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.

Tasks:

1. **Data Collection and Modeling:**
    - Define the data structure to collect real-time traffic data from sensors (e.g., vehicle counts, speeds) at various intersections across the city.

```
import java.time.LocalDateTime

public class TrafficData {

    private int intersectionId;

    private int vehicleCount;

    private double averageSpeed;

    private LocalDateTime timestamp

    public TrafficData(int intersectionId, int vehicleCount, double averageSpeed,
LocalDateTime timestamp) {

        this.intersectionId = intersectionId;

        this.vehicleCount = vehicleCount;

        this.averageSpeed = averageSpeed;

        this.timestamp = timestamp;

    }

    public int getIntersectionId() {

        return intersectionId;

    }
```

```
public void setIntersectionId(int intersectionId) {
    this.intersectionId = intersectionId;
}
public int getVehicleCount() {
    return vehicleCount;
}
public void setVehicleCount(int vehicleCount) {
    this.vehicleCount = vehicleCount;
}
public double getAverageSpeed() {
    return averageSpeed;
}
public void setAverageSpeed(double averageSpeed) {
    this.averageSpeed = averageSpeed;
}
public LocalDateTime getTimestamp() {
    return timestamp;
}
public void setTimestamp(LocalDateTime timestamp) {
    this.timestamp = timestamp;
}
public String toString() {
    return "TrafficData{" +
            "intersectionId=" + intersectionId +
```

```
            ", vehicleCount=" + vehicleCount +
            ", averageSpeed=" + averageSpeed +
            ", timestamp=" + timestamp +
            '}';
    }
}
```

2. **Algorithm Design:**
    - Develop algorithms to analyze the collected data and optimize traffic signal timings dynamically based on current traffic conditions.
    - Consider factors such as traffic density, vehicle queues, peak hours, and pedestrian crossings in your algorithm.

```
public class TrafficSignalOptimizer {
private static final int MAX_GREEN_TIME = 60;
private static final int MIN_GREEN_TIME = 30;
private static final int PEAK_HOUR_THRESHOLD = 50; for
peak hours
public int calculateGreenTime(TrafficData data) {
int greenTime;
if (data.getVehicleCount() > PEAK_HOUR_THRESHOLD) {
greenTime = Math.min(MAX_GREEN_TIME, MIN_GREEN_TIME +
(data.getVehicleCount() / 10)); // Simple formula for green time
} else {
greenTime = MIN_GREEN_TIME;
}
return greenTime;
```

```
}

public void adjustSignalTiming(TrafficData data) {

int greenTime = calculateGreenTime(data);

System.out.println("Adjusting green time to: " + greenTime + " seconds");

}

}
```

3. **Implementation:**
    - Implement a Java application that integrates with traffic sensors and controls traffic signals at selected intersections.
    - Ensure the application can adjust signal timings in real-time to respond to changing traffic patterns and optimize flow.

```
import java.util.Timer;

import java.util.TimerTask;

public class TrafficSignalControl {

private TrafficSignalOptimizer optimizer = new TrafficSignalOptimizer();

private Timer timer = new Timer();

public void startTrafficControl() {

timer.scheduleAtFixedRate(new TimerTask() {

public void run() {

TrafficData data = collectTrafficData();

optimizer.adjustSignalTiming(data);

}

}, 0, 5000);

}

private TrafficData collectTrafficData() {

return new TrafficData("Intersection1", System.currentTimeMillis(), (int)
```

```
(Math.random() * 100), 30 + Math.random() * 10, (int) (Math.random() * 20));

}

public static void main(String[] args) {

TrafficSignalControl control = new TrafficSignalControl();

control.startTrafficControl();

}

}
```

4. **Visualization and Reporting:**
    - Develop visualizations to monitor traffic conditions and signal timings in real-time.
    - Generate reports on traffic flow improvements, average wait times, and overall congestion reduction achieved.

```
package com.example.trafficsignals;

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;

public class HelloApplication 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);
        }
        }
```

5. **User Interaction:**
    - Design a user interface for traffic managers to monitor and manually adjust signal timings if needed.
    - Provide a dashboard for city officials to view performance metrics and historical data.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
```

```
public class TrafficManagerUI extends Application {
public void start(Stage stage) {
stage.setTitle("Traffic Manager Interface");
Label statusLabel = new Label("Current Signal Status: Normal");
Button overrideButton = new Button("Manual Override");
overrideButton.setOnAction(e -> {
statusLabel.setText("Signal Status: Manually Adjusted");
});
VBox vbox = new VBox(10, statusLabel, overrideButton);
Scene scene = new Scene(vbox, 300, 200);
stage.setScene(scene);
stage.show();
}
public static void main(String[] args) {
launch(args);
}
}
```