SecureVault: A Flask-Based Dashboard for Leak Detection and Visualization

Problem Statement & Project Goals

Problem Statement

In the digital era, organizations face increasing threats from data leaks, which can compromise sensitive information and damage reputations. Traditional methods of monitoring such leaks often lack real-time visualization and user-friendly interfaces, making it challenging for security teams to respond promptly and effectively.

Project Goals

- Develop a secure, web-based dashboard to monitor and visualize data leaks in real-time.
- Implement user authentication to ensure that only authorized personnel can access sensitive leak information.
- Provide interactive charts and tables to analyze leak patterns over time and by type.
- Enable data filtering and export functionalities to assist in detailed analysis and reporting.

Architecture & Design Decisions

System Architecture

The SecureVault system follows a modular architecture comprising the following components:

1. **Frontend**: Built using HTML, CSS (Bootstrap), and JavaScript, providing an intuitive user interface.

2. **Backend**: Developed with Flask, a lightweight Python web framework, handling routing, authentication, and data processing.
3. **Database**: Utilizes SQLite for storing user credentials and leak data.
4. **Visualization**: Employs Chart.js for rendering dynamic charts based on leak data.

Design Decisions

- **Flask Framework**: Chosen for its simplicity and flexibility, allowing rapid development and easy integration with other Python libraries.
- **Chart.js**: Selected for its ability to create responsive and interactive charts with minimal configuration.
- **Bootstrap**: Used to ensure a responsive and consistent design across different devices and screen sizes.
- **Authentication**: Implemented using Flask-Login to manage user sessions securely.

Key Code Snippets and Flow Diagrams

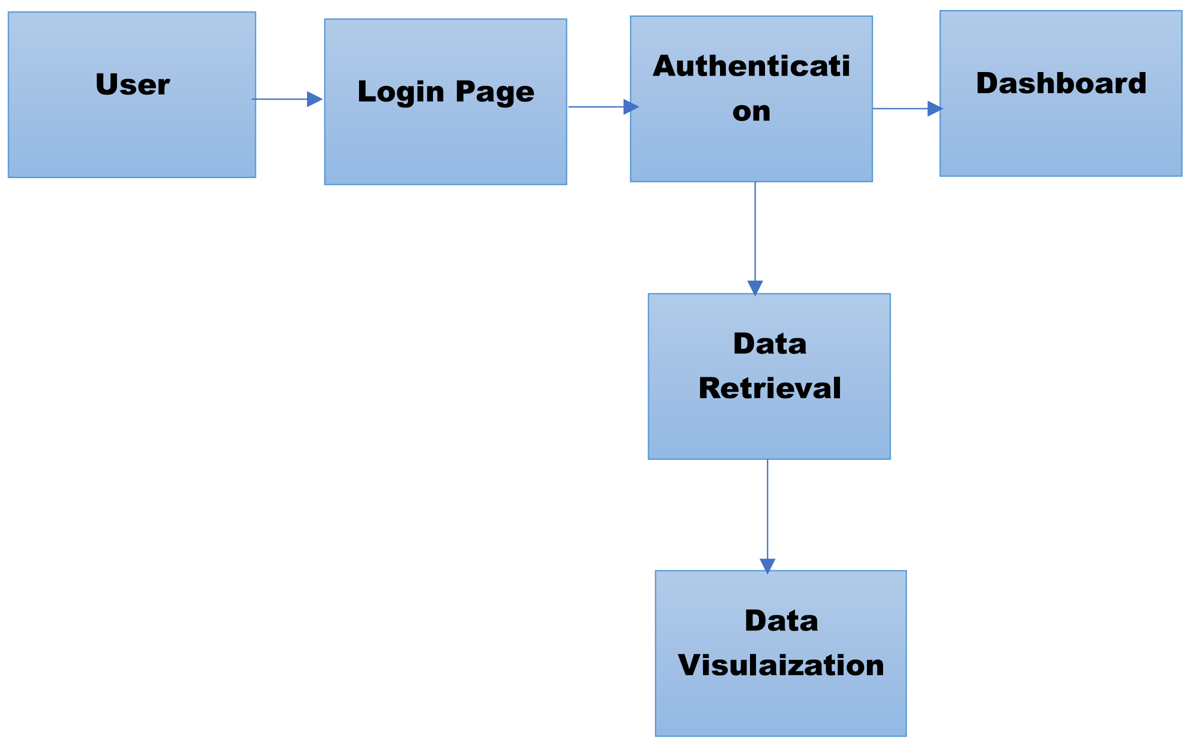
User Authentication

```
from flask_login import LoginManager, login_user, login_required, logout_user

login_manager = LoginManager()

login_manager.init_app(app)

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

def login():

    # Authentication logic here…..
```

Data Visualization

```
var ctx = document.getElementById('leakChart').getContext('2d');
var leakChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: [...],
        datasets: [{
            label: 'Leak Count',
            data: [...],
            backgroundColor: 'rgba(75, 192, 192, 0.2)',
            borderColor: 'rgba(75, 192, 192, 1)',
            borderWidth: 1
        }]
    },
    options: {
        responsive: true,
        ...
    }
});
```

Flow Diagram



ML/NLP Logic and Policy Configuration

ML/NLP Integration

While the current iteration of SecureVault focuses on data visualization and monitoring, future enhancements may include:

- **Anomaly Detection**: Implementing machine learning models to identify unusual patterns in leak data.
- **Natural Language Processing**: Analyzing textual data from leaks to categorize and prioritize them based on content.

Policy Configuration

Policies can be configured to:

- Trigger alerts when certain thresholds are exceeded.
- Restrict access to specific data based on user roles.
- Schedule regular reports for compliance and auditing purposes.

Challenges Faced and Solutions

Challenge 1: Ensuring Secure Authentication

- **Issue**: Implementing a secure and reliable authentication system.
- **Solution**: Utilized Flask-Login to manage user sessions and protect routes, ensuring that only authenticated users can access sensitive data.

Challenge 2: Dynamic Data Visualization

- **Issue**: Rendering real-time charts that update based on user input and data changes.
- **Solution**: Integrated Chart.js to create responsive charts that fetch and display data dynamically using AJAX calls.

Challenge 3: Data Export Functionality

- **Issue**: Allowing users to export filtered data without server errors.
- **Solution**: Implemented a route that converts filtered data into CSV format and uses Flask's Response object to facilitate downloads without errors.

Conclusion

SecureVault successfully addresses the need for a secure and interactive platform to monitor data leaks. By combining robust authentication mechanisms with dynamic data visualization, it empowers security teams to respond swiftly to potential threats. Future enhancements, including ML and NLP integrations, will further augment its capabilities, making it an indispensable tool in the cybersecurity landscape.