A picture containing text, clipart

Description automatically generatedText, logo

Description automatically generated

CSA0989 PROGRAMMING IN JAVA

## Smart Traffic Signal Optimization:

## Smart Traffic Signal Optimization aims to improve traffic flow and reduce congestion using real-time data and adaptive signal control. The system uses sensors and cameras to gather traffic data, processes it with algorithms, and adjusts traffic signals dynamically. Here are key components and benefits of such a system.

**Pseudo Code Implementation**

Initialize sensors and cameras

Initialize traffic signal controllers

Initialize central processing unit (CPU)

Function collectTrafficData():

For each sensor and camera:

Read traffic data

Send data to CPU

Function processTrafficData(data):

Analyze traffic patterns

Predict traffic flow

Calculate optimal signal timings

Function updateTrafficSignals(timings):

For each traffic signal controller:

Set signal timings based on calculated timings

Function main():

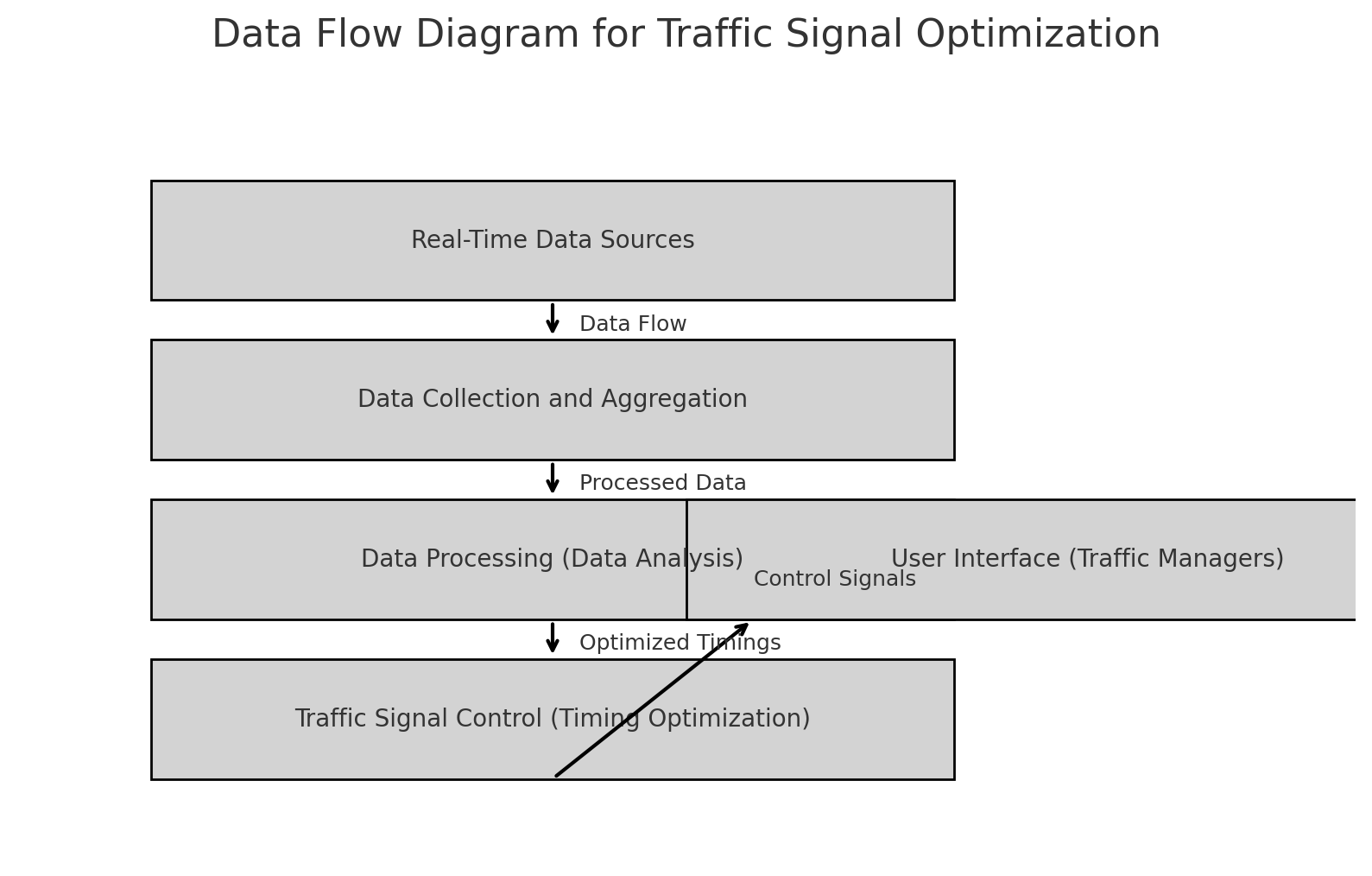
While system is running:

trafficData = collectTrafficData()

signalTimings = processTrafficData(trafficData)

updateTrafficSignals(signalTimings)

**Data Flow Diagram**



**Documentation:**

* **Modular Design:** Separate classes for data (IntersectionData) and control (TrafficSignalController) ensure a clean, maintainable structure.
* **Default Timings:** Standard signal timings are used as a safe fallback.
* **Data-Driven Adjustments:** Real-time data adjusts signal timings dynamically.
* **Manual Override:** Allows traffic managers to manually adjust timings if necessary.
* **HashMap:** Efficiently stores and retrieves data for intersections and controllers.
* **Custom Classes:** Encapsulate relevant data and methods for managing traffic signals.

**CONCLUSION**

* **Real-Time Data Sources:**

Collects data from sensors, CCTV, etc.

* **Data Collection and Aggregation:**

Aggregates data like vehicle count, speed, queue length, and pedestrian count.

* **Data Processing:**

Analyses data to determine optimal signal timings.

* **Traffic Signal Control:**

Adjusts signal timings based on processed data.

* **User Interface (UI):**

Displays information for traffic managers to monitor and manually adjust settings

**User Interface:**

**-Dashboard**: A user-friendly interface displays real-time traffic conditions and signal timings, providing traffic managers with actionable insights.

- **Alerts**: Notifications or alerts are sent to traffic managers for any anomalies or necessary interventions, ensuring timely responses to unexpected situations.

In conclusion, this smart traffic signal management system integrates advanced technologies and data-driven approaches to optimize traffic flow in busy city environments. By continuously collecting, analysing, and acting on real-time data, the system can significantly reduce congestion, enhance travel efficiency, and improve overall urban mobility.

Code:

import java.util.\*;

class TrafficSignalOptimization

{

    private static class IntersectionData

    {

        int vehicleCount, averageSpeed, queueLength, pedestrianCount;

        IntersectionData(int vehicleCount, int averageSpeed, int queueLength, int pedestrianCount)

        {

            this.vehicleCount = vehicleCount;

            this.averageSpeed = averageSpeed;

            this.queueLength = queueLength;

            this.pedestrianCount = pedestrianCount;

        }

    }

    private static class TrafficSignalController

    {

        String intersectionId;

        int greenLightDuration;

        int redLightDuration;

        int pedestrianSignalDuration;

        TrafficSignalController(String intersectionId)

        {

            this.intersectionId = intersectionId;

            this.greenLightDuration = 30;

            this.redLightDuration = 30;

            this.pedestrianSignalDuration = 10;

        }

        void updateTimings(int greenDuration, int redDuration, int pedestrianDuration)

        {

            this.greenLightDuration = greenDuration;

            this.redLightDuration = redDuration;

            this.pedestrianSignalDuration = pedestrianDuration;

            System.out.println("Updated timings for intersection " + intersectionId + ":");

            System.out.println("Green Light Duration: " + greenLightDuration + " seconds");

            System.out.println("Red Light Duration: " + redLightDuration + " seconds");

            System.out.println("Pedestrian Signal Duration: " + pedestrianSignalDuration + " seconds");

        }

    }

    public static void main(String args[])

    {

        Map<String, IntersectionData>

        intersections = new HashMap<>();

        intersections.put("Intersection1",

        new IntersectionData(50, 45, 10, 20));

        intersections.put("Intersection2",

        new IntersectionData(30, 35, 5, 10));

        Map<String, TrafficSignalController> controllers = new HashMap<>();

        for (String id : intersections.keySet())

        {

            controllers.put(id, new TrafficSignalController(id));

        }

        for (Map.Entry<String, IntersectionData> entry : intersections.entrySet())

        {

            String intersectionId = entry.getKey();

            IntersectionData data = entry.getValue();

            TrafficSignalController controller = controllers.get(intersectionId);

            int greenDuration = 30;

            int redDuration = 30;

            int pedestrianDuration = 10;

            if (data.vehicleCount > 40)

            {

                greenDuration = 60;

                redDuration = 20;

            }

            if (data.pedestrianCount > 15)

            {

                pedestrianDuration = 30;

            }

            controller.updateTimings(greenDuration, redDuration, pedestrianDuration);

        }

        Scanner sc=new Scanner(System.in);

        System.out.println("Adjust signal timings:");

        System.out.print("Enter intersection ID: ");

        String intersectionId = sc.nextLine();

        System.out.print("Enter new green light duration (seconds): ");

        int newGreenDuration = sc.nextInt();

        System.out.print("Enter new red light duration (seconds): ");

        int newRedDuration = sc.nextInt();

        System.out.print("Enter new pedestrian signal duration (seconds): ");

        int newPedestrianDuration = sc.nextInt();

        TrafficSignalController userController = controllers.get(intersectionId);

        if (userController != null)

        {

            userController.updateTimings(newGreenDuration, newRedDuration, newPedestrianDuration);

        }

        else

        {

            System.out.println("Intersection ID not found.");

        }

        sc.close();

    }

}