PROJECT REPORT ON

USB PHYSICAL SECURITY

Submitted by

Satvik Shrivastava

23EO5-ST#IS#6653

Under the Supervision of

UPENDRA

Senior Security Analyst

KRISHNA

Junior Security Analyst

Registered And Head Office

D.NO: 11-9-18, 1st Floor,

Majjivari Street, Kothapeta,

Vijayawada - 520001.

+91 9550055338 / +91 7901336873

contact@suprajatechnologies.com

Table Of Contents

1. Introduction

1.1 Overall Description

USB devices are widely utilized for data transfer on different systems, but they can also pose a major security threat. Malicious individuals take advantage of the natural trust in USB devices to insert malware, steal information, or disrupt activities. Utilizing data mining methods on USB behavior patterns from various devices allows us to create a system that can detect possible risks and weaknesses. This system is designed to examine data transferred through USB devices and identify potentially malicious behavior that may suggest attempted exploitation.

Due to the increase in USB-related threats like BadUSB, organizations need to improve their security measures. USB drives can spread harmful code, unauthorized access, or corrupt data. Utilizing both data mining and behavioral analysis offers a more thorough evaluation of possible risks and identification of patterns suggesting harmful behavior. This system is capable of automatically isolating suspicious USB activity, notifying users about possible threats, and protecting the integrity of data and devices currently being utilized.

2. Existing System

Pre-existing antivirus and anti-malware software that protects systems from USB threats as essential parts of endpoint security. These security solutions come with functions that scan USB devices instantly upon being plugged into a system. This allows them to promptly identify any malware or harmful files saved on the USB drives, ultimately stopping the activation of dangerous code that may jeopardize the device or the larger network. This ability to scan in real time makes sure that any possible dangers are eliminated before they can cause harm. Some advanced antivirus programs not only block malware but also track the behavior of files on USB devices, detecting any abnormal or suspicious actions that could signal the existence of more complex threats like ransomware. By employing proactive defense strategies, these programs establish an essential defense system to combat the constantly changing threats presented by USB devices in the digital environment.

3. Proposed System

The system features a login system that caters to two user types: administrators and standard users. The administrator has higher levels of access, including all the functions of a regular user, as well as the power to oversee and manage user behaviors. One important feature is the option for both administrators and users to turn off or turn on the USB ports on their devices. Furthermore, the system includes a feature that tracks the location of the user trying to access it. With the right login information, a user or an admin can manage the status of the USB ports. Yet, when a user attempts to activate the USB port, the system will initially retrieve the user's location and compare it with a pre-established database. If the user's location corresponds to a recognized secure entry in the database, the system grants access to the USB port; otherwise, the request is rejected.

4. System Design

4.1. Feasibility study

The feasibility study evaluates the practicality of implementing a system designed to manage USB security and vulnerabilities. The system aims to allow administrators to enable or disable USB ports on devices, track user login locations, and restrict USB access based on these locations. This feasibility study assesses the technical, operational, economic, and legal aspects of the project to ensure its viability.

4.2. Economic feasibility

Costs at the beginning: The main expenses consist of hardware (servers for database and location tracking), development (hiring developers skilled in security and system development), and software licensing fees (for the database and any proprietary APIs).

Continuing expenses consist of server upkeep, charges for cloud storage, and routine security evaluations. Nevertheless, the increased security and decreased risk of data breaches and system exploits balance out these expenses.

4.3 Technical Feasibility

4.3.1. Requirements for the System

- ➢ **Hardware**: The system needs devices that have regular USB ports, servers to store user data and locations, and a stable internet connection for tracking locations and communicating with the database
- ➢ **Software**: The system depends on tools like Python (including Tkinter for UI, SQLAlchemy for database management, and geopy for location services), a backend database (SQLite), and encryption protocols for protecting data transmissions.
- ➢ **Security Protocols**: To ensure secure communication between the client, server, and database, SSL/TLS encryption and strong authentication mechanisms will be implemented.

5. Implementation

The USB Physical Security Software is designed to enhance the security of computer systems by regulating USB access based on predefined policies and geofencing parameters. The software integrates various security measures such as Role-Based Access Control (RBAC), geofencing, auditing, and logging to monitor and manage USB device usage across multiple users. Developed entirely using Python and Batch scripts, this software provides a user-friendly graphical interface built with Tkinter, while SQLAlchemy is employed for robust database management. This section outlines the detailed implementation of the software, describing its modular components and overall system architecture.

5.1 Module Description-

1. **Role-Based Access Control (RBAC)**

   The RBAC module is a core component of the software that restricts USB access based on user roles. Admins can define specific roles and assign USB access permissions to each role. This module ensures that only authorized users can connect USB devices, enhancing overall security. Users are categorized under roles like 'Admin' and 'User', each with different access levels. This module also includes functionality for adding new users, modifying existing roles, and setting custom policies for different user groups.

2. **Geofencing in USB Access**

The geofencing module restricts USB access based on geographical location. Using the user's location data, which is fetched using the Browser's Location API, the software checks if the user's device is within an approved geographic boundary before allowing USB access. This module works by interfacing with external location services to verify the user's location and applying geofencing rules that are set by the Admin. If a user attempts to connect a USB device outside the permitted area, access is denied and an alert is logged.

3. **Auditing and Logs**

   The auditing and logging module is essential for tracking all USB activities within the system. Every USB event, including device connection and removal, is recorded with a timestamp, user information, and action type. Logs are stored in an SQL database using SQLAlchemy, allowing for structured storage and easy retrieval of audit data. The Admin can view logs for all users, filter them based on specific criteria such as date or user role, and export them for further analysis.

4. **User and Policy Management**

   This module allows the Admin to manage users and define security policies. It provides functionalities for adding new users, assigning roles, and setting USB access policies such as read-only access or complete block for certain device types. Policies can be tailored for individual users or applied globally to user groups. The module also supports policy updates, ensuring that changes in security protocols can be quickly and efficiently applied.

5. **Graphical User Interface (GUI) with Tkinter**

   The GUI module, developed using Tkinter, provides an intuitive interface for both Admins and regular users. Admins can easily navigate through various management options such as user addition, policy setting, and log viewing. The GUI also includes interactive dialogs for user authentication, role assignment, and real-time alerts for unauthorized USB access attempts.

6. **Database Management with SQLAlchemy**

   SQLAlchemy is used for managing the software's relational database, which stores all user data, roles, policies, and logs. The ORM (Object-Relational Mapping) capabilities of SQLAlchemy allow for seamless interaction between Python code and the SQL database, facilitating complex queries and ensuring data integrity. The database schema is designed to efficiently handle large volumes of data while maintaining quick access times for auditing and reporting purposes.

5.2 System Architecture

The system architecture of the USB Physical Security Software is modular and designed for scalability. The architecture consists of the following layers:

- **Presentation Layer**: This layer includes the Tkinter-based GUI, which interacts with the end-users. It provides various interfaces for user authentication, policy management, and log viewing. All user actions are captured through this layer and forwarded to the corresponding backend services.

- **Business Logic Layer**: This is the core layer that implements the primary functionalities such as RBAC, geofencing, auditing, and user management. Python scripts handle the logic for enforcing security policies, checking geofencing conditions, and processing user requests based on their roles and permissions.
- **Data Access Layer**: Managed by SQLAlchemy, this layer facilitates all interactions with the SQL database. It handles the creation, reading, updating, and deletion (CRUD) operations on the database, ensuring that all data, including user roles, logs, and policy settings, are stored and retrieved efficiently.
- **External Interface Layer**: This layer integrates with external services for geolocation data to implement the geofencing functionality. It uses network requests to fetch IP-based location data, which is then processed to determine whether a user is within an authorized area.

Each module and layer work together to provide a comprehensive security solution for managing USB access, ensuring that sensitive data remains protected and unauthorized usage is promptly detected and reported.

6. Working

6.1 Bash Scripts

Bash Scripts are text files containing a series of commands that are executed by the Unix shell, known as "bash" (Bourne Again SHell). These scripts are often used to automate repetitive tasks, manage system operations, or configure software environments. Bash scripts are powerful tools in systems administration and software development due to their simplicity and flexibility in executing command-line instructions, processing text, managing files, and interacting with the operating system.

Usage of Bash Scripts in USB Physical Security Software

In the USB Physical Security Software, Bash scripts are utilized for several critical tasks to enhance the software's functionality and security. Here are some specific uses:

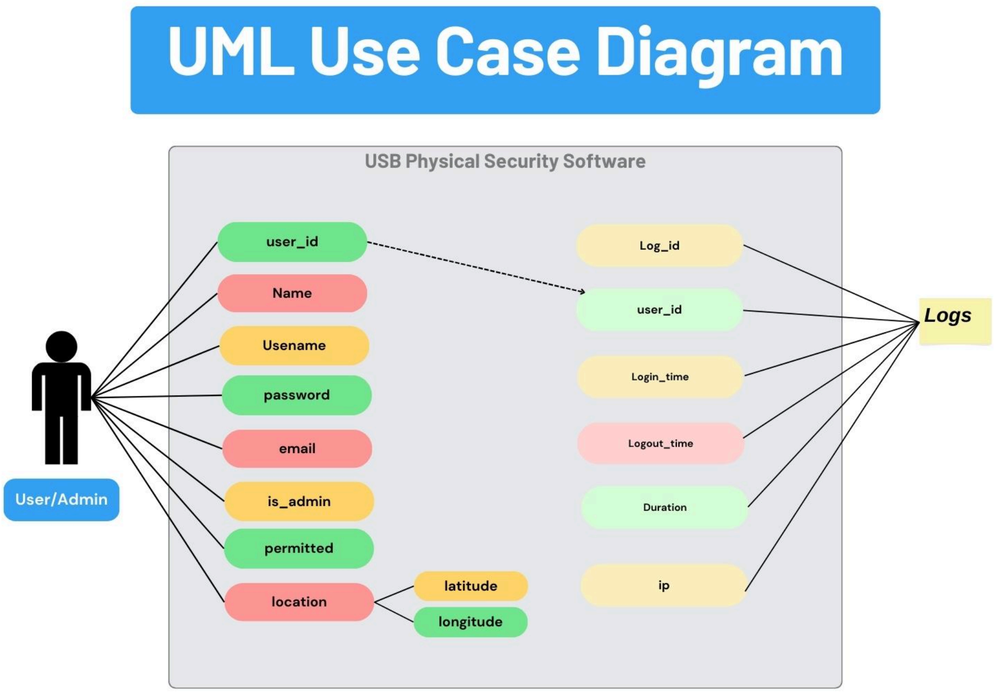
1. **Automating USB Device Monitoring**: Bash scripts are used to monitor the connection and disconnection of USB devices. By leveraging the operating system's built-in commands, these scripts detect when a USB is plugged in or removed, allowing the software to enforce security policies or log events as needed.
2. **Enforcing Security Policies**: When certain USB security policies are triggered, such as blocking unauthorized devices or enforcing geofencing rules, Bash scripts can execute commands that disable USB ports or modify user permissions dynamically.
3. **Interfacing with the Operating System**: The software uses Bash scripts to interact directly with the underlying operating system for tasks like retrieving system information (e.g., active user sessions or network configurations) that help enforce security policies or auditing.
4. **Executing System-Level Commands**: Tasks such as configuring firewall rules, managing user roles, or adjusting file permissions on connected USB drives are facilitated by Bash scripts, which run commands that would otherwise require manual intervention.

6.2 UML Diagram

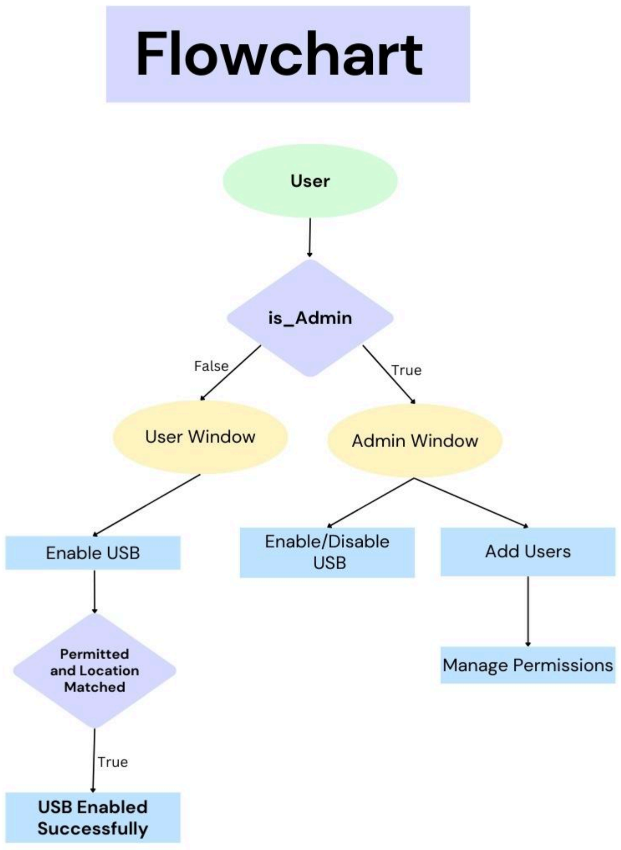
Here is the graphical implementation of the database model that is incorporated into the Software.

Here's a brief explanation of the components:

1. **User/Admin Entity**:
   - This represents the user or administrator interacting with the USB Physical Security Software. Both users and admins can access the system, but they likely have different levels of permissions and capabilities.
2. **User Attributes**:
   - The left side of the diagram lists attributes associated with a user in the system:
     - **user_id**: A unique identifier for the user.
     - **Name**: The full name of the user.
     - **Username**: The username used to log into the system.
     - **password**: The password for user authentication.
     - **email**: The user's email address.
     - **is_admin**: A boolean value indicating if the user has administrative privileges.
     - **Permit**: Indicates if the user is allowed certain permissions or access.
     - **location**: The user's location, which is further detailed by:
       - **latitude** and **longitude**: Geographic coordinates for the user's location.
3. **Log Attributes**:
   - The right side of the diagram shows the attributes related to logging user activities:
     - **Log_id**: A unique identifier for each log entry.
     - **user_id**: References the user associated with the log entry.
     - **Login_time**: The timestamp when the user logged in.
     - **Logout_time**: The timestamp when the user logs out.
     - **Duration**: The total time the user was logged in.
     - **ip**: The IP address of the user during the session.
4. **Relationships**:
   - Solid lines indicate direct associations between the user/admin and their attributes or logs.
   - A dashed line connects the user_id attribute from the user entity to the user_id in logs, showing a relationship where logs are linked to specific users.
5. **Logs**:
   - The "Logs" label at the far right represents the collection of all log entries. It suggests that logs are essential to the system, providing a record of user activities for security and auditing purposes.



6.3 Process Flow



This flowchart illustrates the decision-making process and functionality within the USB Physical Security Software. Here's a brief explanation:

1. **Start (User Node)**:
    - The flowchart begins with a **User** accessing the system. The user can be either a regular user or an admin.
2. **Decision: is_Admin**:
    - The first decision point checks if the user has administrative privileges (is_Admin).
    - If **True**, the user is an admin and the flow proceeds to the **Admin Window**.
    - If **False**, the user is a regular user, and the flow proceeds to the **User Window**.
3. **User Window**:

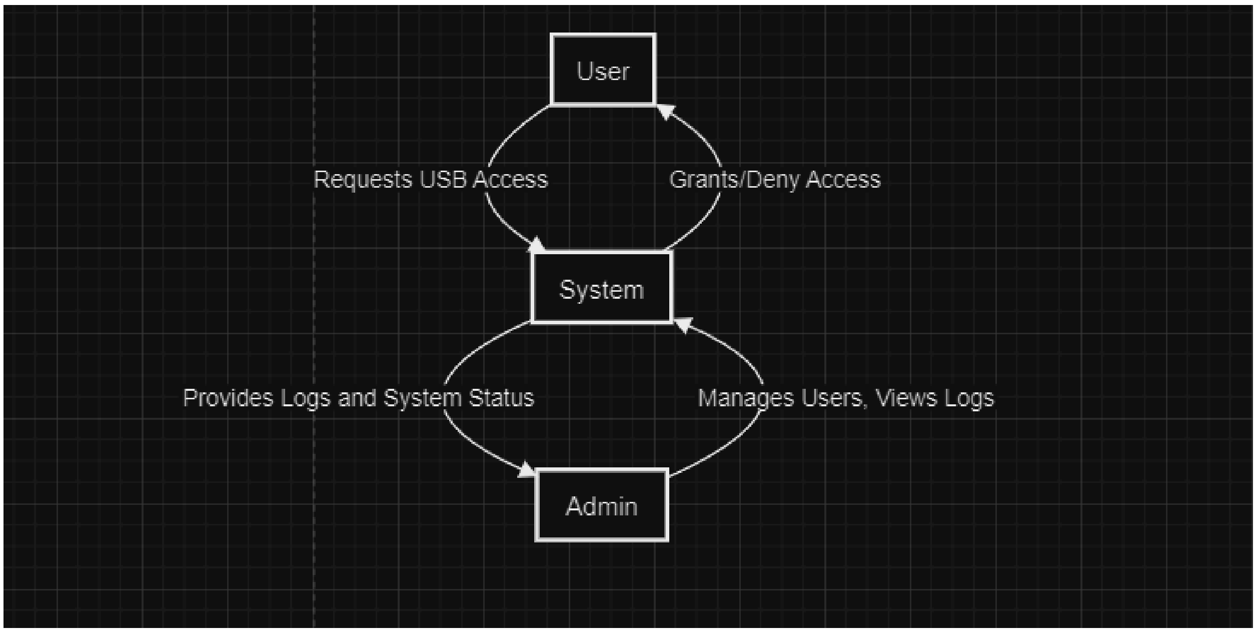
- In the **User Window**, the user has the option to **Enable USB**. This function might be used to grant themselves access to USB ports if their permissions allow it.

4. **Admin Window**:
   - The **Admin Window** offers more controls:
     - **Enable/Disable USB**: The admin can enable or disable USB access, potentially for different users.
     - **Add Users**: The admin can add new users to the system.
     - **Manage Permissions**: The admin can set or modify the permissions for existing users, such as granting or revoking access to USB ports based on various conditions.
5. **Check: Permitted and Location Matched**:
   - After the **Enable USB** action is selected, there is a check to verify if the user is permitted to enable the USB and if their location matches a predefined secure location.
   - If both conditions are **True**, the USB is enabled successfully.
6. **Outcome: USB Enabled Successfully**:
   - If all checks pass, the process concludes with the USB being enabled successfully for the user.

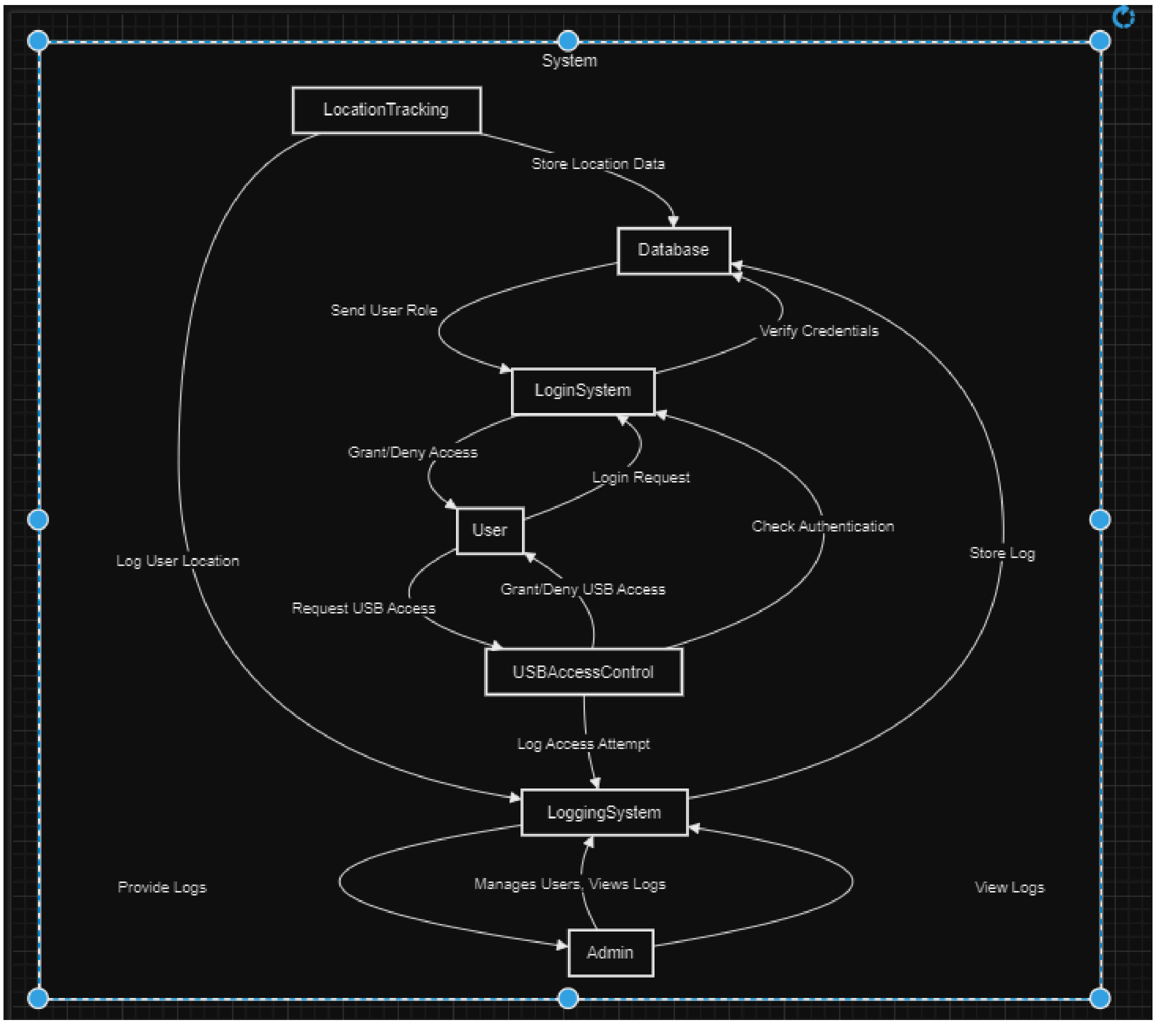
7. System Design

7.1 Data Flow Diagram

- **Level 0 (Context Diagram)**: Illustrates the interaction between the system and external entities such as users (admin and general users) and the database. It shows how data flows between the user interface, security checks, logging system, and the database.



- **Level 1 DFD**: Expands on the Context Diagram, detailing the flow of data within the system, such as data input through user login, processing for authentication, USB access control, location tracking, and storing information in the database.

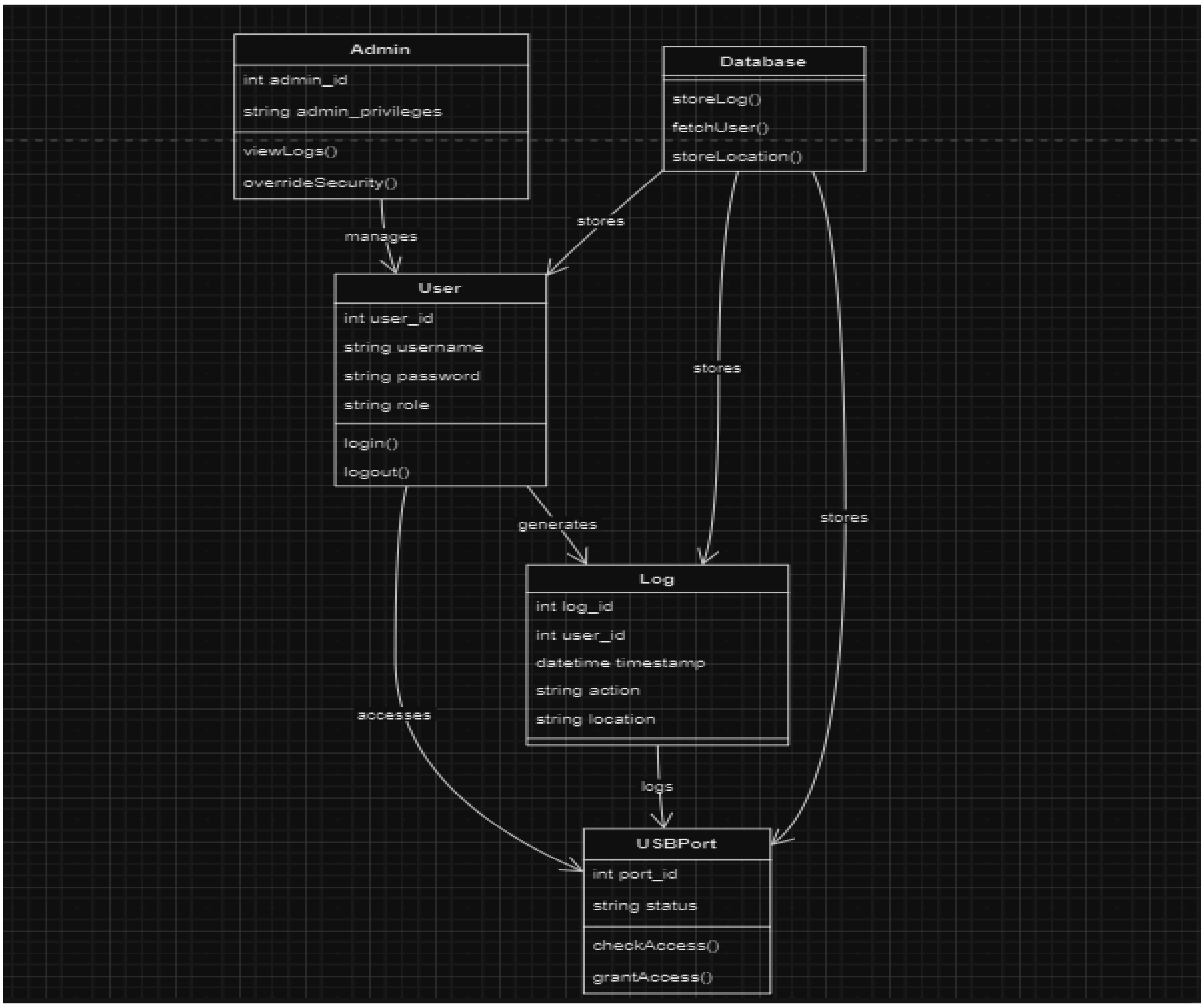


7.2 Use-Case Diagram

- **Actors**:
  - **Admin**: Manages user access, views logs, and can override USB security settings.
  - **User**: Logs in to access USB ports and is subject to security checks.
- **Use-Cases**:
  - **Login**: Both user and admin must log in to access their respective functionalities.
  - **USB Access Control**: Users request access to USB ports, which is either granted or denied based on authentication.
  - **View Logs**: The admin can view detailed logs of user activities.
  - **Location Tracking**: The system automatically tracks and logs the location of the user.

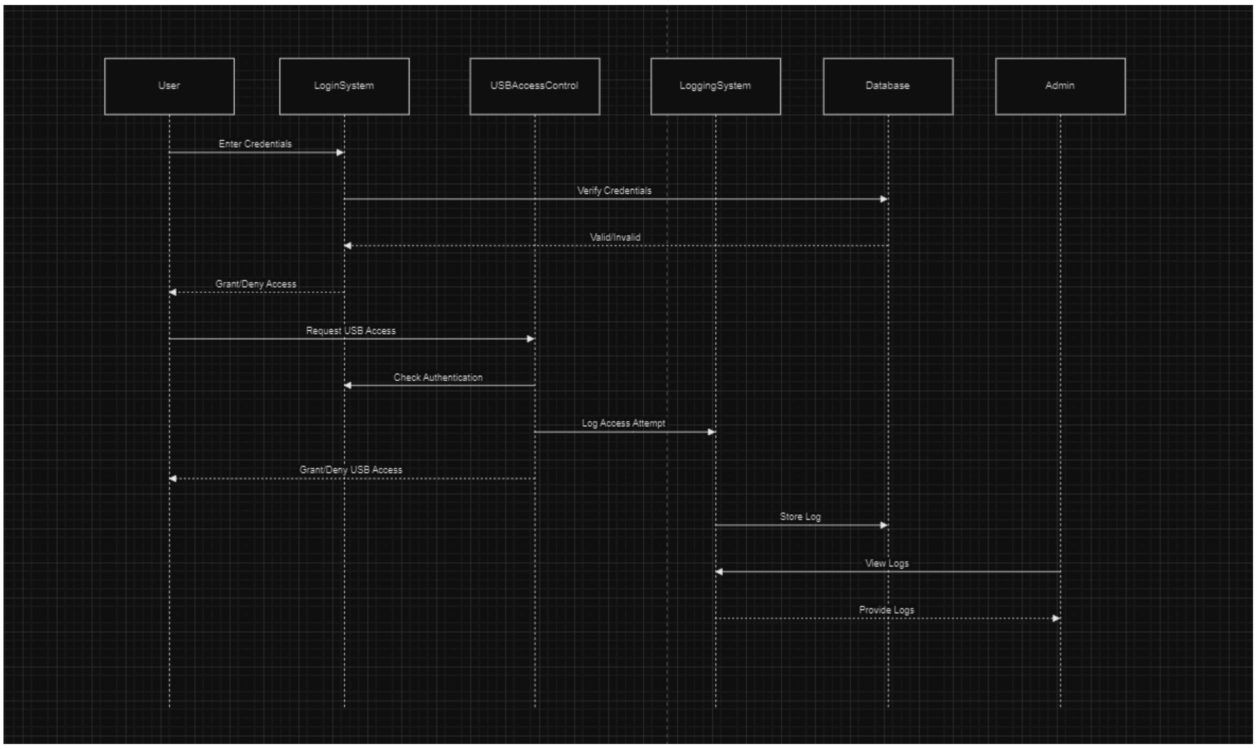
7.3 Class Diagram

- **Classes**:
  - **User**: Attributes include user_id, username, password, role, and methods like login(), and logout().
  - **Admin**: Inherits from User, with additional methods like viewLogs(), and overrideSecurity().
  - **USBPort**: Attributes include port_id, and status, with methods like checkAccess(), grantAccess().
  - **Log**: Attributes include log_id, user_id, timestamp, action, location.
  - **Database**: Handles all data storage and retrieval, with methods like storeLog(), fetchUser(), storeLocation().



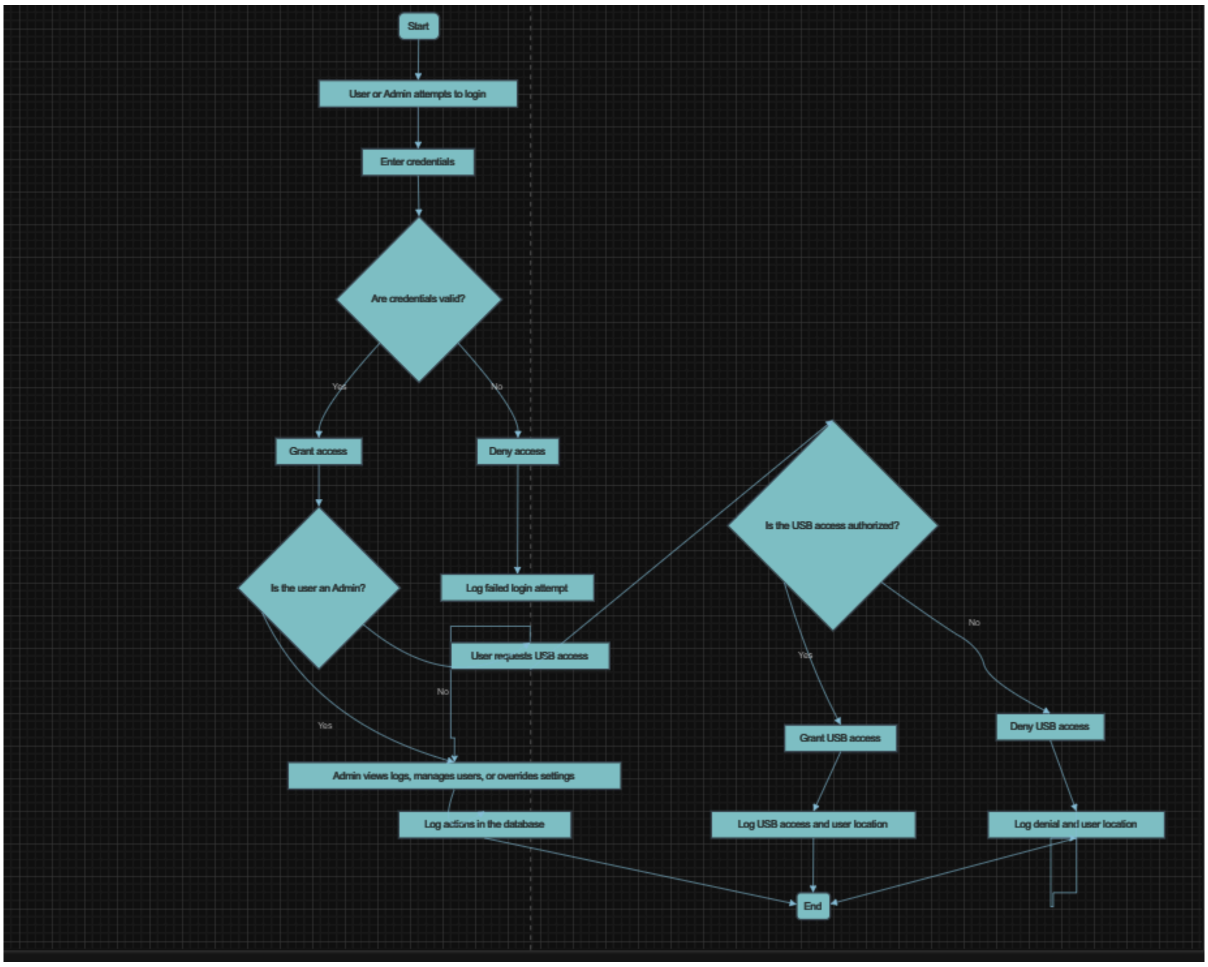
7.4 Sequence Diagram

- **Login Sequence**: This shows the interaction between the user, the login system, and the database, detailing the steps from user input to authentication and access granted.
- **USB Access Sequence**: Illustrates the steps taken when a user attempts to access a USB port, including security checks, logging, and access control.
- **Log Viewing Sequence**: Depicts the process an admin follows to view logs, from the request to data retrieval from the database.

- 

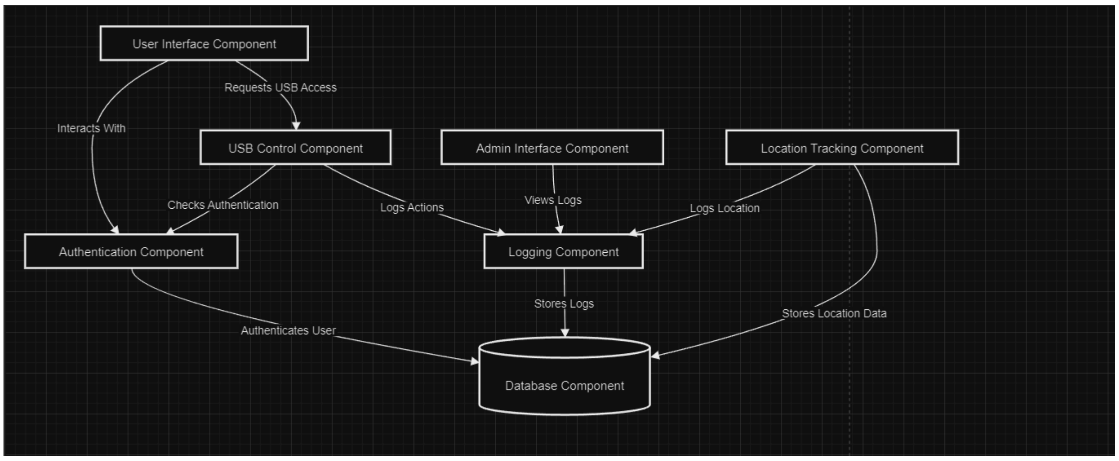
7.5 Activity Diagram

- **User Login Activity**: Details the steps a user goes through to log in, including entering credentials, system authentication, and either granting or denying access.
- **USB Access Control Activity**: Shows the process flow from the moment a user requests USB access to the decision-making process and logging of actions.
- **Admin Override Activity**: Demonstrates the workflow an admin follows to override USB security settings, including system checks and updates to the logs.


Start
User or Admin attempts to login
Enter credentials
Are credentials valid?
Yes
No
Grant access
Deny access
Is the user an Admin?
Log failed login attempt
User requests USB access
No
Yes
Admin views logs, manages users, or overrides settings
Log actions in the database
Is the USB access authorized?
Yes
No
Grant USB access
Deny USB access
Log USB access and user location
Log denial and user location
End

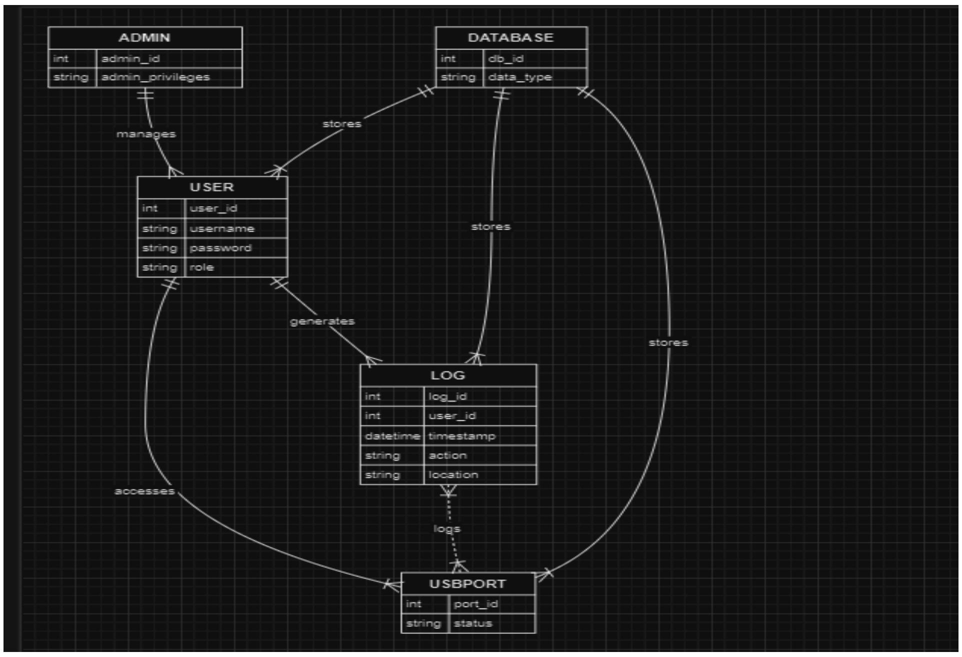
7.6 Component Diagram

- **Components**:
    - **User Interface Component**: Handles interactions between the user and the system.
    - **Authentication Component**: Manages user authentication and access control.
    - **USB Control Component**: Handles USB access requests and security enforcement.
    - **Logging Component**: Manages the creation, storage, and retrieval of logs.
    - **Database Component**: Responsible for data storage, including user credentials, logs, and location data.



7.7 ER-Diagram

- **Entities**:
    - **User**: Attributes include user_id, username, password, role.
    - **Admin**: Sub-entity of User with additional attributes like admin_privileges.
    - **Log**: Attributes include log_id, user_id, timestamp, action, location.
    - **USBPort**: Attributes include port_id, status.
- **Relationships**:
    - **User-Log**: One-to-many relationship where each user can generate multiple logs.
    - **Admin-User**: Admin has access to manage users.
    - **Log-USBPort**: Logs are associated with specific USB port actions.



8. Requirement Specification

8.1 Functional Requirements

- **User Authentication**: The system supports user and admin login functionalities. Users need to authenticate themselves before accessing the USB ports.
- **USB Port Security**: Prevents unauthorized access to USB ports to protect the system from data theft.
- **Location Tracking**: The system tracks and provides the location of the user accessing the USB ports.
- **Logging**: Logs the time and details of user logins and logouts for audit and security purposes.
- **Database Storage**: Stores user information, location data, and access logs in a secure database.

8.2 Software Requirements

- **Programming Language**: Python
- **Graphical User Interface**: Tkinter

- **Database**: SQLite or any compatible database system for storing logs and user information.
- **Libraries**: Standard Python libraries, along with any additional libraries required for location tracking, database management, and security protocols.

8.3 Operating Systems Supported

- **Windows**: Fully supported
- **Linux**: Supported with possible modifications for specific distributions
- **macOS**: Supported with minor adjustments if necessary

8.4 Technologies and Languages Used to Develop

- **Languages**: Python
- **Technologies**:
  - **Tkinter**: For creating the graphical user interface.
  - **SQLite**: For managing the database.
  - **Geolocation APIs**: For tracking the user's location.

8.5 Hardware Requirements

- **Computer System**: Any standard computer system capable of running Python.
- **USB Ports**: Functional USB ports for testing and demonstration.
- **Internet Connection**: Required for geolocation services.

9. System Test

System testing is crucial to ensure that your USB security project operates as expected, providing security, reliability, and usability. Below is an elaboration of each aspect of the system test.

9.1 Types of Test

9.1.1 Unit Testing

Objective:

The goal of unit testing is to verify the correctness of individual components or modules within the system. Each component is tested in isolation to ensure that it functions as expected. For this project, key components such as the login modules, database connections, and location tracking functions are tested separately.

- **Login Modules**: Tests will verify that both user and admin login functions correctly, including handling of valid and invalid credentials.

- **Database Connections**: Ensure that the system can connect to the database, perform CRUD (Create, Read, Update, Delete) operations, and handle exceptions gracefully.
- **Location Tracking Functions**: Test that the location tracking module accurately captures and logs the user's location.

9.1.2 Integration Testing

Objective:

Integration testing focuses on ensuring that different components of the system work together seamlessly. This involves testing the interfaces between modules, such as user authentication, USB port security, and logging, to ensure they interact correctly.

- **User Authentication and USB Port Security**: Test that once a user is authenticated, the system correctly applies USB access policies.
- **Logging System**: Ensure that all actions, including login attempts, USB access requests, and location data, are accurately logged and stored.

9.1.3 Functional Testing

Objective:

Functional testing verifies that the system meets its specified functional requirements. This involves testing the overall functionality of the system to ensure it behaves as expected in various scenarios.

- **Prevention of Unauthorized USB Access**: Verify that the system blocks unauthorized attempts to access USB ports.
- **Accurate Logging**: Ensure that all user activities and system events are logged with the correct details, such as timestamps and locations.

9.1.4 System Testing

Objective:

System testing involves testing the complete system as a whole to ensure that it meets all the specified requirements. This comprehensive test covers all integrated components, verifying that they function together correctly.

- **End-to-End Testing**: Test the entire user journey, from login to performing tasks and logging out, ensuring the system behaves as expected.
- **Performance Testing**: Assess the system's performance under various conditions, including load testing to see how the system handles multiple users and heavy database activity.

9.1.5 White Box Testing

Objective:

White box testing involves analyzing the internal structure and workings of the system. This includes reviewing the code, logic, and data flow within the application.

- **Code Reviews**: Examine the code for security vulnerabilities, inefficiencies, and logical errors.
- **Logical Path Testing**: Test all possible logical paths within the system to ensure there are no hidden bugs or vulnerabilities.

9.1.6 Black Box Testing

Objective:

Black box testing focuses on testing the system's functionality without considering its internal code structure. This method tests the system from an end-user perspective, validating inputs and outputs.

- **Input/Output Validation**: Test how the system handles various inputs (e.g., login credentials, USB device insertion) and ensure the outputs (e.g., access granted/denied, log entries) are correct.
- **User Interaction**: Test the system's behavior based on typical user interactions to ensure it meets usability and functionality expectations.

9.2 Test Strategy and Approach

9.2.1 Test Objectives

The test strategy is designed to ensure that the system meets its primary objectives in terms of security, reliability, and usability.

- **Ensure Security**: The primary objective is to validate that the system effectively prevents unauthorized access to USB ports and other sensitive areas.
- **Reliability**: The system should reliably log all user activities and track locations accurately without any data loss or corruption.
- **Usability**: The user interface should be intuitive and easy to navigate, ensuring that users can operate the system without extensive training.

9.2.2 Features to be Tested

Key features of the system that are crucial to its overall functionality and security are identified for thorough testing.

- **Login Functionality**: Test the processes for both user and admin logins to ensure proper authentication and access control.
- **USB Access Control**: Verify that the system effectively blocks unauthorized attempts to access USB ports while allowing authorized access.

- **Location Tracking**: Test the accuracy and reliability of the location data captured and stored by the system.
- **Logging**: Ensure that all activities, including login attempts, USB access, and location tracking, are properly logged with accurate details.

9.3 Integration Testing

9.3.1 Test Results

Result:

The integration of various modules such as user login, location tracking, and logging has been tested to ensure compatibility and functionality. The tests confirmed that the modules interact as expected, with no significant issues detected. All components work together seamlessly, providing a secure and reliable system.

9.4 Acceptance Testing

9.4.1 Test Results

Result:

The system has undergone rigorous testing against the project's requirements and has met all defined acceptance criteria. This indicates that the system is ready for deployment, as it successfully prevents unauthorized USB access, accurately logs user activities, and provides a user-friendly interface.

10. Conclusion

The USB security project successfully protects the system from unauthorized access to USB ports, ensuring data security and integrity. With its robust logging, location tracking, and user authentication features, the system is an effective cybersecurity tool. Through comprehensive testing, the system has been validated to meet all functional and security requirements.