**JAVA ASSIGNMENT**

**Smart Traffic Signal Optimization**

Scenario: You are part of a team working on an initiative to optimize traffic signal management in a busy city to reduce congestion and improve traffic flow efficiency using smart technologies.

**Data Collection and Modeling:**

**Data Structure Definition:** To collect real-time traffic data from sensors, you need a well-defined data structure. Consider using classes in Java to represent different entities:

class TrafficData {

int vehicleCount;

double vehicleSpeed;

String intersectionId;

long timestamp;

}

**Traffic Signal Optimization Algorithm:** You'll need algorithms that consider various factors such as traffic density, vehicle queues, peak hours, and pedestrian crossings. Here's an outline of the steps involved in the algorithm:

1. **Data Ingestion:** Collect real-time data from sensors.
2. **Traffic Analysis:** Analysis the data to determine traffic density and queue lengths.
3. **Signal Timing Adjustment:** Adjust signal timings based on analysis data.
4. **Peak Hour Consideration:** Give priority to higher traffic volumes during peak hours.
5. **Pedestrian Crossings:** Ensure pedestrian crossing times are adequately managed.

function optimizeTrafficSignals(intersectionData):

for each intersection in intersectionData:

currentTraffic = getCurrentTraffic(intersection)

if isPeakHour(currentTime):

adjustSignalTiming(intersection, currentTraffic, peakHour=True)

else:

adjustSignalTiming(intersection, currentTraffic, peakHour=False)

if pedestrianWaiting(intersection):

adjustForPedestrianCrossing(intersection)

**Visualization and Reporting**

**Real-time Monitoring:** You can use JavaFX or a web-based dashboard (e.g., using Spring Boot and Thymeleaf) to visualize real-time traffic conditions and signal timings.

**Reporting:** Generate reports on metrics like traffic flow improvements, average wait times, and overall congestion reduction. Libraries like Apache POI can help generate Excel reports.

**User Interaction**

**User Interface Design:** Design an intuitive UI using JavaFX or a web-based interface for traffic managers and city officials. Include features for real-time monitoring, manual signal adjustment, and performance metric viewing.

**CODE :**

**package com.example.TrafficLight;**

**import javafx.animation.KeyFrame;**

**import javafx.animation.Timeline;**

**import javafx.application.Application;**

**import javafx.scene.Scene;**

**import javafx.scene.layout.StackPane;**

**import javafx.scene.paint.Color;**

**import javafx.scene.shape.Circle;**

**import javafx.scene.layout.VBox;**

**import javafx.stage.Stage;**

**import javafx.util.Duration;**

**import java.io.IOException;**

**class TrafficLight extends Application {**

**public void start(Stage primaryStage) {**

**Circle redLight = new Circle(50, Color.RED);**

**Circle yellowLight = new Circle(50, Color.GRAY);**

**Circle greenLight = new Circle(50, Color.GRAY);**

**VBox root = new VBox(10);**

**root.getChildren().addAll(redLight, yellowLight, greenLight);**

**Scene scene = new Scene(root, 200, 600);**

**primaryStage.setTitle("Traffic Signal Animation");**

**primaryStage.setScene(scene);**

**primaryStage.show();**

**Timeline timeline = new Timeline(**

**new KeyFrame(Duration.seconds(0), e -> {**

**redLight.setFill(Color.RED);**

**yellowLight.setFill(Color.GRAY);**

**greenLight.setFill(Color.GRAY);**

**}),**

**new KeyFrame(Duration.seconds(3), e -> {**

**redLight.setFill(Color.GRAY);**

**yellowLight.setFill(Color.YELLOW);**

**greenLight.setFill(Color.GRAY);**

**}),**

**new KeyFrame(Duration.seconds(6), e -> {**

**redLight.setFill(Color.GRAY);**

**yellowLight.setFill(Color.GRAY);**

**greenLight.setFill(Color.GREEN);**

**}),**

**new KeyFrame(Duration.seconds(9), e -> {**

**redLight.setFill(Color.RED);**

**yellowLight.setFill(Color.GRAY);**

**greenLight.setFill(Color.GRAY);**

**})**

**);**

**timeline.setCycleCount(Timeline.INDEFINITE);**

**timeline.play();**

**}**

**public static void main(String[] args) {**

**launch(args);**

**}**

**OUTPUT:**

****

**Deliverables**

1. **Data Flow Diagram:**
   * Illustrate the data flow from sensors to the central system, processing, and feedback to traffic signals.
2. **Pseudocode and Implementation:**
   * Provide detailed pseudocode and corresponding Java code for your algorithms.
3. **Documentation:**
   * Explain your design decisions, data structures, assumptions, and potential improvements.
4. **User Interface:**
   * Develop and document the UI for traffic managers and city officials.
5. **Testing:**
   * Create comprehensive test cases to validate the system under different scenarios.