



**NATIONAL ENGINEERING COLLEGE, K.R.NAGAR, KOVILPATTI - 628 503**

(An Autonomous Institution – Affiliated to Anna University, Chennai)

**Department of Computer Science and Engineering**

**23CS45C – DATABASE MANAGEMENT SYSTEM**

**MICRO PROJECT**

Report on

**TRAFFIC MANAGEMENT SYSTEM**

Team Members:

R.Jerina Gnana Puishpa-2312021(B3)  
D.Princy Christina-2312079(B2)  
M.Muniyammal-2312407(B1)

Course Instructor

Dr. G.SIVAKAMASUNDARI  
Associate. Professor  
Dept of CSE, NEC

# INTRODUCTION

The Traffic Management System is a smart solution designed to address congestion, rule violations, and emergency delays in modern cities. Built using Object-Oriented Programming (OOP) principles like inheritance and polymorphism, the system includes modules for traffic signal control, vehicle tracking, rule enforcement, emergency handling, and real-time communication.

It uses a Database Management System (DBMS) to manage data like vehicle entries, signal timings, and violation reports, with controllers coordinating key actions. Features such as file logging, exception handling, and utility tools (e.g., time conversion and ID formatting) enhance system reliability and scalability. This project supports smart city goals by ensuring efficient, safe, and real-time traffic flow management.

## **OBJECTIVE**

To transform conventional traffic control methods through a comprehensive digital infrastructure designed to meet the demands of modern urban transportation. This Java-based solution leverages core Object-Oriented Programming (OOP) concepts and robust database management to create a modular, scalable platform capable of monitoring and regulating city traffic in real time. The system addresses key challenges such as traffic congestion, rule violations, emergency vehicle delays, and inefficient signal coordination by integrating advanced modules for signal control, vehicle tracking, automated rule enforcement, and emergency response prioritization. By digitizing traffic data collection and enforcement mechanisms, the system reduces the reliance on manual oversight and significantly minimizes human error. It features a secure, user-centric interface for traffic controllers, administrators, and law enforcement to efficiently access and manage critical traffic information. With real-time updates and display board communication, the system ensures timely dissemination of traffic advisories and rerouting instructions.

The backend leverages structured repositories to support seamless CRUD operations on traffic signals, vehicle records, and violation logs, while utility services enhance functionality through tools like data validation, timestamp conversion, and ID formatting. In addition, comprehensive logging and exception handling mechanisms ensure the system's reliability under real-world conditions.

Beyond day-to-day traffic operations, the system empowers city planners and enforcement agencies with actionable insights through customizable reports and analytics. These features support data-driven decisions to optimize traffic flow, plan infrastructure upgrades, and enforce regulations effectively.

## **ER DIAGRAM**

An Entity-Relationship (ER) diagram is a visual tool used to design the structure of a database by identifying the entities involved and the relationships between them. The ER diagram for the Traffic Management System defines the core entities, attributes, and their interrelations to simulate and manage city traffic efficiently.

**1.Traffic Signals:**This entity represents signal status of the junctions.

ID: A unique ID to identify the traffic Signals.

Location: location of the signal.

Status: Represents in which state the signal is.

Installation Date: Date in which it is installed.

**2.Traffic Cameras:**This entity stores information about traffic flow at various locations.

ID: A unique ID to identify the traffic cameras.

Location: location in which the camera is present.

Status: Represents in which state the camera is.

Resolution: Specifies the resolution of the camera.

**3.Traffic Incidents:**This entity records incidents captured by surveillance or enforcement.

ID: Unique identifier for each monitoring instance.

Location: The location of the incident.

Type: Type of the incident.

Severity: It specifies the level of the cause.

Reported Time: The current time when an incident occurred is reported.

### **Relationships:**

Passes: Represents the movement of vehicles through monitored traffic intersections, captured by cameras and influenced by traffic signals.

Violates: Identifies rule-breaking incidents such as speeding, signal violations, or unauthorized maneuvers, detected via traffic enforcement mechanisms.

Detected\_at: Specifies the exact traffic location where incidents or violations were recorded, linking signals, cameras, and enforcement data.

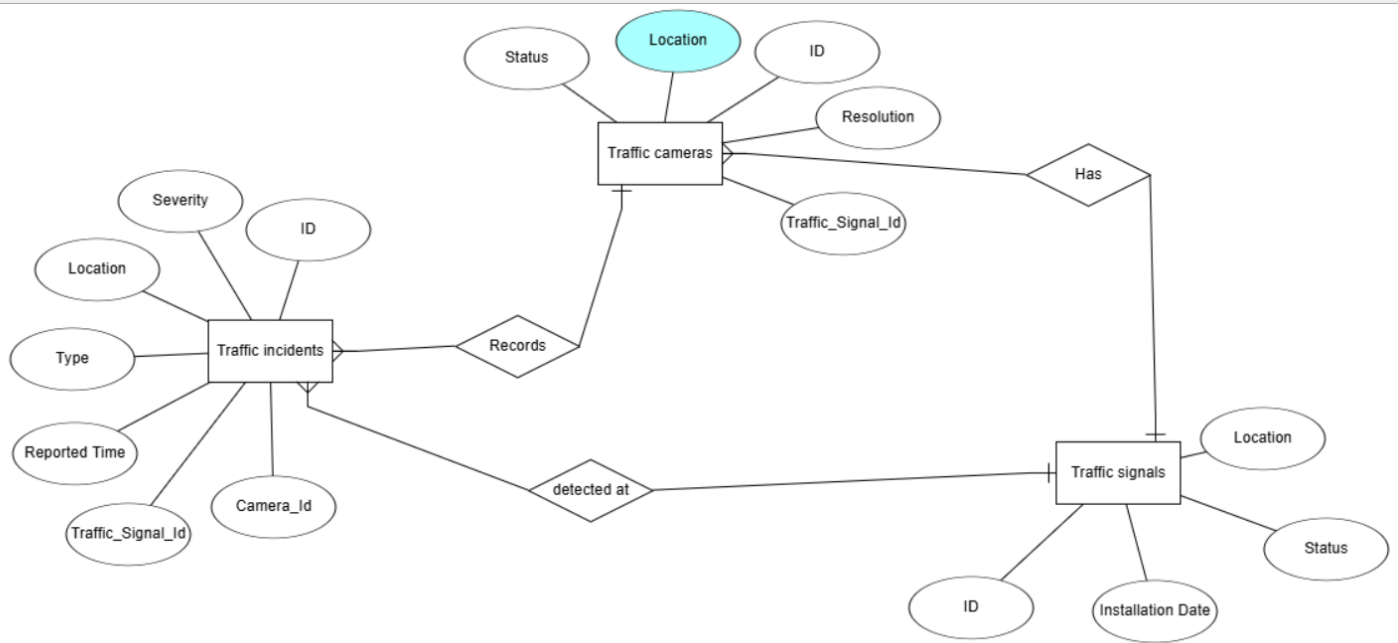


Fig 1: Entity Relationship Diagram for Traffic Management System.

## SCHEMA DIAGRAM

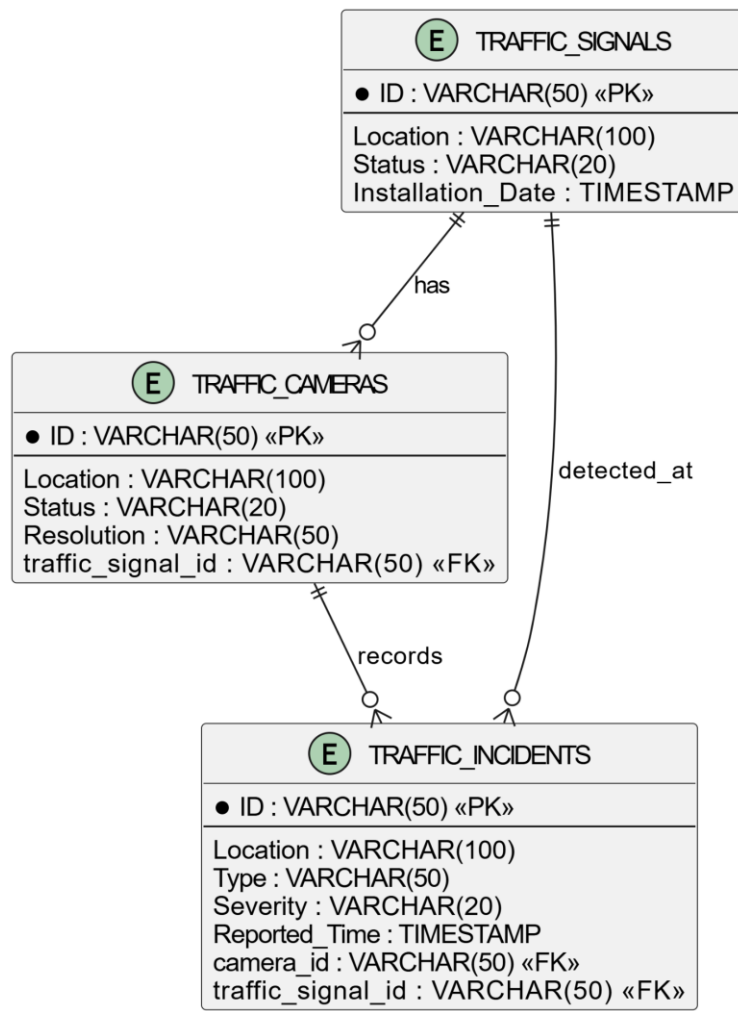


Fig 2: Schema Diagram

This schema directly implements the ER diagram you provided earlier and includes the attributes you've specified. The Sales table serves as the junction table that implements the many-to-many relationship between MonitoringEnforcement, Vehicle and Traffic.

### **1. Traffic Signals Table:**

**Primary Key:** signal\_id

**Attributes:**

- location (VARCHAR)
- status (VARCHAR)
- installation\_date (DATETIME)

### **2. Traffic Cameras Table:**

**Primary Key:** camera\_id

**Attributes:**

- location (VARCHAR)
- status (VARCHAR)
- resolution (VARCHAR)
- signal\_id (FK)

### **3. Traffic Incidents Table:**

**Primary Key:** incident\_id

**Foreign Keys:**

- camera\_id
- signal\_id

**Other Attributes:**

- location (VARCHAR)
- type (VARCHAR)
- severity (VARCHAR)
- reported\_time (DATETIME)

### Relationships Explained:

One traffic signal can be monitored by multiple cameras

Each camera can record multiple traffic incidents over time

Incidents are linked to both signals (malfunction-related issues) and cameras (visual proof of violations)

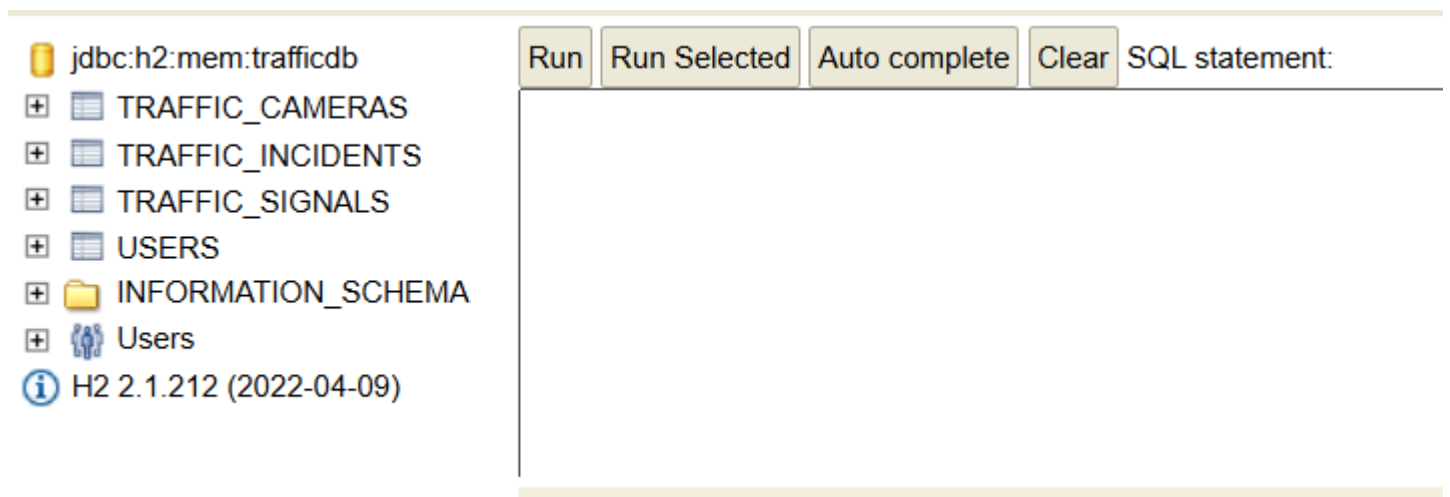
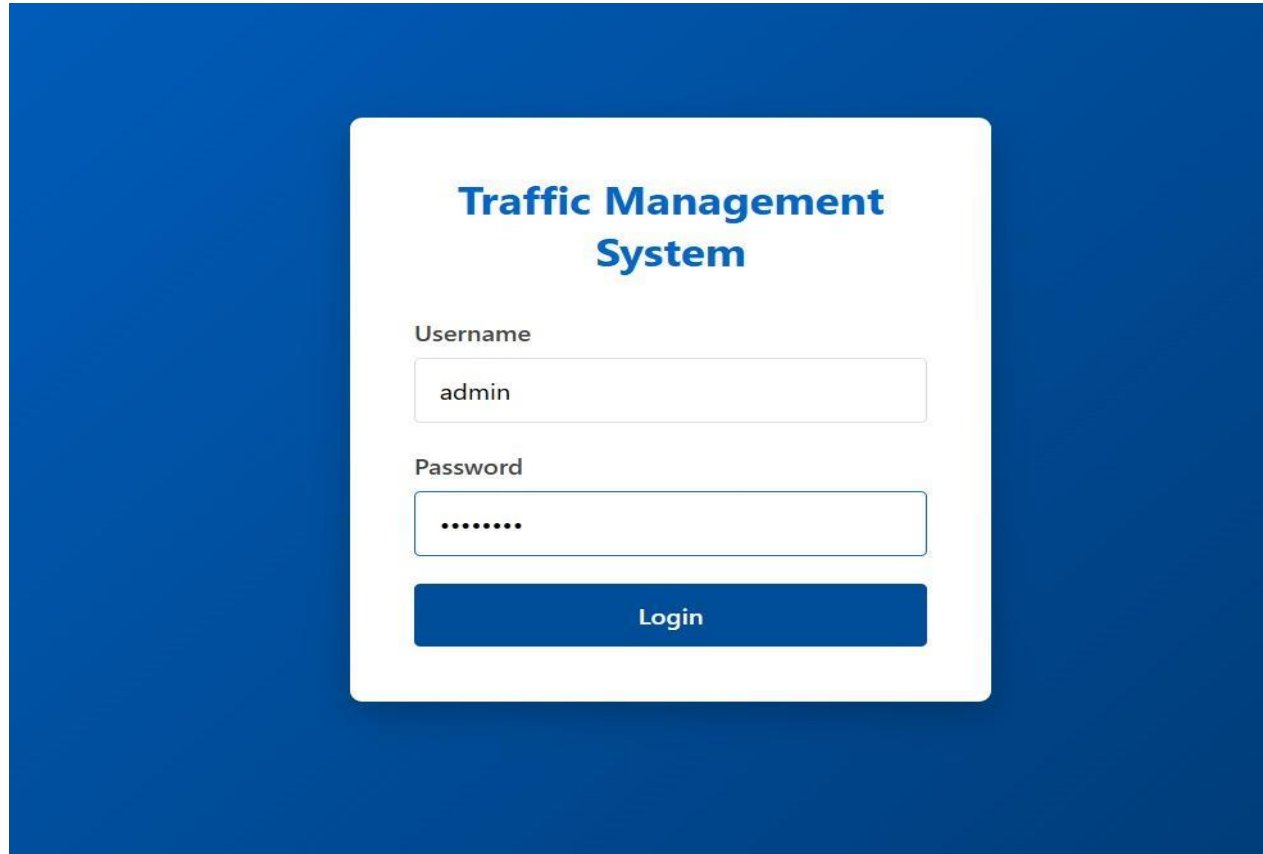


Fig 3: Database Creation in H2 Console



## Login Module (UI Design):



The image shows a login interface for a 'Traffic Management System'. It features a dark blue background with a white rectangular login box in the center. The box contains the title 'Traffic Management System' in bold blue text. Below the title are two input fields: 'Username' with the text 'admin' and 'Password' with masked characters '.....'. A blue 'Login' button is positioned at the bottom of the form.

Fig 4: Login Module

This is a web-based Admin Management interface, running on localhost:8084/h2-console. It is part of an administrative module designed to manage admin accounts within a system.

## **Admin Module - Traffic Management System**

### **Functionalities:**

#### **View Traffic Signals, Cameras, and Incidents:**

Displays all registered traffic signals, cameras, and incidents stored in the system.  
Each entry includes ID, Location, Status, Installation/Reported Date, and Actions.

#### **Add New Traffic Signal, Camera, or Incident:**

Separate input fields for entering location, status, type, and additional attributes.  
A green "Add" button submits the data to create a new record in the system.

#### **Delete Traffic Data:**

The Delete button allows the admin to remove unwanted or outdated records from the database.  
Data deletion is instant and reflected across the system.

#### **Update Existing Records (Optional):**

Admins can modify traffic signal status, change camera details, or update incident severity.  
A dynamic input form enables instant updates in the H2 database.

**Traffic Cameras**

3

ents

**Add Traffic Camera** ×

Location

Bellingham South

Status

Online ▼

Resolution

HD (720p) ▼

Cancel Save

MAINTENANCE HD

Fig 5:Traffic Cameras

## **Features:**

### **Simple and Clean UI Layout:**

- Uses table format to display traffic signals, cameras, and incident records clearly.
- Easy navigation between different sections for better accessibility.

### **CRUD Operations for Traffic Management:**

- Admins can Create, Read, Update, and Delete records for Traffic Signals, Cameras, and Incidents.
- All operations are connected to the H2 database for seamless data handling.

### **Instant Update Interface with RESTful API:**

- Uses AJAX requests to fetch, insert, and delete data dynamically.
- Backend powered by Spring Boot and H2 Database, ensuring fast performance.

## **SOURCE CODE**

### **Application.properties:**

```
server.port=8084
spring.datasource.url=jdbc:h2:mem:trafficdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
```

```
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

```
logging.level.org.springframework=INFO
logging.level.com.traffic.management=DEBUG
```

```
spring.security.user.password=none
```

### **AuthController.Java:**

```
package com.traffic.management.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.traffic.management.dto.LoginRequest;
import com.traffic.management.service.UserService;
```

```

@RestController
@RequestMapping("/api/auth")
public class AuthController {

    @Autowired
    private UserService userService;

    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody LoginRequest loginRequest) {
        if (userService.authenticate(loginRequest.getUsername(), loginRequest.getPassword())) {
            return ResponseEntity.ok().build();
        } else {
            return ResponseEntity.status(401).build();
        }
    }
}

```

#### **UserController.java:**

```

package com.traffic.management.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.traffic.management.entity.User;
import com.traffic.management.service.UserService;

@RestController
@RequestMapping("/api/users")
public class UserController {

    @Autowired

```

```

private UserService userService;

@GetMapping
public List<User> getAllUsers() {
    return userService.getAllUsers();
}

@GetMapping("/{id}")
public ResponseEntity<User> getUserById(@PathVariable Long id) {
    return userService.getUserById(id)
        .map(ResponseEntity::ok)
        .orElse(ResponseEntity.notFound().build());
}

@PutMapping("/{id}")
public ResponseEntity<User> updateUser(@PathVariable Long id, @RequestBody User user) {
    // Only allow updating the admin user
    if (user.getUsername() != null && !user.getUsername().equals("admin")) {
        return ResponseEntity.badRequest().body(null);
    }

    return userService.getUserById(id)
        .map(existingUser -> {
            user.setId(id);
            return ResponseEntity.ok(userService.saveUser(user));
        })
        .orElse(ResponseEntity.notFound().build());
}
}

```

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **I. VISION**

To produce globally competent, innovative and socially responsible computing professionals

### **II. MISSION**

- To provide world-class teaching-learning and research facilities
- To stimulate students' logical thinking, creativity, and communication skills
- To cultivate awareness about emerging trends through self-initiative
- To instill a sense of societal and ethical responsibilities
- To collaborate with industries and government organizations

### **III. PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

PEO 1: Achieve their professional career in industry/academia by applying the acquired knowledge of computer science and engineering.

PEO 2: Engage in life-long learning and enhance their capabilities by embracing cutting edge technical advancements.

PEO 3: Excel in collaboration with interdisciplinary teams and diverse stake holders for persevering successful start-ups.

### **IV. PROGRAM SPECIFIC OUTCOMES (PSOs)**

PSO1: Build domain specific expertise by showcasing deliverables in the field of Application development, Business Intelligence, Computational Intelligence and Cyber Security

PSO2: Build knowledge base for students to solve complex technical problems through participation in global contests and hackathons.



**PO Statement:**

PO 1: Apply knowledge of mathematics, natural science, computing and engineering fundamentals, with specializations in computational intelligence, web development, business intelligence, and cyber security to develop solutions to complex engineering problems

PO 2: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences with holistic considerations for sustainable development

PO 3: Design creative solutions for complex engineering problems and design systems, components or processes to meet identified needs with appropriate consideration for public health and safety, whole-life cost, net zero carbon as well as resource, cultural, societal, and environmental considerations as required

PO 4: Conduct investigations of complex engineering problems using research methods including research-based knowledge, design of experiments, analysis and interpretation of data, and synthesis of information to provide valid conclusions

PO 5: Create, select and apply, and recognize limitations of appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling, to complex engineering problems

PO 6: When solving complex engineering problems, analyze and evaluate sustainable development impacts to: society, the economy, sustainability, health and safety, legal frameworks, and the environment

PO 7: Apply ethical principles and commit to professional ethics and norms of engineering practice and adhere to relevant national and international laws. Demonstrate an understanding of the need for diversity and inclusion

PO 8: Function effectively as an individual, and as a member or leader in diverse and inclusive teams and in multi-disciplinary, face-to-face, remote and distributed settings

PO 9: Communicate effectively and inclusively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, taking into account cultural, language, and learning differences

PO10: Apply knowledge and understanding of engineering management principles and economic

decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments

PO11: Recognize the need for, and have the preparation and ability for i) independent and lifelong learning  
ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change

**Possible PO's Mapped with Micro Project:**

<b>TRAFFIC MANAGEMENT SYSTEM</b>						
<b>PO1</b>	<b>PO2</b>	<b>PO3</b>	<b>PO5</b>	<b>PO8</b>	<b>PO9</b>	<b>PO11</b>