# 

# 

# **School of Future Tech**

# 

# **Mini Project Report**

# on

## 

## **Traffic Light Simulation Using C++**

## **by**

### **GROUP NO 5**

### **Git hub link -:** [**https://github.com/sahil24302021/c-mini-project**](https://github.com/sahil24302021/c-mini-project)

### **Team Member -: 1. SAHIL KUMAR 2.MUKESH CHOUDHARY 3.SHIVAM SHA**

### 

### 

### **index**

1. Introduction to the Case Study.
2. Problem Statement / Case Background (Abstract).
3. Case Study Design.
4. Methods & Algorithms Technology Applied in the Case Study.
5. Case Study Implementation Details and Snapshots.
6. Case Study Results and Conclusion.
7. References.

### **1. Introduction to the Case Study**

Traffic control systems are critical infrastructure in modern urban environments, designed to manage the flow of vehicles and pedestrians to ensure safety and efficiency. As cities grow, understanding the logic behind these systems becomes essential for computer science engineers.

This case study focuses on the design and implementation of a **Traffic Light Simulation** using the **C++ programming language**. The project simulates the timing and visual logic of a real-world traffic signal, cycling through Red, Green, and Orange lights based on specific time intervals.

The simulation demonstrates how software interacts with system time and console output to create real-time, user-friendly visual applications. It highlights the use of Object-Oriented Programming (OOP) to model real-world devices.

### **2. Problem Statement / Case Background (Abstract)**

**Background** Real-world traffic lights operate on precise timers and state machines. Implementing this logic requires a program that can handle "states" (Red, Green, Orange) and "delays" (waiting for a specific duration) without freezing the entire system logic permanently.

**Abstract** This case study presents a C++ application that simulates a traffic control system. The objective is to create a continuous loop that:

1. Displays the current active signal (Red, Green, or Orange) using visual graphics.
2. Uses a countdown timer to show how long the current signal will remain active.
3. Utilizes ANSI color codes to render the output in the appropriate colors (Red text for Stop, Green for Go, etc.).
4. Refreshes the console screen dynamically to create an animation effect rather than a scrolling log.

The system is evaluated based on its ability to maintain accurate timing and transition smoothly between states.

### **3. Case Study Design**

The Traffic Light Simulation is designed as a modular Class-based system:

**Class Structure: TrafficSignal** The core of the project is the TrafficSignal class, which encapsulates all data and behaviors of the traffic light.

* **Attributes:** Stores the duration for each light phase (redDuration, greenDuration, orangeDuration).
* **Constructor:** Initializes the traffic light with custom timings.

**Operational Flow:**

1. **Initialization:** The user creates an instance of TrafficSignal with specific time values (e.g., 5 seconds for Red, 5 for Green, 2 for Orange).
2. **Phase Execution (runPhase):** The system enters a specific color phase.
3. **Drawing (drawGraphic):** The system clears the console and redraws the ASCII art of the traffic light, highlighting the active bulb (Red, Orange, or Green) and the countdown timer.
4. **Timing:** The system sleeps for 1 second (std::this\_thread::sleep\_for), decrements the timer, and redraws the graphic until the phase is over.
5. **Looping:** The start() method runs an infinite loop, cycling through the sequence RED -> GREEN -> ORANGE.

### 4. Methods & Algorithms Technology Applied

**Key Methods & Algorithms:**

* **Object-Oriented Programming (OOP):** usage of Classes and Objects to encapsulate the traffic signal logic.
* **Console Manipulation:**
  + system("clear"): Used to clear the terminal screen before every frame update to create a smooth animation effect.
  + **ANSI Escape Codes:** Used to print colored text (e.g., \033[31m for Red) to the terminal, enhancing the user interface.
* **Time & Threading:**
  + std::this\_thread::sleep\_for: Used to pause the program execution for exactly 1 second between updates.
  + std::chrono::seconds: Used to define the time duration in a standard, cross-platform way.

**Technology Stack:**

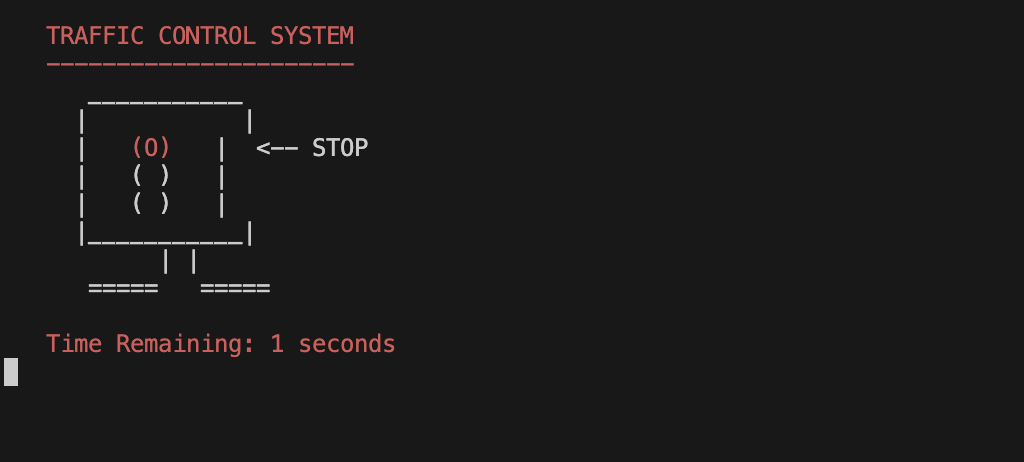
* **Language:** C++ (Standard C++11 or later).
* **Libraries:**
  + <iostream>: For Input/Output.
  + <thread> & <chrono>: For handling time delays and threading.
  + <cstdlib>: For system commands.
  + <string>: For string manipulation.
* **Environment:** MacOS Terminal / VS Code (Compatible with any standard C++ compiler).

### **5. Case Study Implementation Details and Snapshots**

The implementation works by defining a TrafficSignal class. The visual representation is constructed using ASCII characters.

**Code Structure Overview:**

* **Class Definition:** Defines the structure of the signal.
* **drawGraphic Function:** This function handles the UI. It checks the activeLight string and decides which "bulb" to colorize in the ASCII art.
  + *Logic:* If activeLight == "RED", the top bulb (O) is printed in Red; others remain empty ( ).
* **runPhase Function:** A loop that counts down from duration to 0. In every iteration, it calls drawGraphic and then sleeps for 1 second.
* **main Function:** Instantiates the object TrafficSignal mySignal(5, 5, 2); and calls .start().





### **6. Case Study Results and Conclusion**

**Findings**

* The application successfully simulates a continuous traffic cycle.
* The use of system("clear") provides a seamless user experience, making the terminal look like a static graphical interface rather than a scrolling text log.
* The modular design allows for easy adjustment of signal timings (e.g., changing Red light from 5s to 10s requires changing only one number in main).

**Conclusion** This project successfully demonstrates the capability of C++ to handle real-time simulation and system-level console manipulation. It bridges the gap between basic logic programming and interactive application development. The resulting system is a lightweight, efficient simulation of real-world traffic control logic.

### 7. References

1. Bjarne Stroustrup, *The C++ Programming Language*.
2. cplusplus.com - Documentation on <thread> and <chrono>.
3. GeeksforGeeks - Tutorials on C++ Classes and Objects.
4. Standard C++ Foundation - Documentation on ANSI Color Codes for Terminal Output.