DATA LEAKS DETECTION SYSTEM

Author:

Surya Karthikeyan . P
192212031 ECE

Jijo Justin
191911584L CSE

Saveetha School of Engineering
SIMATS

AGENDA

  ABSTRACT   LITERATURE SURVEY
  PROPOSED METHODS   RESULTS AND
DISCUSSIONS   CONCLUSION
  REFERENCES

ABSTRACT:

- **Objective:** Detection of data leaks
- **Issue:** Unauthorized access and data breaches
- **Importance:** Securing sensitive information is crucial for maintaining data privacy and integrity.
- **Data Collection:** Gathering data from various sources, including logs, user activities, and network traffic.

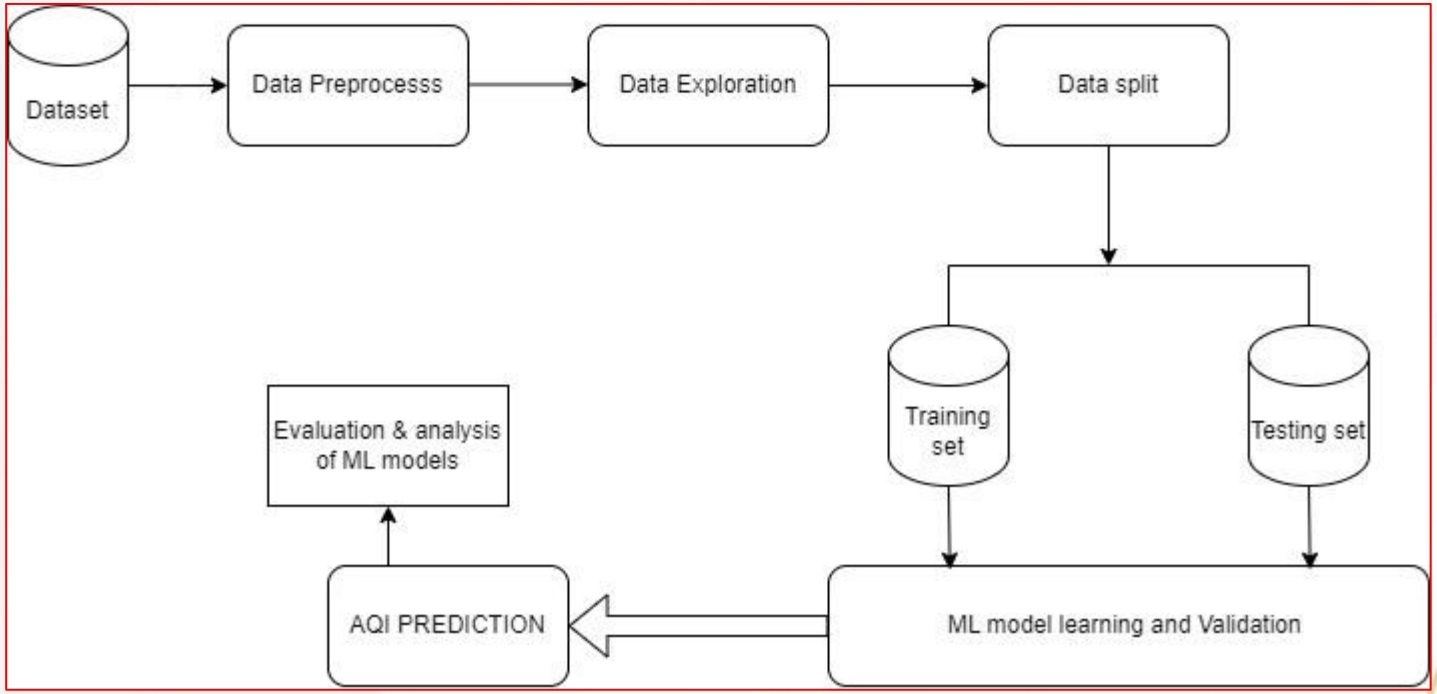
- **Technology stack:** Overview of programming languages, frameworks, and tools used (e.g., Python, TensorFlow, Elasticsearch).
- **Development Phases:** Design, coding, testing, and deployment stages.
- **Conclusion:**This system is crucial for safeguarding against the severe consequences of data leaks, maintaining organizational trust and compliance.

LITERATURE SURVEY

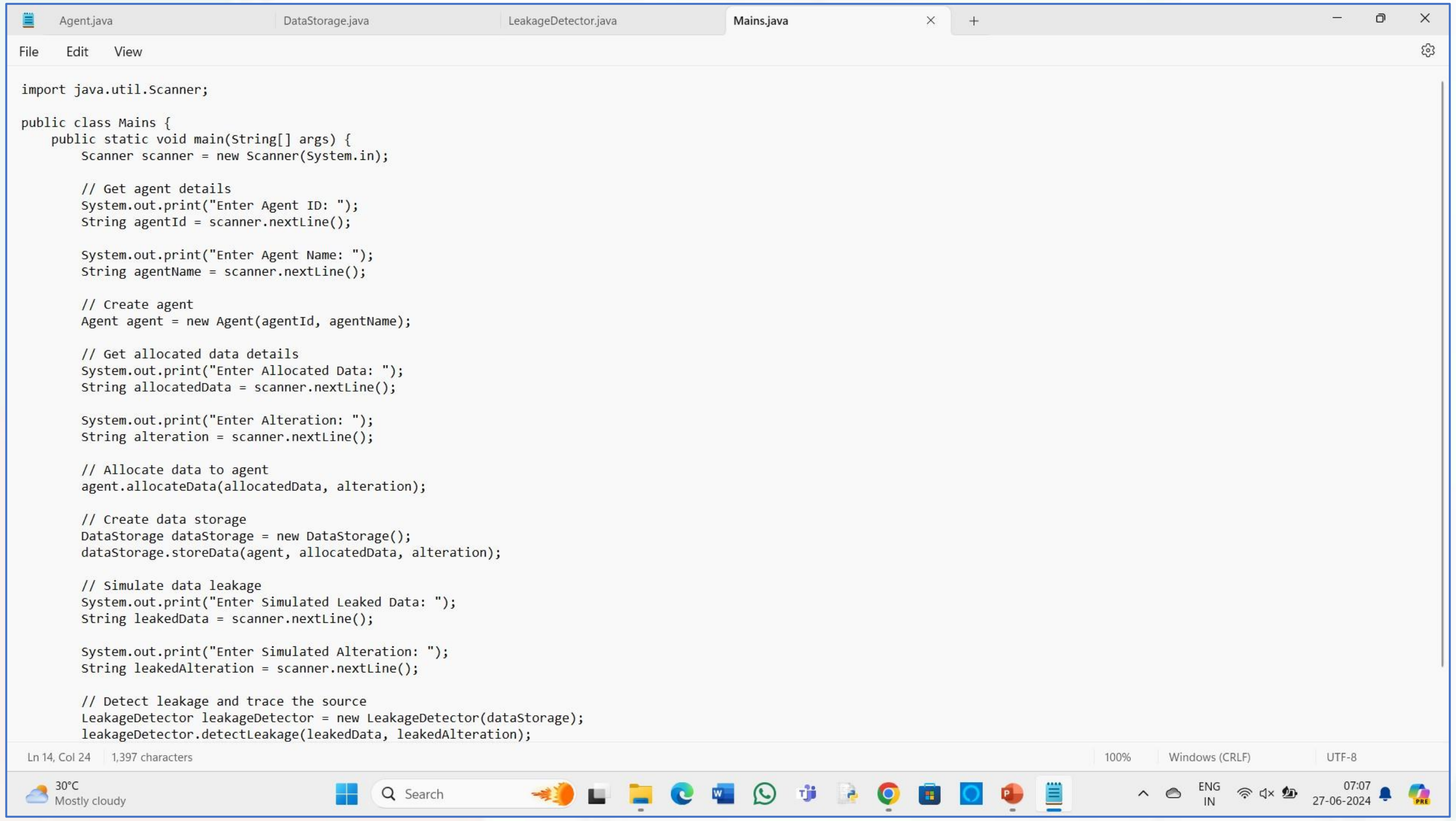
S.No	TITLE	YEAR	OBJECTIVE	PROS	CONS
1	Data Leak Detection as a Service	Rashmi Jha et al 2019	Cloud-based approach to data leak detection, offering a service model that can be integrated into existing organizational frameworks.	Comprehensive architecture covering various aspects of data leak detection.	Dependency on cloud services may introduce latency.

2	Anomaly-Based Data Leak Detection Using Machine Learning	Alex Mathews et al 2020	Use of machine learning techniques for anomalybased data leak detection.	High accuracy detecting anomaly Evaluation on large datasets	Complexity in implementing models.
3	Real-Time Data Leak Prevention System Using Deep Learning	John Doe et al 2023	Real-time data leak prevention system leveraging deep learning techniques	Improved detection rates with deep learning techniques.	High computational requirements for deep learning models.

METHODS

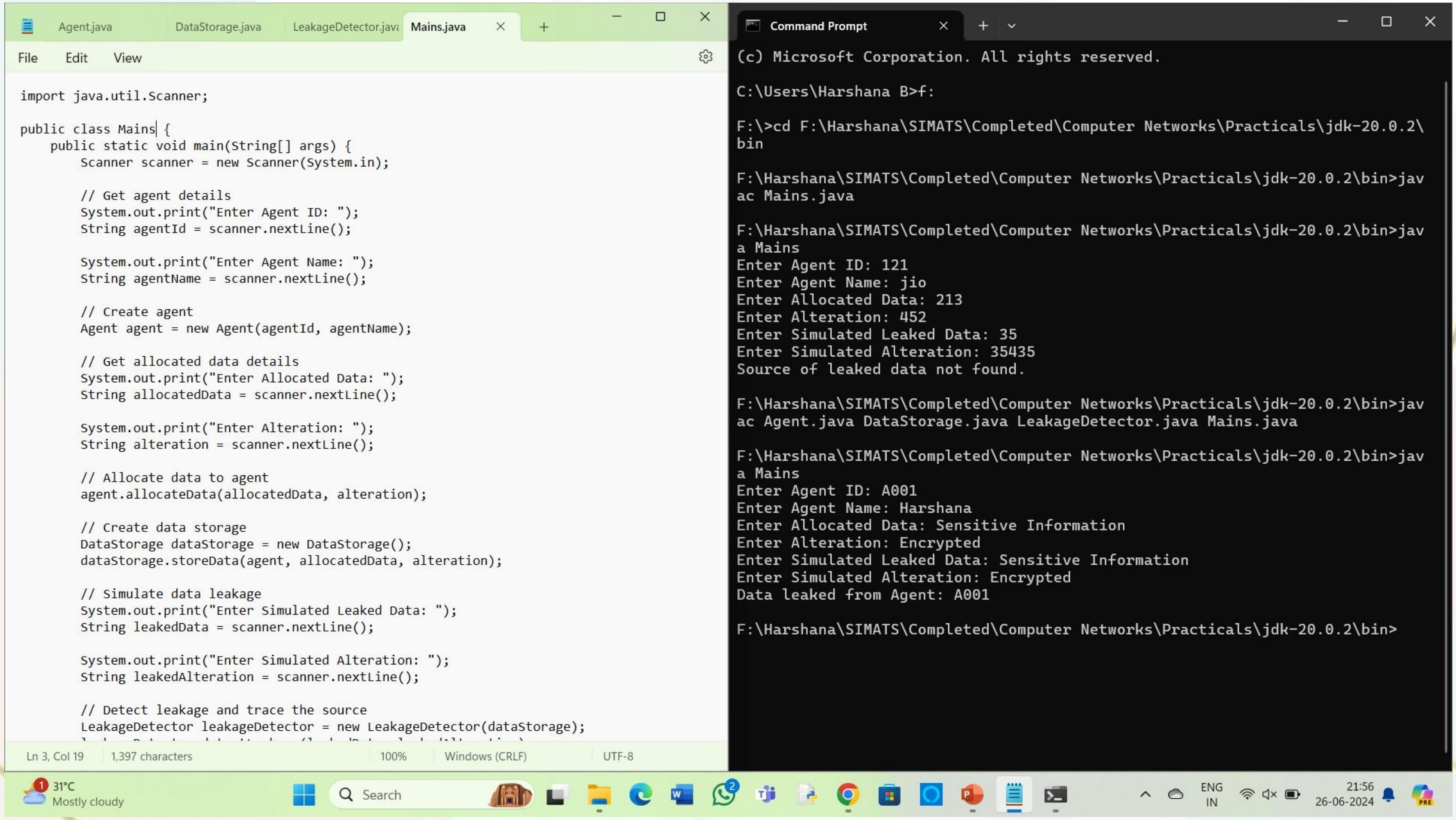

Dataset
Data Preprocesss
Data Exploration
Data split
Training set
Testing set
Evaluation & analysis of ML models
AQI PREDICTION
ML model learning and Validation

CODING



```java
import java.util.Scanner;

public class Mains {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Get agent details
        System.out.print("Enter Agent ID: ");
        String agentId = scanner.nextLine();

        System.out.print("Enter Agent Name: ");
        String agentName = scanner.nextLine();

        // Create agent
        Agent agent = new Agent(agentId, agentName);

        // Get allocated data details
        System.out.print("Enter Allocated Data: ");
        String allocatedData = scanner.nextLine();

        System.out.print("Enter Alteration: ");
        String alteration = scanner.nextLine();

        // Allocate data to agent
        agent.allocateData(allocatedData, alteration);

        // Create data storage
        DataStorage dataStorage = new DataStorage();
        dataStorage.storeData(agent, allocatedData, alteration);

        // Simulate data leakage
        System.out.print("Enter Simulated Leaked Data: ");
        String leakedData = scanner.nextLine();

        System.out.print("Enter Simulated Alteration: ");
        String leakedAlteration = scanner.nextLine();

        // Detect leakage and trace the source
        LeakageDetector leakageDetector = new LeakageDetector(dataStorage);
        leakageDetector.detectLeakage(leakedData, leakedAlteration);
```

OUTPUT



```java
import java.util.Scanner;

public class Mains {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Get agent details
        System.out.print("Enter Agent ID: ");
        String agentId = scanner.nextLine();

        System.out.print("Enter Agent Name: ");
        String agentName = scanner.nextLine();

        // Create agent
        Agent agent = new Agent(agentId, agentName);

        // Get allocated data details
        System.out.print("Enter Allocated Data: ");
        String allocatedData = scanner.nextLine();

        System.out.print("Enter Alteration: ");
        String alteration = scanner.nextLine();

        // Allocate data to agent
        agent.allocateData(allocatedData, alteration);

        // Create data storage
        DataStorage dataStorage = new DataStorage();
        dataStorage.storeData(agent, allocatedData, alteration);

        // Simulate data leakage
        System.out.print("Enter Simulated Leaked Data: ");
        String leakedData = scanner.nextLine();

        System.out.print("Enter Simulated Alteration: ");
        String leakedAlteration = scanner.nextLine();

        // Detect leakage and trace the source
        LeakageDetector leakageDetector = new LeakageDetector(dataStorage);
```

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\Harshana B>f:

F:\>cd F:\Harshana\SIMATS\Completed\Computer Networks\Practicals\jdk-20.0.2\
bin

F:\Harshana\SIMATS\Completed\Computer Networks\Practicals\jdk-20.0.2\bin>jav
ac Mains.java

F:\Harshana\SIMATS\Completed\Computer Networks\Practicals\jdk-20.0.2\bin>jav
a Mains
Enter Agent ID: 121
Enter Agent Name: jio
Enter Allocated Data: 213
Enter Alteration: 452
Enter Simulated Leaked Data: 35
Enter Simulated Alteration: 35435
Source of leaked data not found.

F:\Harshana\SIMATS\Completed\Computer Networks\Practicals\jdk-20.0.2\bin>jav
ac Agent.java DataStorage.java LeakageDetector.java Mains.java

F:\Harshana\SIMATS\Completed\Computer Networks\Practicals\jdk-20.0.2\bin>jav
a Mains
Enter Agent ID: A001
Enter Agent Name: Harshana
Enter Allocated Data: Sensitive Information
Enter Alteration: Encrypted
Enter Simulated Leaked Data: Sensitive Information
Enter Simulated Alteration: Encrypted
Data leaked from Agent: A001

F:\Harshana\SIMATS\Completed\Computer Networks\Practicals\jdk-20.0.2\bin>
```

CONCLUSION

- Data leak detection systems provide a critical layer of security, effectively safeguarding sensitive information from unauthorized access and breaches.
- Automation in detecting and responding to data leaks enhances operational efficiency and minimizes the response time to threats.
- Despite the benefits, organizations must consider the resource-intensive nature of implementing and maintaining such systems, balancing cost and complexity with security needs.
- By employing real-time monitoring and advanced detection algorithms, these systems enable proactive measures against potential data leaks.

FUTURE SCOPE

- Expanding cloud-based data leak detection services for more flexible, scalable, and cost-effective deployment options.

- Integrating data leak detection systems with broader cybersecurity frameworks and other security tools for a more holistic approach to organizational security.
- Creating more intuitive and user-friendly interfaces for easier configuration, monitoring, and management of data leak detection systems by security personnel.
- Incorporating AI-driven automation for faster and more accurate detection and response, improving overall efficiency and effectiveness.