TRAFFIC CONGESTION MANAGEMENT SYSTEM

1.INTRODUCTION:

Traffic congestion is a ubiquitous issue in urban areas, impacting the efficiency of transportation systems and causing delays that affect daily life. In particular, the management of traffic flow at junctions is crucial for maintaining a smooth and safe movement of vehicles. The main focuses is on addressing the challenges of traffic congestion at a four-road junction. Within this intersection, our system incorporates four signals, each regulating the flow of traffic from different directions. The primary goal is to streamline traffic movement by controlling signal timings efficiently while also prioritizing emergency vehicles, such as ambulances, to ensure prompt and uninterrupted passage when necessary.

2.OBJECTIVE:

The main objective is to develop an intelligent traffic management system that optimizes signal control at a four-road junction.
Aims:

1. Minimize Congestion: Implement signal timings and coordination strategies to reduce congestion, enabling smoother vehicular movement within the junction.

2. Prioritize Emergency Vehicles: Create a specialized mechanism that identifies and responds to emergency vehicles, particularly ambulances, granting them immediate passage by manipulating signal timings to ensure their swift and safe traversal, regardless of the regular signal cycle.

3. Enhance Efficiency: Improve overall traffic flow by dynamically adjusting signal timings based on traffic volume and patterns, considering peak hours, and regulating the timing sequence for each road.

4. Ensure Safety: Promote a safe environment for both vehicles and pedestrians by synchronizing signal changes to prevent collisions and provide clear directions for traffic movement.

PROBLEM STATEMENT:

Traffic congestion at junctions is a persistent challenge that hampers the smooth operation of urban transportation systems. At a four-road junction, the

concurrent movement of vehicles from multiple directions often leads to gridlocks and delays, impacting the overall efficiency of the road network. The existing traffic signal systems often lack adaptability, leading to inefficient management of traffic during peak hours and emergencies.

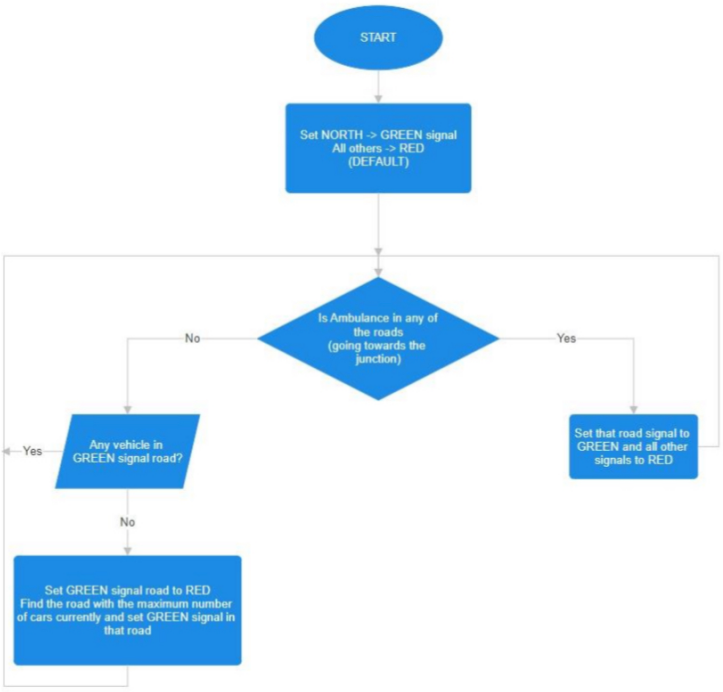
Moreover, the absence of a specialized mechanism to prioritize emergency vehicles like ambulances can result in critical delays that may jeopardize lives.The absence of a comprehensive and adaptive traffic management system that integrates both regular traffic flow control and emergency vehicle prioritization is the focal problem we aim to address. Developing a solution that not only optimizes traffic movement but also recognizes and expedites emergency vehicles through intelligent signal manipulation is crucial to mitigate congestion and ensure timely response during critical situations.

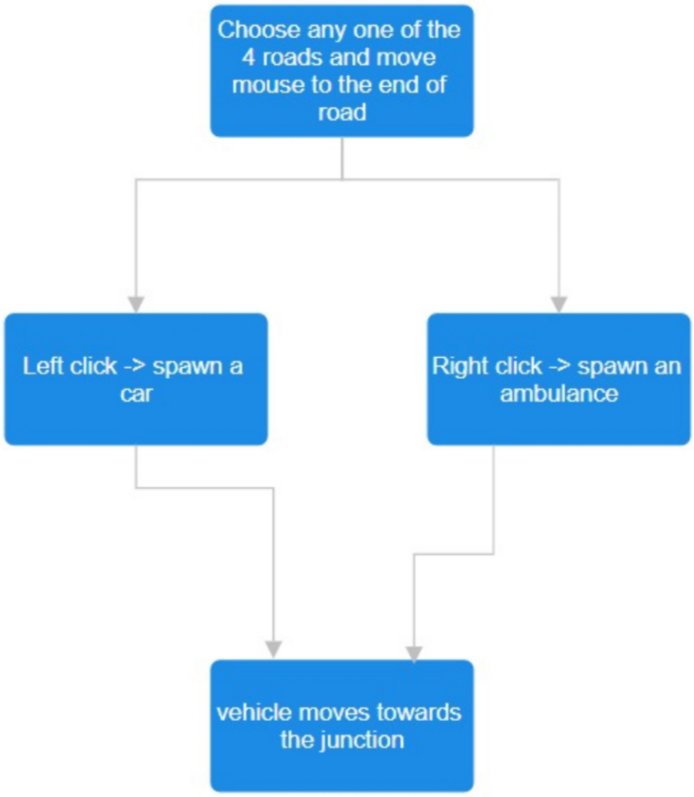
By leveraging innovative technology and adaptive algorithms, our project seeks to tackle these challenges by designing and implementing an intelligent traffic management system that efficiently regulates traffic flow while giving precedence to emergency vehicles, thereby enhancing overall traffic management and ensuring the swift and safe passage of vital emergency services.

REQUIREMENTS SOFTWARE SOFTWARE REQUIREMENTS:

1. Visual Studio
2. Python programming language

SYSTEM DESIGN FLOW DIAGRAMS AND EXPLANATION FLOWCHART:





EXPLANATION:

User Interaction for Vehicle and Ambulance Spawning:

- Vehicle Spawning: Users can spawn a regular vehicle on a road by LEFT clicking at the end of the desired road.

- Ambulance Spawning: Users can spawn an ambulance on a road by RIGHT clicking at the end of the desired road.

Vehicle Movement and Signal Control:
- Upon spawning, vehicles (both regular and ambulance) immediately start moving towards the junction.
- If a traffic signal for a particular road is RED, newly spawned cars align themselves at the end of that road and wait until the signal turns GREEN.
Signal Management Logic:
- The function `check_signal()` runs in a loop to monitor the traffic conditions and manage signal changes.

- Ambulance Priority: If an ambulance spawns on any road, the signal for that road immediately turns GREEN, prioritizing the ambulance's passage. All other road signals turn RED to clear the way.

- Regular Traffic Flow: If there is no ambulance and all cars in the current GREEN signal road have passed, the signal for that road turns RED.

- Determining Next Green Signal Road: The function identifies the road with the maximum number of cars and sets its signal to GREEN, ensuring continuous traffic movement.

Continuous Traffic Management:
- The system continues managing traffic signals based on ambulance priority and regular traffic flow until the program is exited.
- The process ensures efficient traffic flow by dynamically changing signal states to facilitate vehicle movement and prioritize emergency services.

IMPLEMENTATION

IMPLEMENTATION DETAILS - MODULES AND ITIMPLEMENTATION DETAILS

MODULES AND IMPLEMENTATION DETAILS:

1. Pygame Module
- Implementation: Utilized for creating the game window, handling events, rendering shapes, images, and sprites, managing game loops, and updating the screen.

2. Car Class
- Implementation:
- Represents individual cars within the simulation.
- Handles the movement of cars based on their spawn direction, turning behaviour, and interaction with signals.
- Manages the detection of other cars to avoid collisions.
- Implements methods for updating car movements, turning at the junction, and handling internal functions.

3. Signal Class
- Implementation:
- Represents traffic signals at each road in the junction.

- Manages the state of signals (stop or go) and updates their appearance accordingly.
- Allows user interaction to change the signal state.4. Functions (get_car_number, check_signal, add_cars)
- Implementation:
- `get_car_number()`: Identifies the road direction with the highest number of cars.
- `check_signal()`: Checks for the presence of cars on the road with the green signal and updates the signal accordingly.
- `add_cars()`: Adds cars to the simulation based on the mouse click position in different road directions.

4. Game Loop and Event Handling
- Implementation:
- Manages the main game loop for continuous rendering and updates.
- Handles user input events such as key presses for changing signal direction and mouse clicks to add cars.

5. Graphics and Interface
- Implementation:
- Renders the junction, middle lines, border lines, and the central rectangular area.
- Displays cars, traffic signals, and other visual elements using Pygame's drawing functions.
- Allows for user interaction to change signal directions and add cars to different roads.

6. Main Loop Execution
- Implementation:
- Executes the main loop that continually updates the screen, handles user events, and manages the simulation of cars and traffic signals until the user exits the window.

SCREENSHOTS: