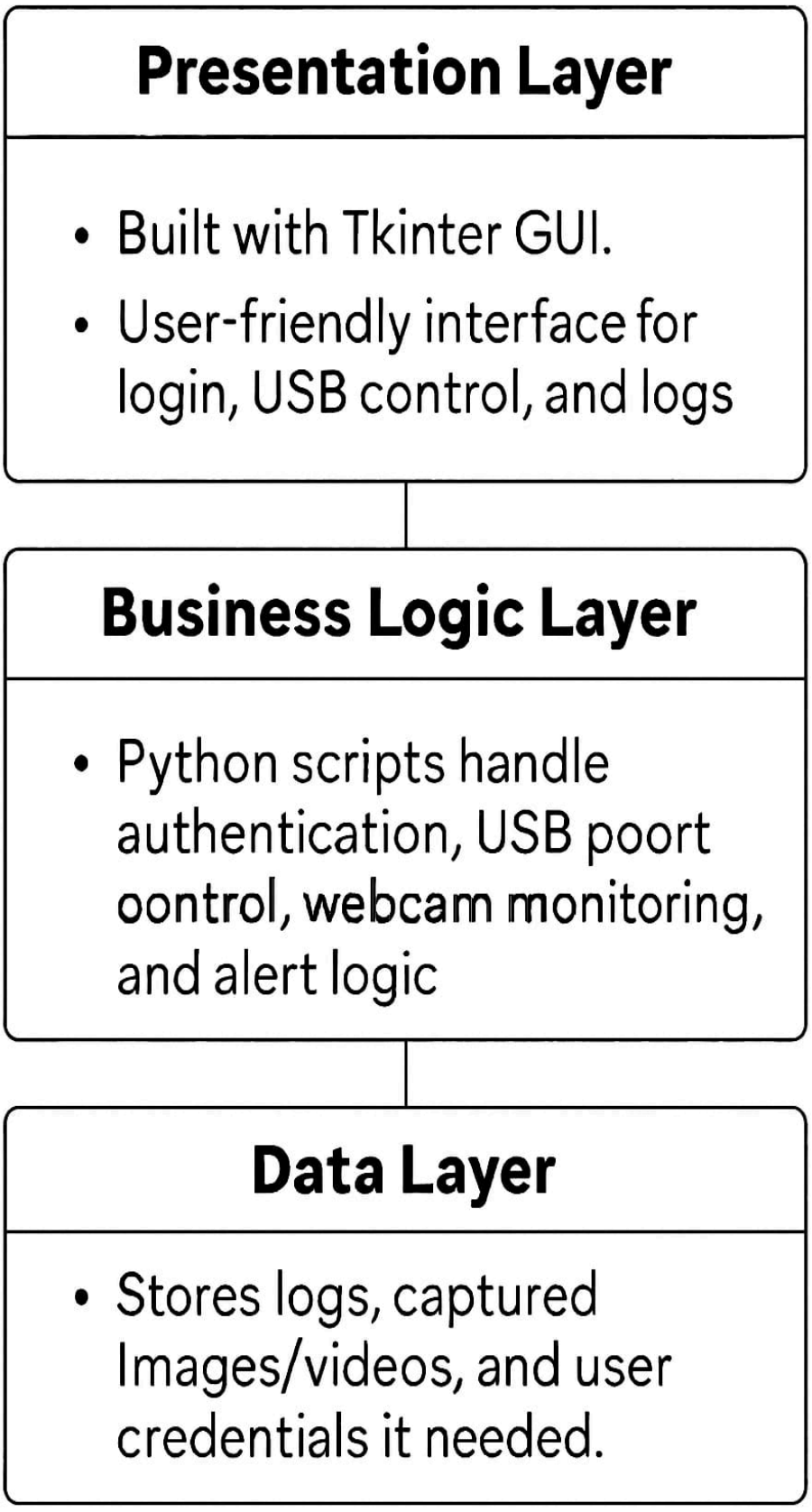
5.1.4 Alert and Reporting Module

- Sends real-time email alerts to the administrator.
- Logs all activities (access attempts, admin actions) in an HTML format for easy review.
- Stores evidence securely.

5.2 System Architecture

Layers:

1. **Presentation Layer**
    - Built with Tkinter GUI.
    - User-friendly interface for login, USB control, and logs.
2. **Business Logic Layer**
    - Python scripts handle authentication, USB port control, webcam monitoring, and alert logic.
3. **Data Layer**
    - Stores logs, captured images/videos, and user credentials if needed.


Presentation Layer
• Built with Tkinter GUI.
• User-friendly interface for login, USB control, and logs
Business Logic Layer
• Python scripts handle authentication, USB poort oontrol, webcam monitoring, and alert logic
Data Layer
• Stores logs, captured Images/videos, and user credentials it needed.

---

6. Algorithm Implementation

6.1 Detection and Control Logic

Unlike spam detection, this project does **not** use Naive Bayes. Instead, it uses simple but effective algorithms:

- **Access Control Logic:**
    - If admin credentials are valid, allow registry changes to enable USB ports.
    - Else, deny access and log the attempt.
- **Registry Modification Logic:**
    - Use `regedit` or Python's `winreg` module to modify USB port settings in the Windows Registry.
- **Intruder Detection Logic:**
    - Continuously read frames from the webcam.
    - If motion is detected when ports are disabled, capture the frame and save evidence.
- **Alert Logic:**
    - Send an email with captured evidence using SMTP.
    - Append alert details to the HTML log file.

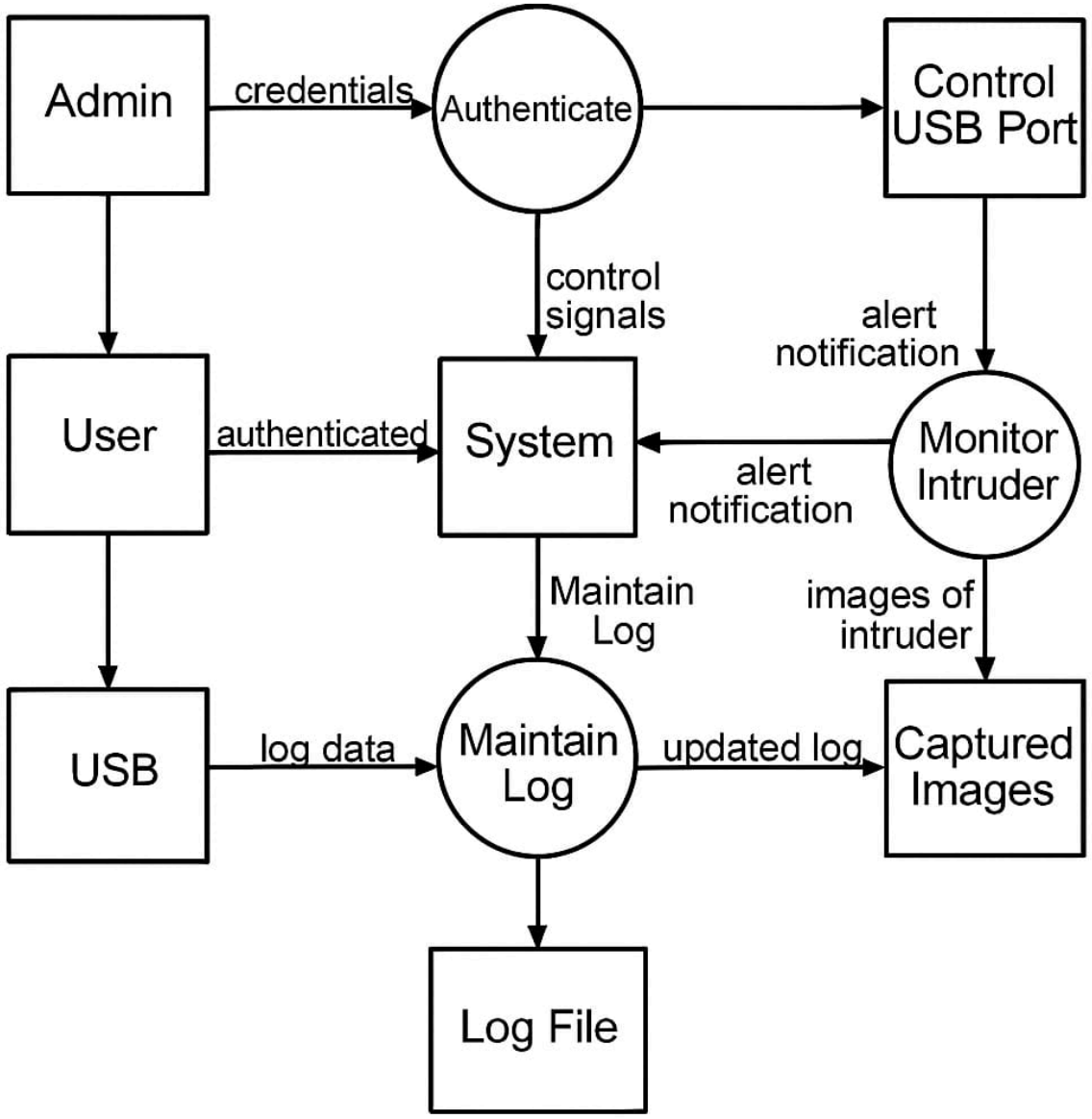
---

7. System Design Diagrams

7.1 Data Flow Diagram (DFD)

Levels:

- **Admin/User** → provides input (credentials, requests)
- **Control Logic** → verifies, modifies registry, triggers monitoring
- **Intruder Detection** → webcam, capture, store
- **Alert System** → send email, update logs



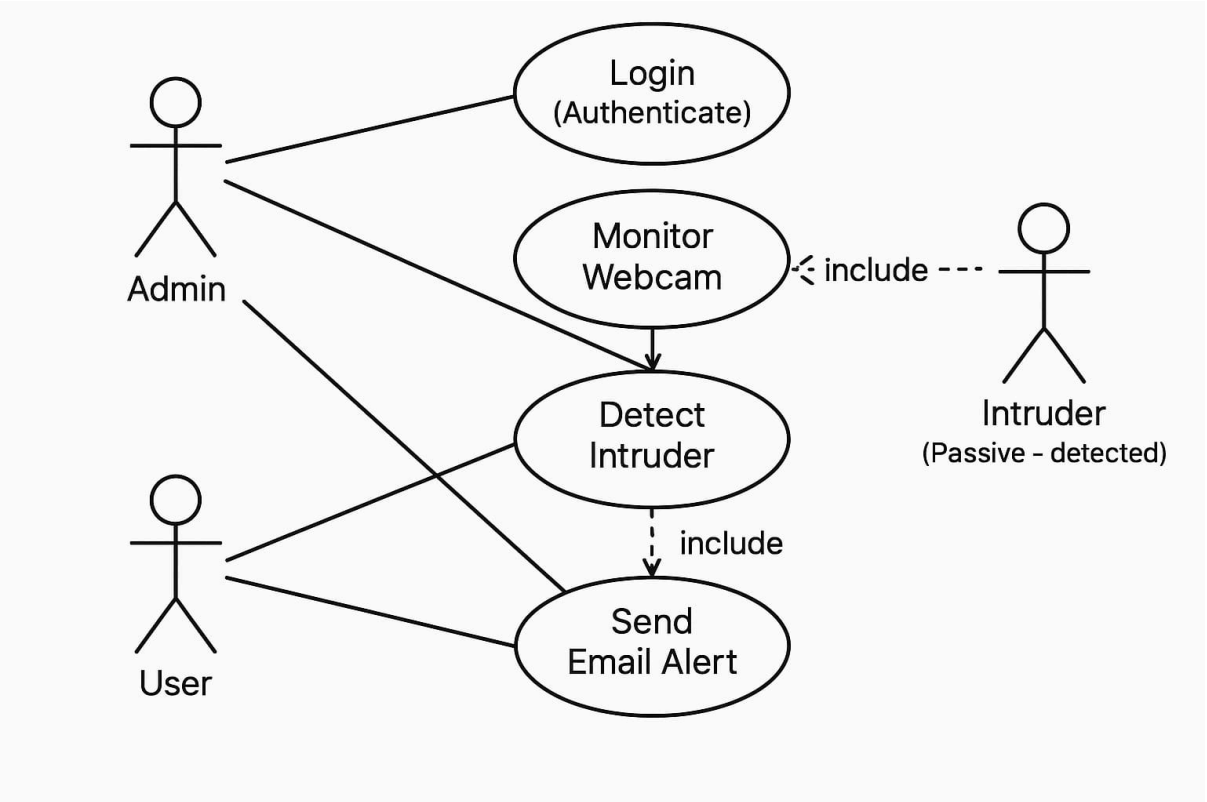
7.2 Use Case Diagram

Actors:

- Admin
- Authorized User
- Intruder

Use Cases:

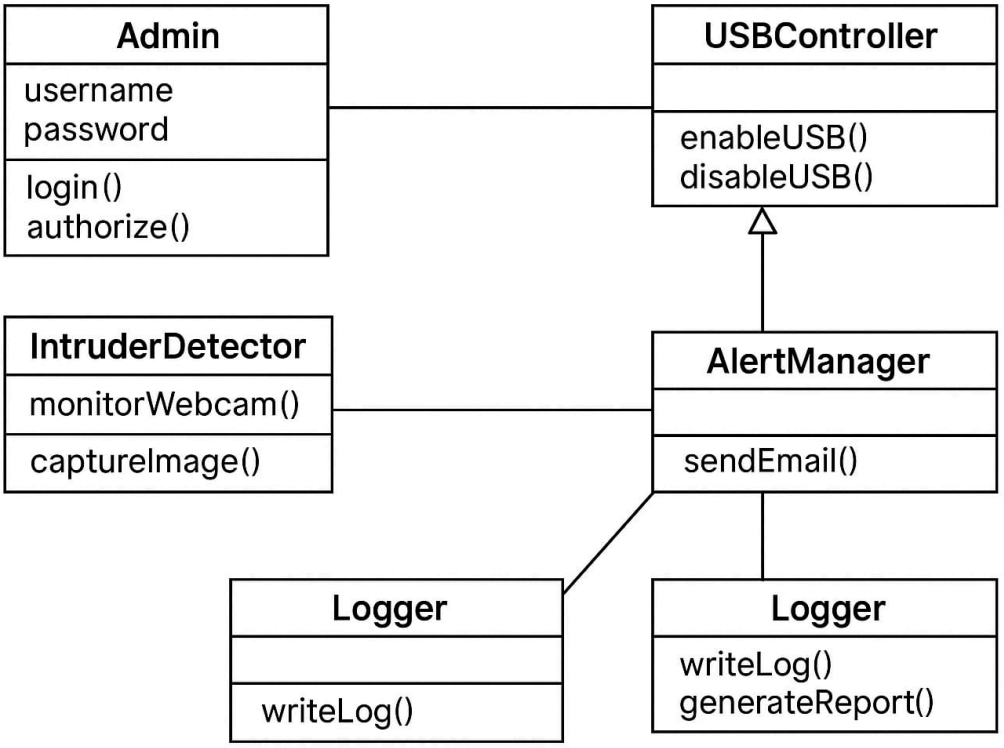
- Authenticate
- Enable/Disable USB ports
- Monitor Webcam
- Detect Intruder
- Send Alert
- View Logs



7.3 Class Diagram

Main Classes:

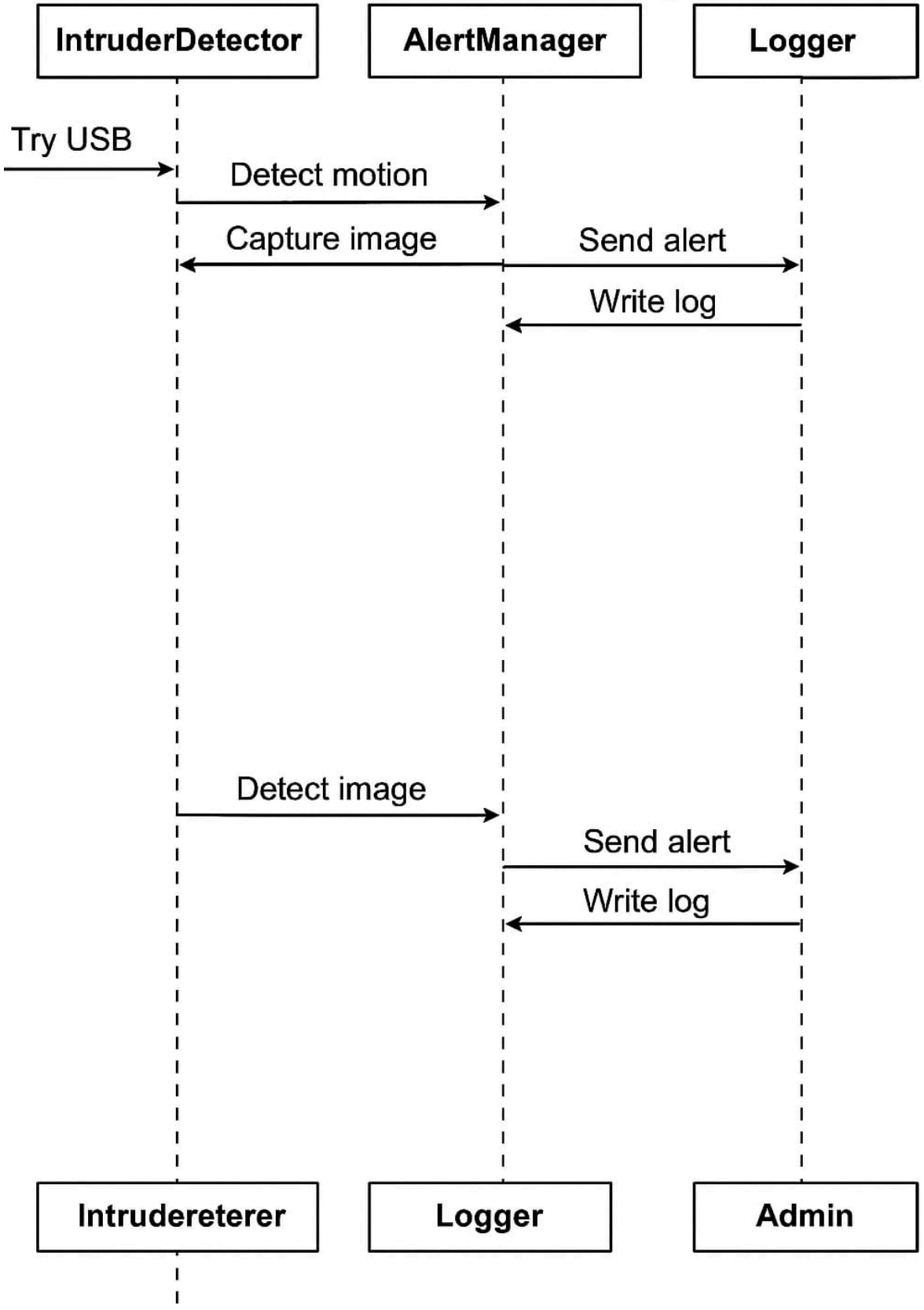
- **Admin** — Attributes: username, password.
- **USBController** — Methods: enable_ports(), disable_ports().
- **IntruderDetector** — Methods: monitor_webcam(), capture_image().
- **AlertManager** — Methods: send_email(), update_log().
- **Logger** — Methods: write_html_log().



7.4 Sequence Diagram

Scenario: Unauthorized Access Attempt

1. **Intruder:** Tries to plug in USB.
2. **IntruderDetector:** Detects motion.
3. **IntruderDetector:** Captures image.
4. **AlertManager:** Sends email to admin.
5. **Logger:** Updates HTML log with timestamp and evidence.


IntruderDetector
AlertManager
Logger
Try USB
Detect motion
Capture image
Send alert
Write log
Detect image
Send alert
Write log
Intrudereterer
Logger
Admin

7.5 Activity Diagram

Activity diagrams describe how a system behaves in terms of the flow of control or data from one activity to another. They focus on the dynamic aspects of the system and are particularly useful for modeling the workflow of business processes or system operations. These diagrams are important for understanding how operations are coordinated.

Purpose of Activity Diagram:

- Describe how activities are coordinated to provide a service.
- Represent the sequence of events needed to achieve an operation.
- Show how activities in a use case relate to one another.
- Model workflows between and within use cases.

Key Elements:

- **Activities**: Units of work within the system that require time and must be completed before progressing.
- **Actions**: Individual steps within an activity.
- **Transitions**: Movement from one activity to another.
- **Control Nodes**:
    - **Initial Node**: Start of the workflow.
    - **Final Node**: End of the workflow.
    - **Forks and Joins**: Used to split and synchronize flows.
    - **Decision and Merge Points**: Represent branching and merging of paths.
- **Tokens**: Represent control or data flowing along the activity edges.
- **States**: Represent waiting for some condition or event.

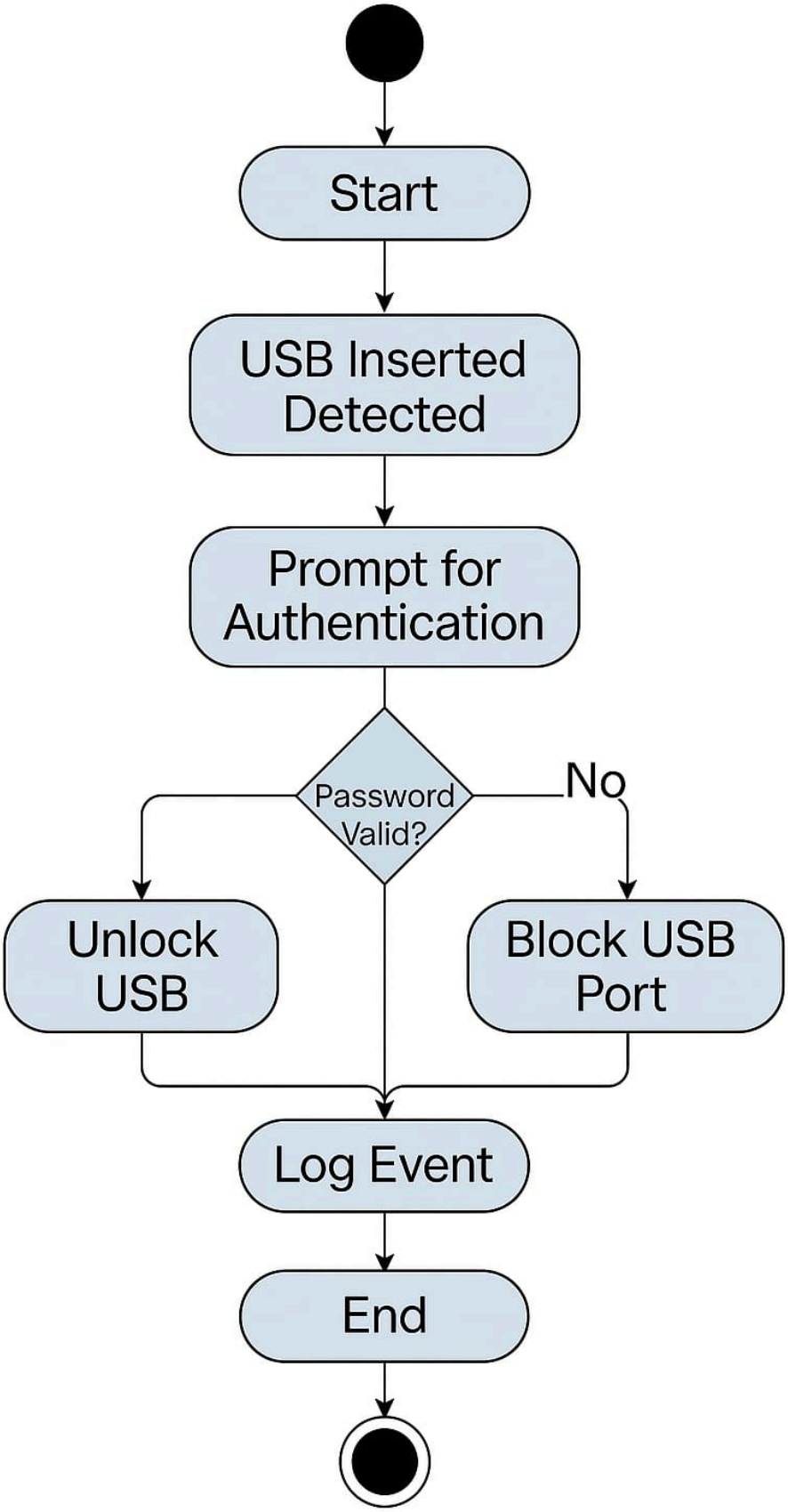
7.5.1 User Activity Flow:

- Login
- Compose Mail
- View Inbox
- View Recent Mails
- View Spam
- Logout

7.5.2 Admin Activity Flow:

- Login
- Add/Update Keywords
- View All Users' Messages
- Manage Spam Rules

- Logout


Start
USB Inserted Detected
Prompt for Authentication
Password Valid?
No
Unlock USB
Block USB Port
Log Event
End

7.6 Component Diagram

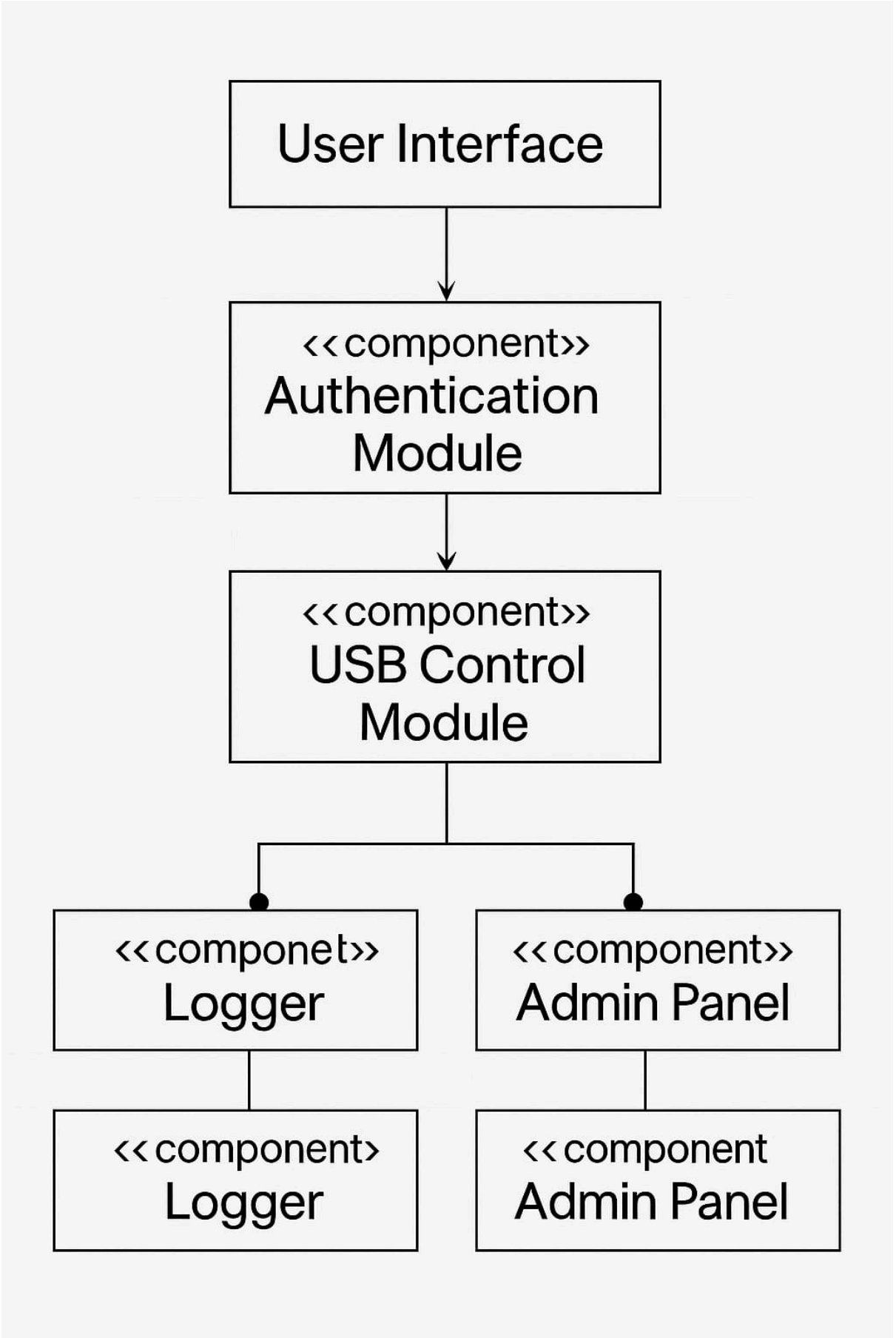
Component diagrams depict how software system components interact and depend on each other. They are particularly useful for visualizing the physical components (modules, packages, etc.) of the system and their interfaces.

7.6.1 User Components:

- UI Component
- Compose Mail Module
- Inbox Module
- Spam Detection Client

7.6.2 Admin Components:

- Admin Panel UI
- Keyword Management Module
- Spam Filtering Engine
- User Data Viewer


User Interface
<<component>>
Authentication Module
<<component>>
USB Control Module
<<componet>>
Logger
<<component>>
Admin Panel
<<component>
Logger
<<component
Admin Panel

7.7 ER Diagram

Entity-Relationship diagrams represent the data structure and relationships in the system. They show how data is stored and how entities (like users, mails, keywords) relate to each other.

7.7.1 User ER Entities:

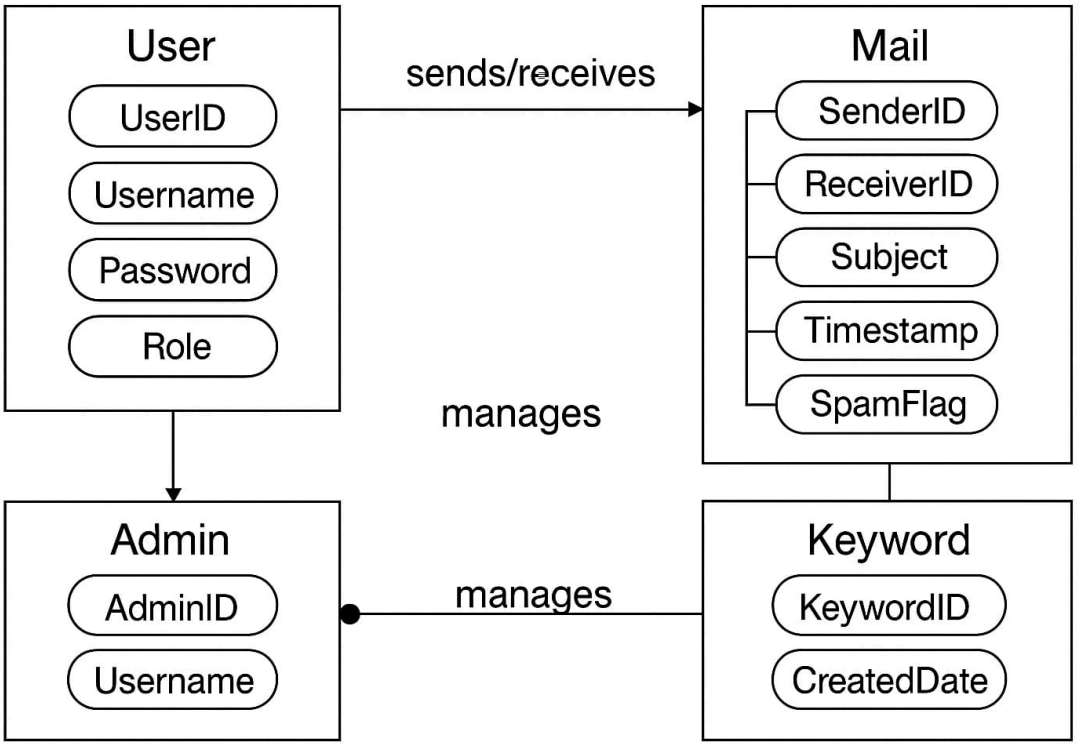
- **User**(UserID, Username, Password, Role)
- **Mail**(MailID, SenderID, ReceiverID, Subject, Body, Timestamp, SpamFlag)

7.7.2 Admin ER Entities:

- **Admin**(AdminID, Username, Password)
- **Keyword**(KeywordID, Word, CreatedDate)

Relationships:

- User sends and receives multiple Mails
- Admin manages Keywords
- Mails are tagged as Spam based on Keywords



8. REQUIREMENT SPECIFICATION

8.1 Functional Requirements

- **R1**: user_accounts() – Allows user registration and authentication.
- **R2**: admin_accounts() – Allows admin authentication and management.
- **R3**: data_handling() – Defines structure and validation of input data.
- **R4**: input_design() – Converts user-friendly input to structured system data.
- **R5**: output_design() – Ensures correct and clear output.
- **R6**: compose_mail() – Users send messages.
- **R7**: view_recent_mail() – Users view latest messages.
- **R8**: view_recent_histories() – Users view message history.
- **R9**: inbox_messages() – Mails stored if no threats found.

- **R10**: spam_messages() – Detected threat mails stored in spam.
- **R11**: add_keywords() – Admin adds keywords for filtering.

8.2 Software Requirements

- Python
- Django

8.3 Operating Systems Supported

- Windows XP
- Windows 7
- Windows 8
- Windows 10

8.4 Technologies and Languages Used

- Front-End: HTML, CSS, JavaScript
- Design: HTML/CSS
- Back-End: Python
- Database: MySQL

8.5 Hardware Requirements

- System: Pentium IV 2.4 GHz
- Hard Disk: 40 GB
- RAM: 2 GB
- Floppy Drive: 1.44 Mb

- Monitor: 14’ Color Monitor
- Mouse: Optical Mouse

9. System Test :

9.1 Types of Tests

9.1.1 Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results**.**

9.1.2 Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

9.1.3 Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined

9.1.4 System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

9.1.5 White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

9.1.6 Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works. 9.2 Test strategy and approach: Field testing will be performed manually and functional tests will be written in detail.

9.2.1 Test objectives:

All field entries must work properly. Pages must be activated from the identified link. The entry screen, messages and responses must not be delayed.

9.2.2 Features to be tested:

Verify that the entries are of the correct format No duplicate entries should be allowed All links should take the user to the correct page.

9.3 Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by

interface defects.The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

9.3.1 Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

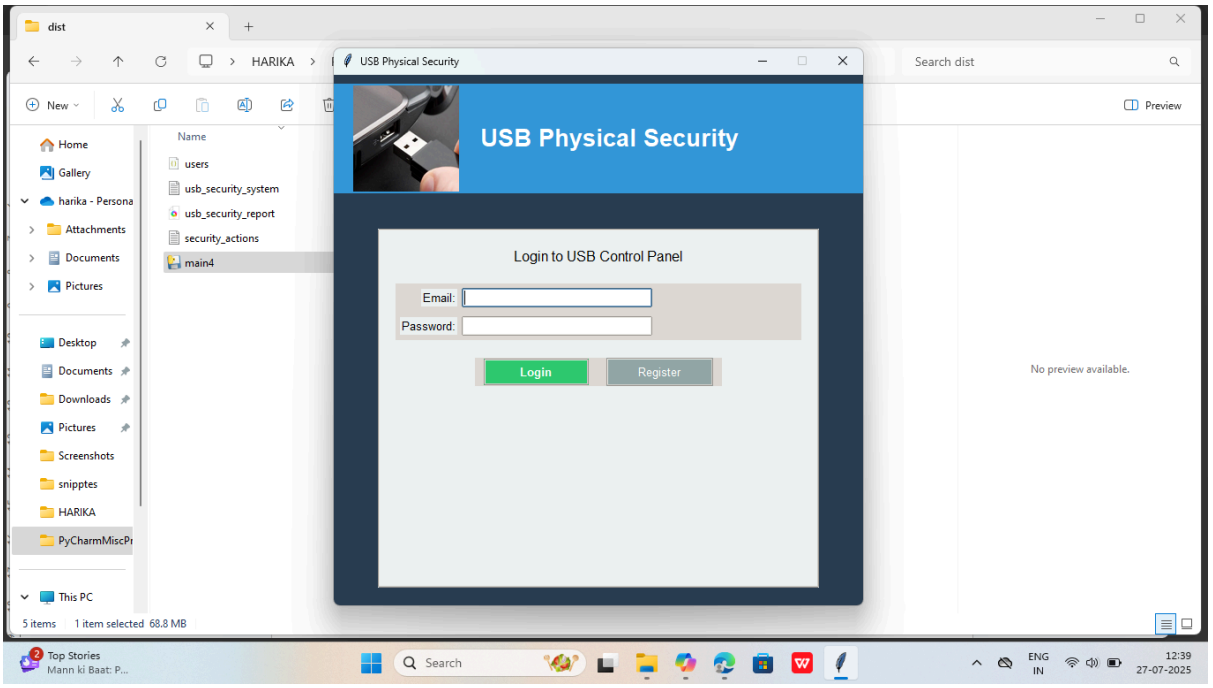
9.4 Acceptance Testing:

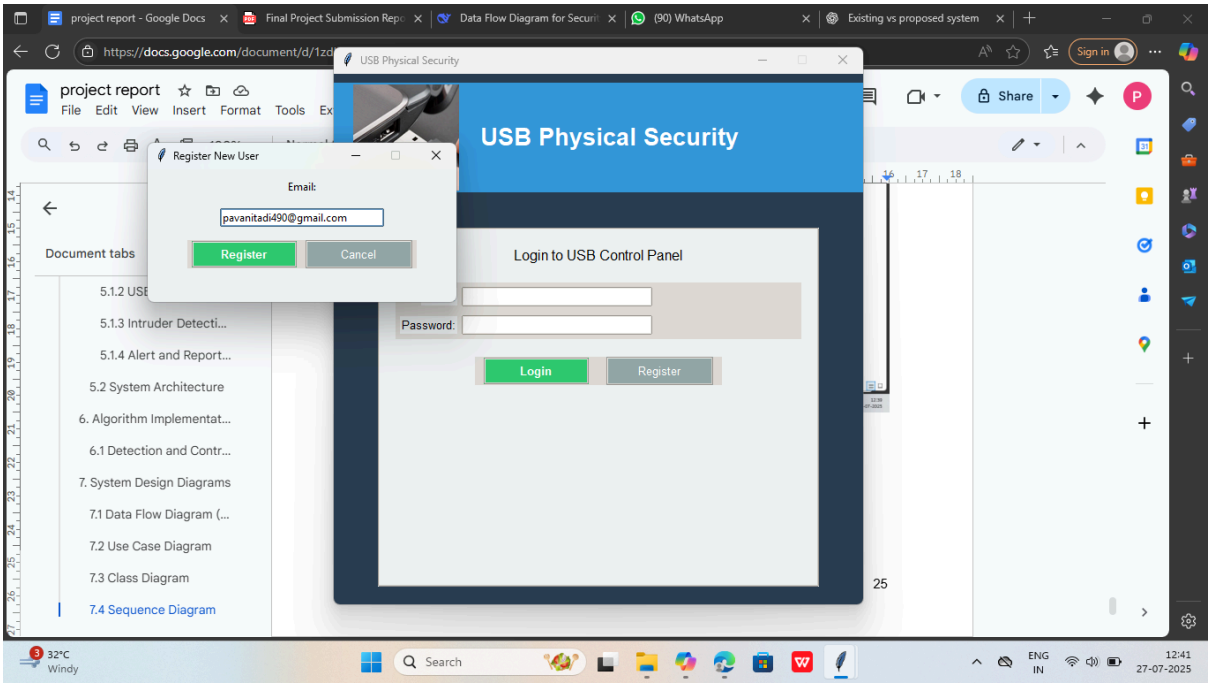
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

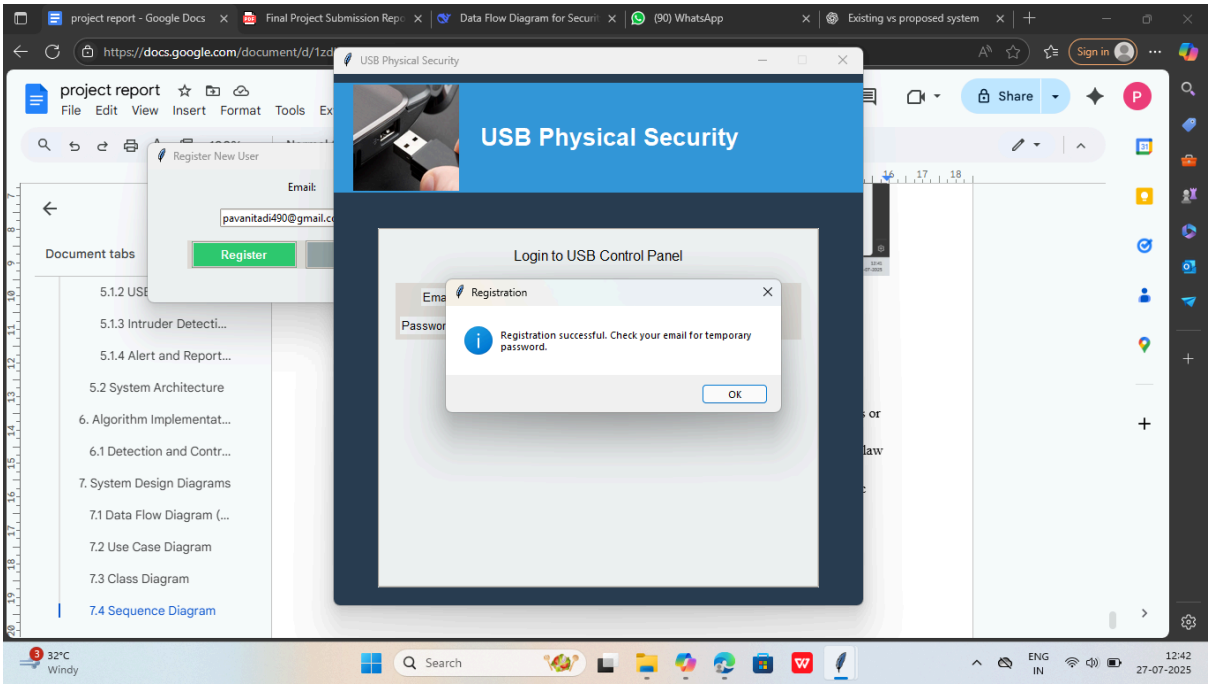
9.4.1 Test Results:

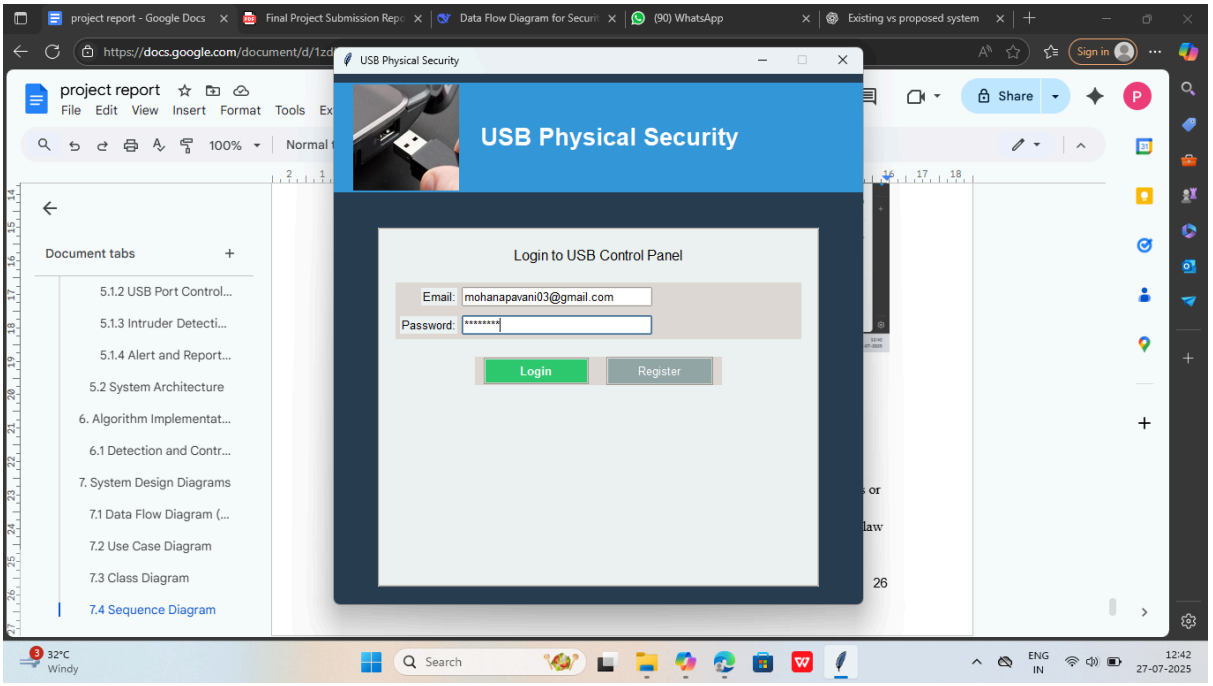
All the test cases mentioned above passed successfully. No defects encountered.

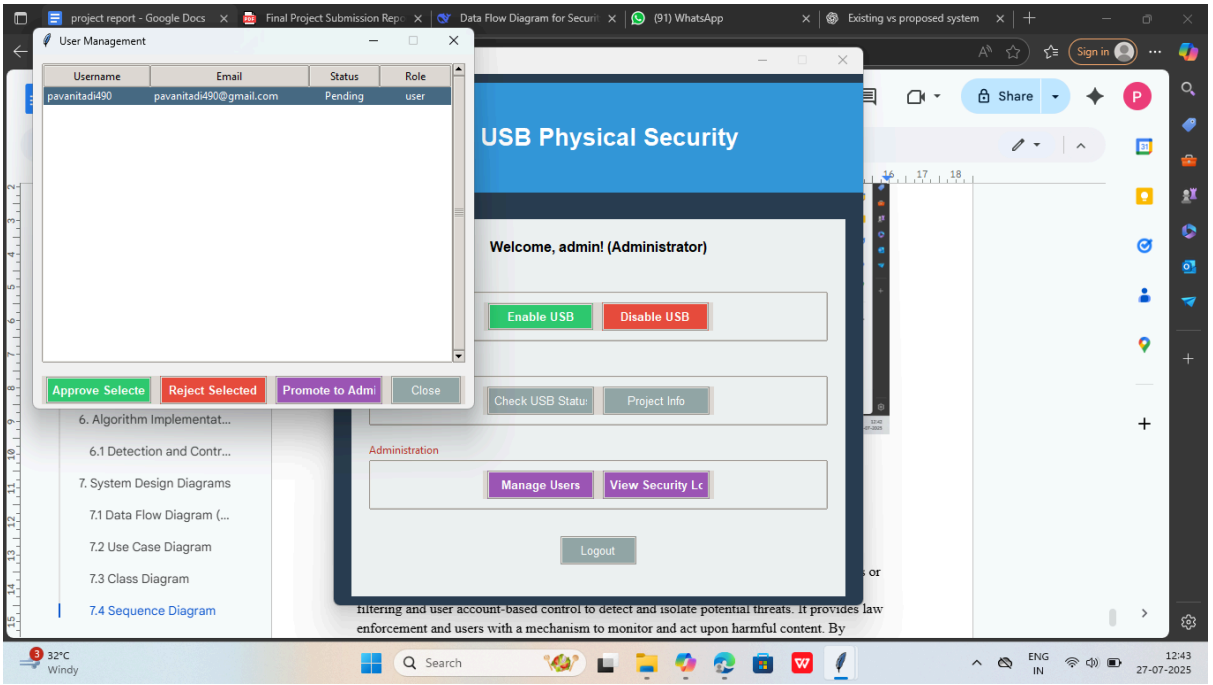
Execution:

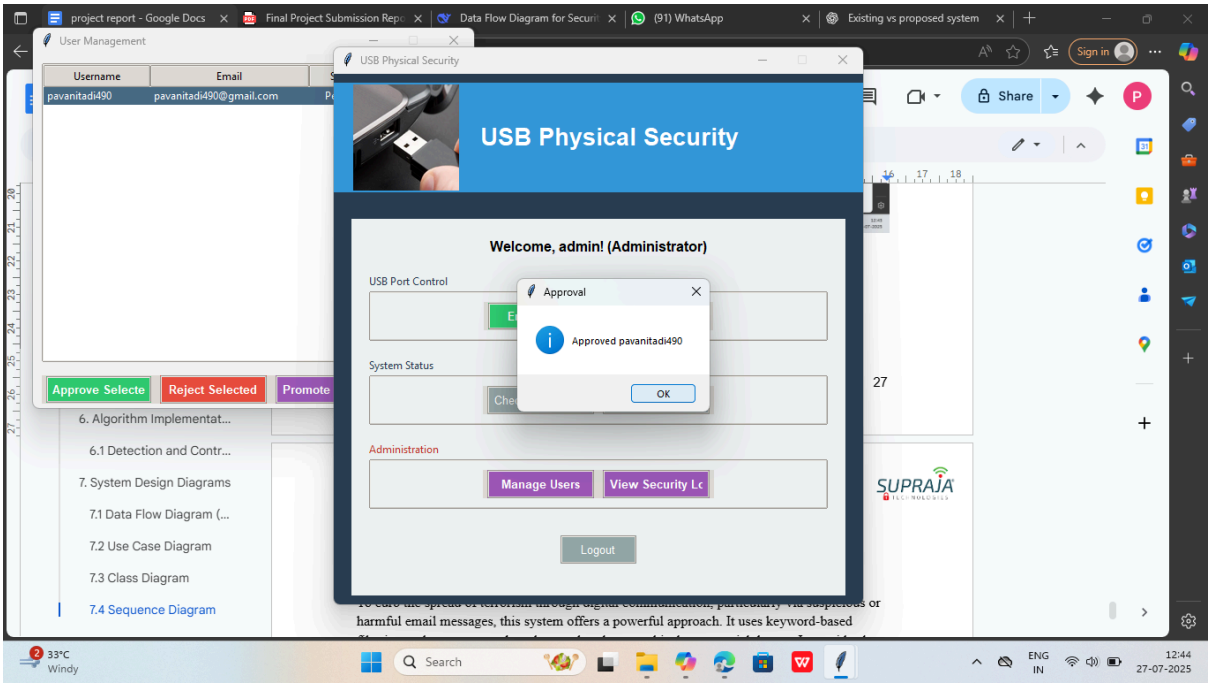


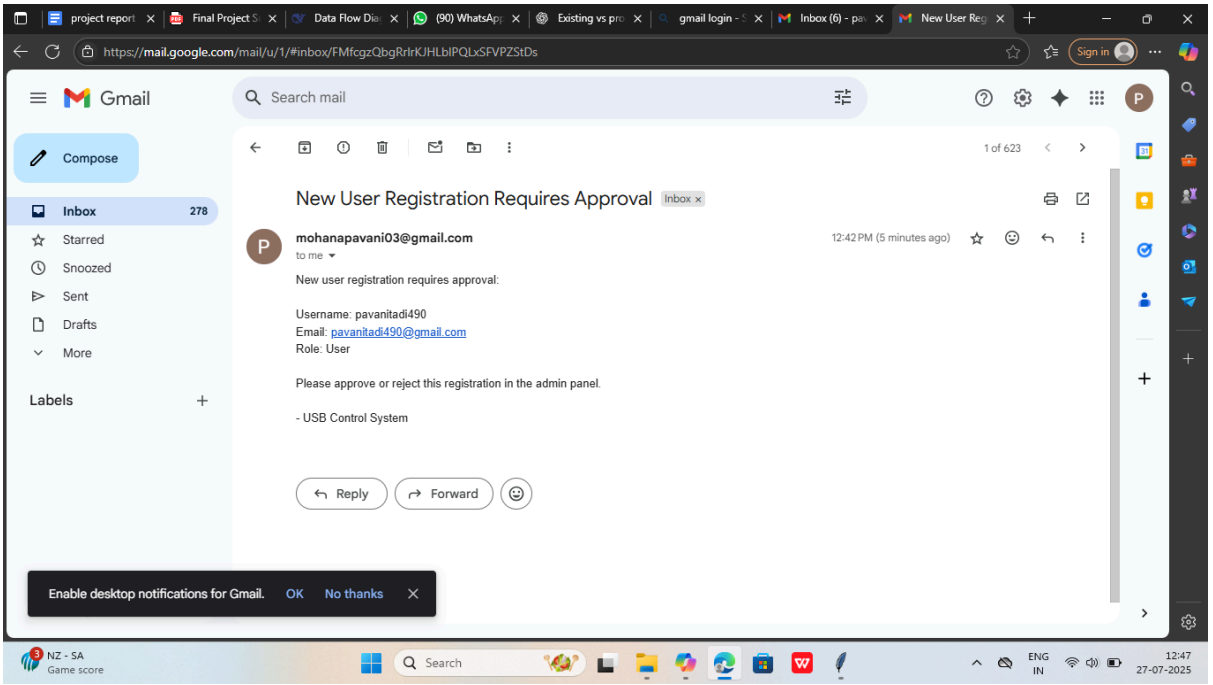

project report - Google Docs
Final Project Submission Repo
Data Flow Diagram for Securi
(90) WhatsApp
Existing vs proposed system
https://docs.google.com/document/d/1zd
Sign in
project report
File Edit View Insert Format Tools Ex
Share
Register New User
Email:
pavanitadi490@gmail.com
Register
Cancel
USB Physical Security
USB Physical Security
Login to USB Control Panel
Password:
Login
Register
Document tabs
5.1.2 USE
5.1.3 Intruder Detecti...
5.1.4 Alert and Report...
5.2 System Architecture
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram
25
32°C
Windy
Search
ENG
IN
12:41
27-07-2025

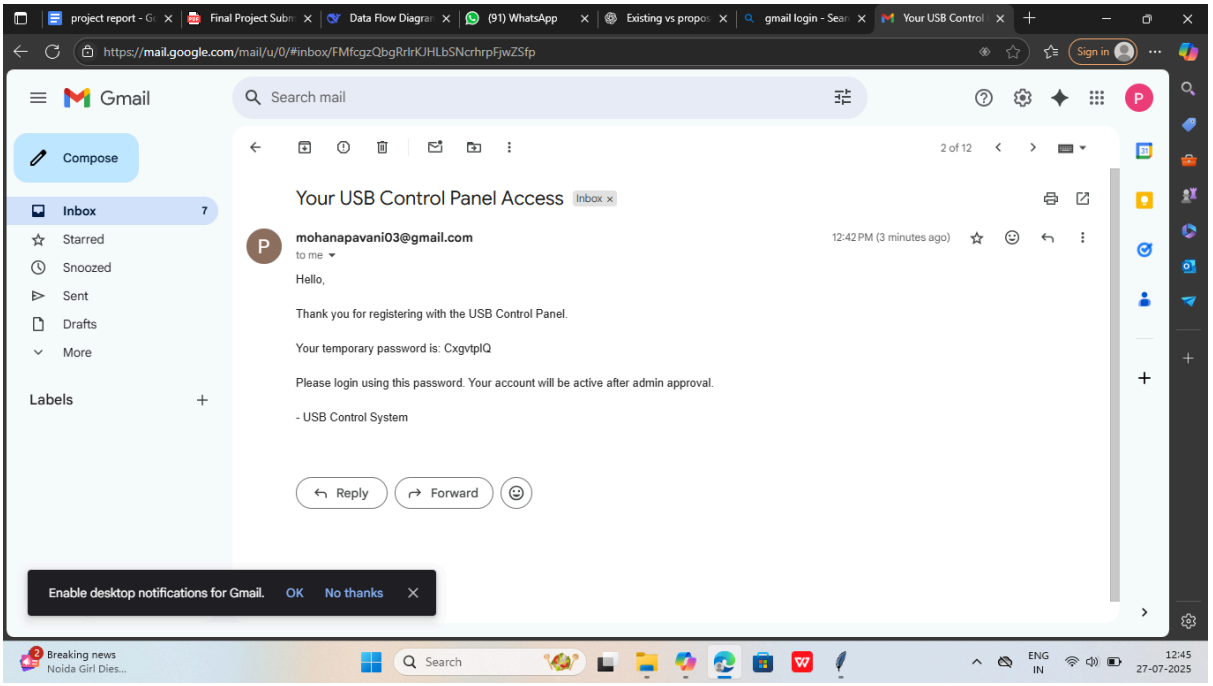

project report - Google Docs
Final Project Submission Repo
Data Flow Diagram for Securi
(90) WhatsApp
Existing vs proposed system
https://docs.google.com/document/d/1zd
Sign in
project report
File Edit View Insert Format Tools Ex
Share
Register New User
Email:
Register
USB Physical Security
USB Physical Security
Login to USB Control Panel
Registration
Registration successful. Check your email for temporary password.
OK
Document tabs
5.1.2 USE
5.1.3 Intruder Detecti...
5.1.4 Alert and Report...
5.2 System Architecture
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram
32°C
Windy
Search
ENG
IN
12:42
27-07-2025

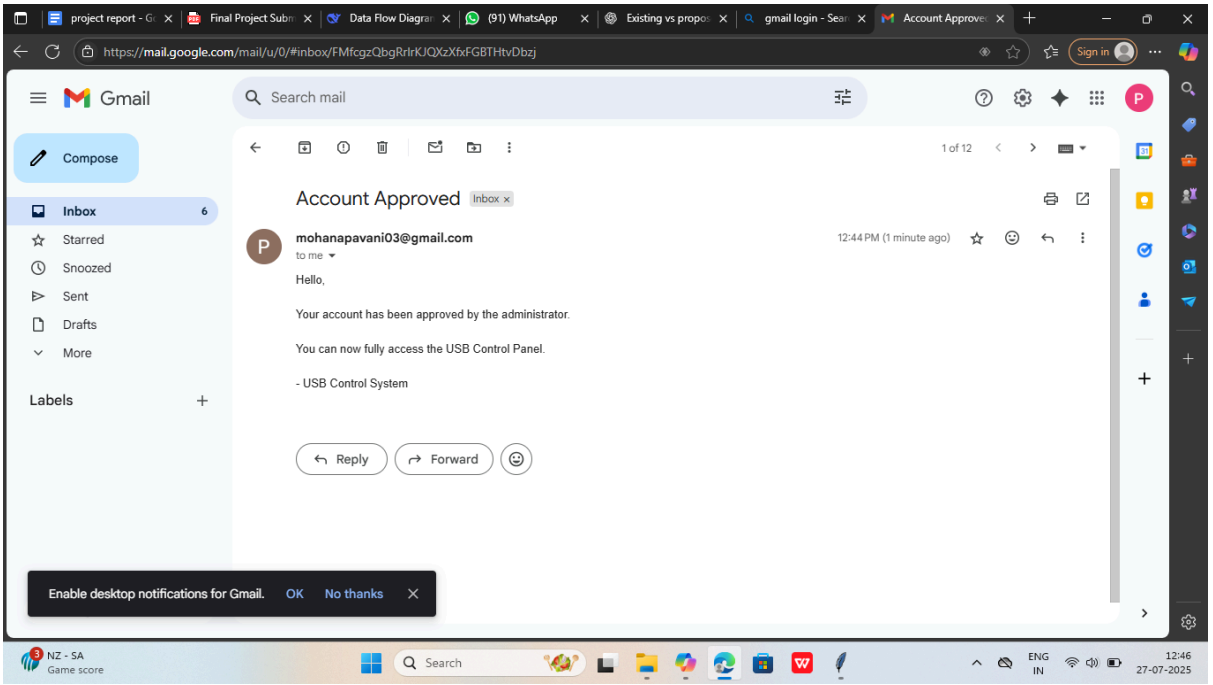

USB Physical Security
USB Physical Security
Login to USB Control Panel
Email: mohanapavani03@gmail.com
Password: ********
Login
Register
project report
Document tabs
5.1.2 USB Port Control...
5.1.3 Intruder Detecti...
5.1.4 Alert and Report...
5.2 System Architecture
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram

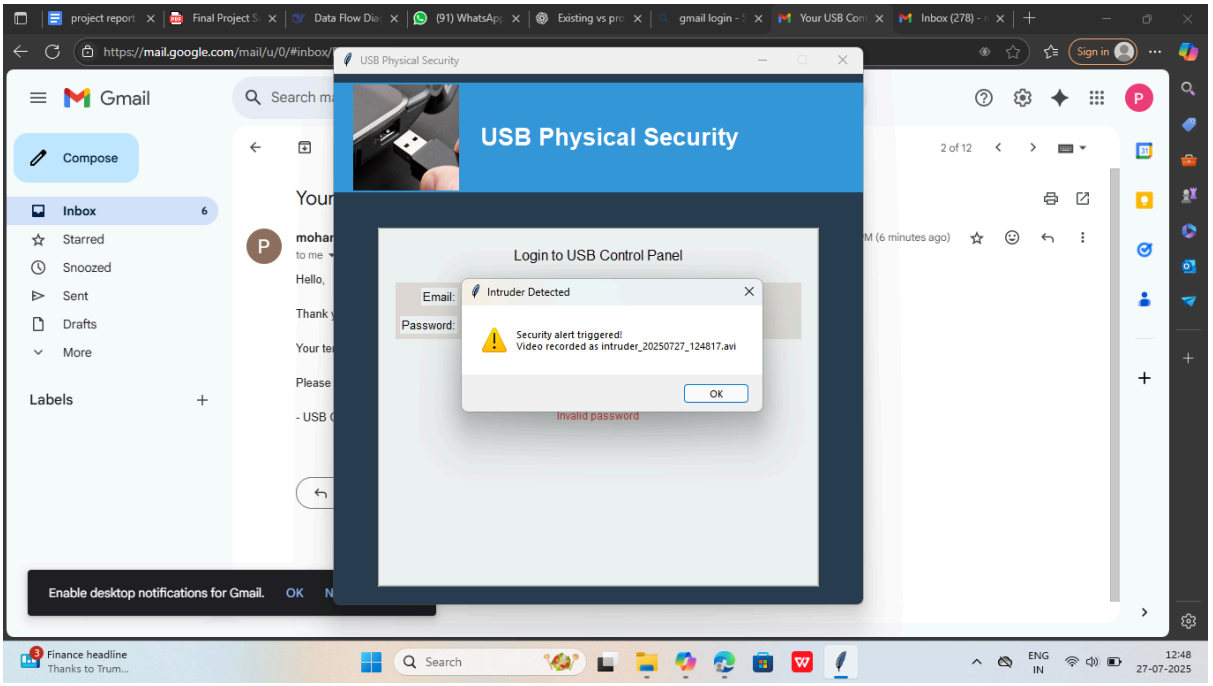

User Management
Username
Email
Status
Role
pavanitadi490
pavanitadi490@gmail.com
Pending
user
Approve Selecte
Reject Selected
Promote to Admi
Close
USB Physical Security
Welcome, admin! (Administrator)
Enable USB
Disable USB
Check USB Statu
Project Info
Administration
Manage Users
View Security Lc
Logout
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram


User Management
Username
Email
pavanitadi490
pavanitadi490@gmail.com
USB Physical Security
USB Physical Security
Welcome, admin! (Administrator)
USB Port Control
Approval
Approved pavanitadi490
OK
System Status
Administration
Manage Users
View Security Lc
Logout
Approve Selecte
Reject Selected
Promote
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram
harmful email messages, this system offers a powerful approach. It uses keyword-based
Share
Sign in
SUPRAJA
33°C
Windy
Search
ENG
IN
12:44
27-07-2025

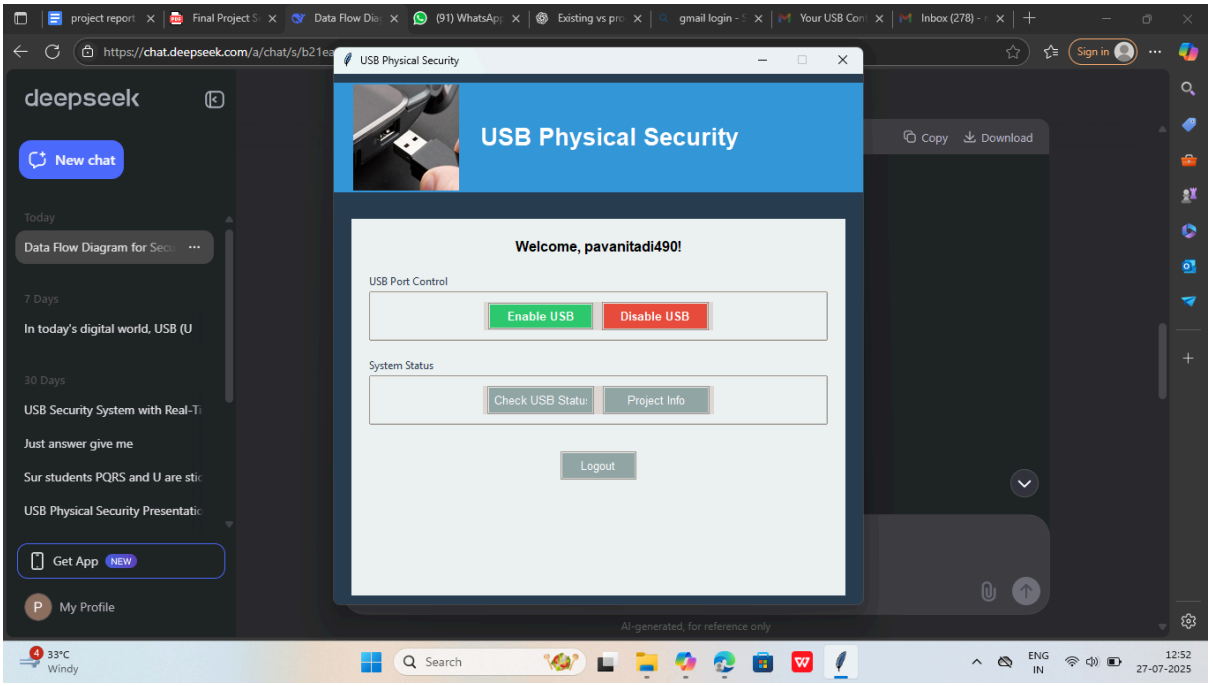

https://mail.google.com/mail/u/1/#inbox/FMfcgzQbgRrlrKJHLblPQLxSFVPZStDs
Gmail
Search mail
Compose
Inbox 278
Starred
Snoozed
Sent
Drafts
More
Labels
1 of 623
New User Registration Requires Approval
Inbox ×
mohanapavani03@gmail.com
to me
12:42 PM (5 minutes ago)
New user registration requires approval:
Username: pavanitadi490
Email: pavanitadi490@gmail.com
Role: User
Please approve or reject this registration in the admin panel.
- USB Control System
Reply
Forward
Enable desktop notifications for Gmail.
OK
No thanks
NZ - SA
Game score
Search
ENG
IN
12:47
27-07-2025

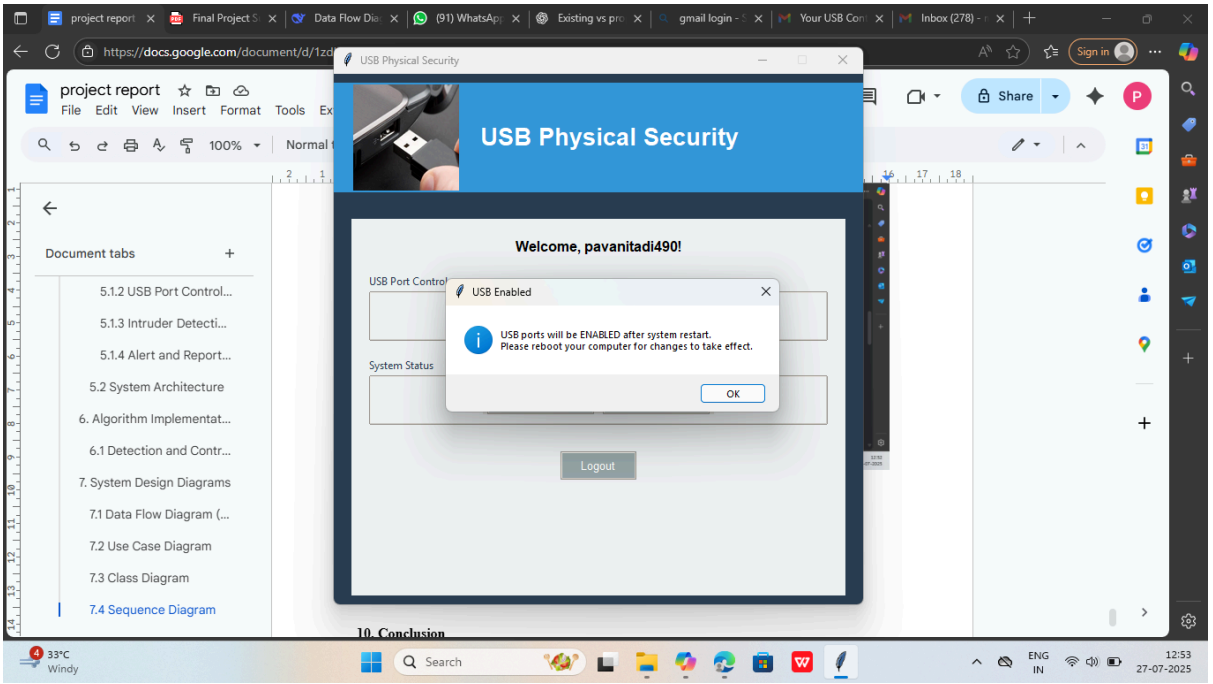

Gmail
Search mail
Compose
Inbox 7
Starred
Snoozed
Sent
Drafts
More
Labels
2 of 12
Your USB Control Panel Access
Inbox
mohanapavani03@gmail.com
to me
12:42 PM (3 minutes ago)
Hello,
Thank you for registering with the USB Control Panel.
Your temporary password is: CxgvtpIQ
Please login using this password. Your account will be active after admin approval.
- USB Control System
Reply
Forward
Enable desktop notifications for Gmail.
OK
No thanks
Search
ENG
IN
12:45
27-07-2025

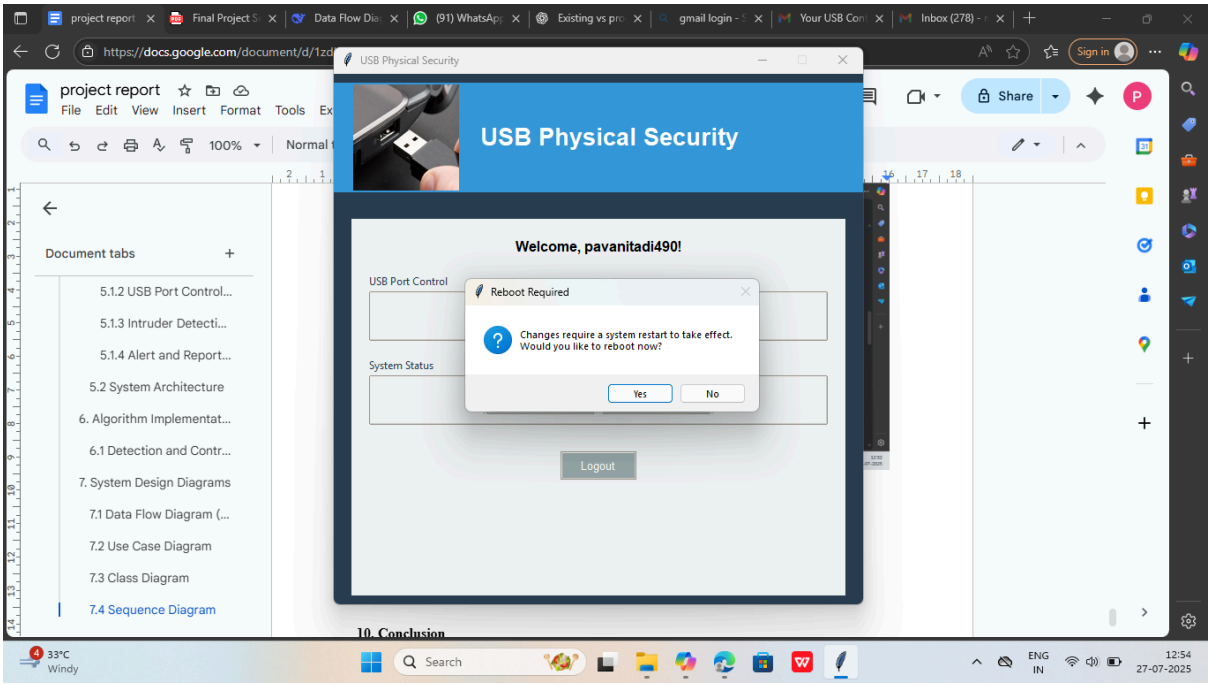

Gmail
Search mail
Compose
Inbox 6
Starred
Snoozed
Sent
Drafts
More
Labels
1 of 12
Account Approved
Inbox
mohanapavani03@gmail.com
to me
12:44 PM (1 minute ago)
Hello,
Your account has been approved by the administrator.
You can now fully access the USB Control Panel.
- USB Control System
Reply
Forward
Enable desktop notifications for Gmail.
OK
No thanks
Search
ENG
IN
12:46
27-07-2025

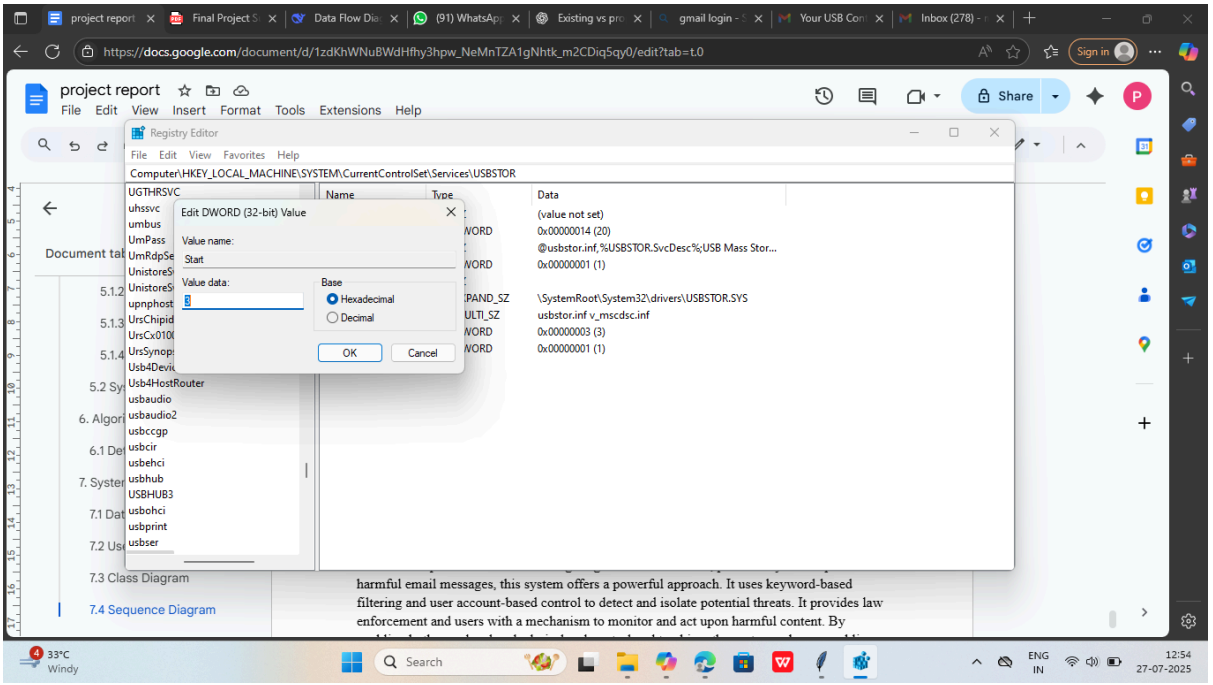


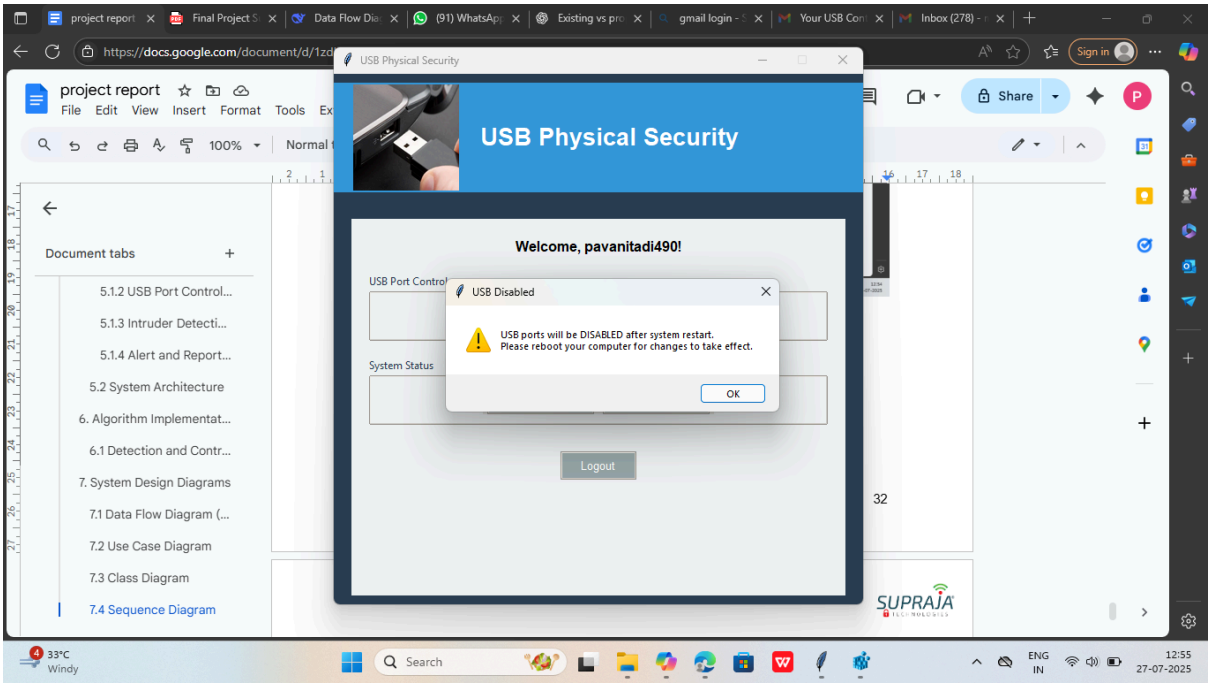
intruder_20250727_124817	27-07-2025 12:48	AVI File	903 KB

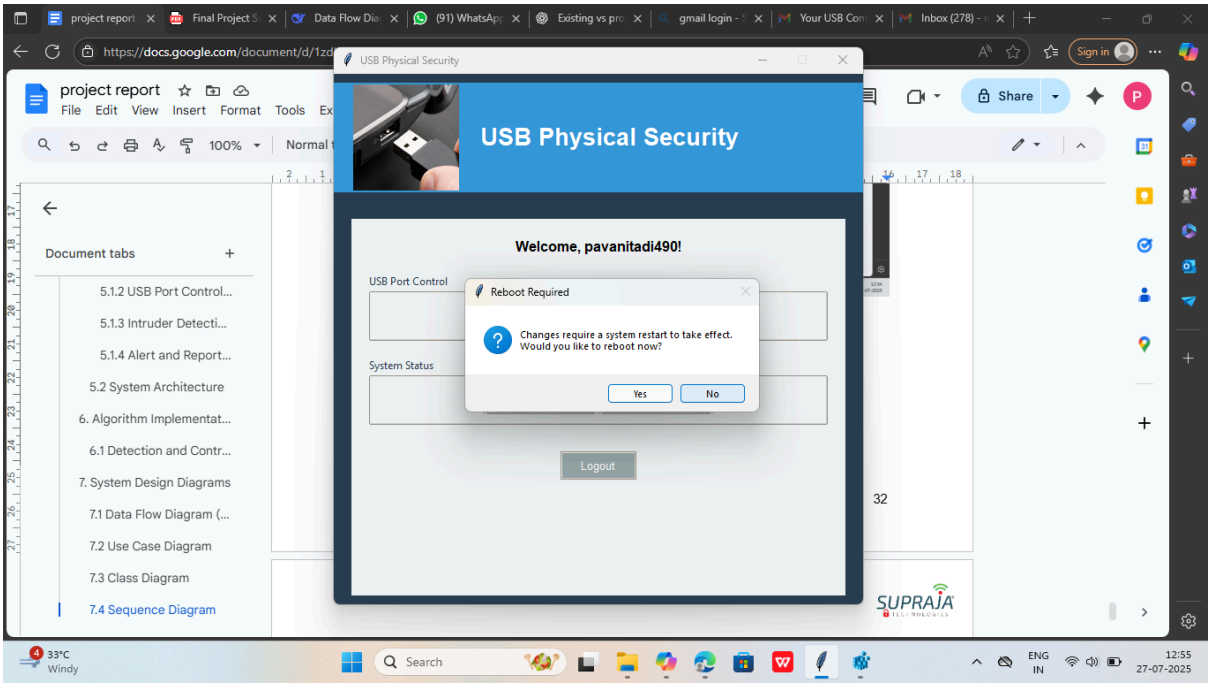

USB Physical Security
USB Physical Security
Welcome, pavanitadi490!
USB Port Control
Enable USB
Disable USB
System Status
Check USB Status
Project Info
Logout
deepseek
New chat
Get App
My Profile

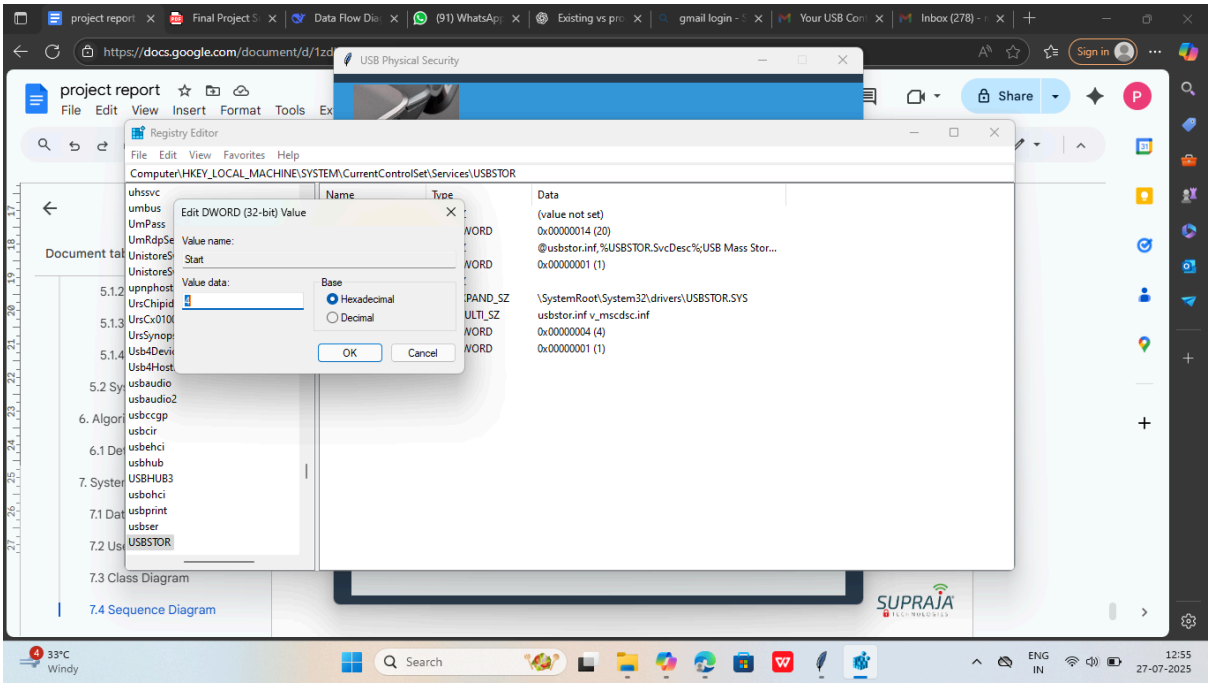

USB Physical Security
USB Physical Security
Welcome, pavanitadi490!
USB Enabled
USB ports will be ENABLED after system restart.
Please reboot your computer for changes to take effect.
OK
Logout
project report
Document tabs
10. Conclusion


USB Physical Security
USB Physical Security
Welcome, pavanitadi490!
USB Port Control
System Status
Logout
Reboot Required
Changes require a system restart to take effect. Would you like to reboot now?
Yes
No
project report
Document tabs
5.1.2 USB Port Control...
5.1.3 Intruder Detecti...
5.1.4 Alert and Report...
5.2 System Architecture
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram
10. Conclusion


project report
Registry Editor
Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USBSTOR
Edit DWORD (32-bit) Value
Value name:
Start
Value data:
3
Base
Hexadecimal
Decimal
OK
Cancel
7.3 Class Diagram
7.4 Sequence Diagram
harmful email messages, this system offers a powerful approach. It uses keyword-based filtering and user account-based control to detect and isolate potential threats. It provides law enforcement and users with a mechanism to monitor and act upon harmful content. By


USB Physical Security
USB Physical Security
Welcome, pavanitadi490!
USB Port Control
System Status
Logout
USB Disabled
USB ports will be DISABLED after system restart.
Please reboot your computer for changes to take effect.
OK
project report
Document tabs
5.1.2 USB Port Control...
5.1.3 Intruder Detecti...
5.1.4 Alert and Report...
5.2 System Architecture
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram


USB Physical Security
USB Physical Security
Welcome, pavanitadi490!
USB Port Control
System Status
Logout
Reboot Required
Changes require a system restart to take effect.
Would you like to reboot now?
Yes
No
project report
Document tabs
5.1.2 USB Port Control...
5.1.3 Intruder Detecti...
5.1.4 Alert and Report...
5.2 System Architecture
6. Algorithm Implementat...
6.1 Detection and Contr...
7. System Design Diagrams
7.1 Data Flow Diagram (...
7.2 Use Case Diagram
7.3 Class Diagram
7.4 Sequence Diagram



10. Conclusion

The USB Physical Security System developed in this project offers a reliable solution to one of the most overlooked yet critical areas of system security—unauthorized USB access. By implementing a robust authentication mechanism and port control logic, the system ensures that only authorized users can connect and use USB devices. This greatly reduces the risk of data theft, malware injection, and other security breaches that typically occur through external USB storage devices.

The project not only strengthens physical access control but also introduces system-level monitoring and administrative control capabilities. Features such as password protection, logging of events, and restricted access provide a strong foundation for enforcing data security policies, especially in organizational and institutional environments.

This system is cost-effective, easy to use, and can be further enhanced with biometric authentication, remote access alerts, and integration with antivirus/firewall systems. Overall, the project successfully demonstrates the importance and effectiveness of implementing USB-level security as part of a comprehensive cybersecurity strategy.