Real-Time Traffic Management System

A PROJECT REPORT

Submitted by

HARI PRIYA E (2303811710422050)

in partial fulfillment of requirements for the award of the course

CGB1221-DATABASE MANAGEMENT SYSTEMS

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE- 2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"Real-Time Traffic Management System"** is the bonafide work of **HARI PRIYA E (2303811710422050)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Mrs.A.Delphin Carolina Rani,M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621 112.

SIGNATURE

Ms. S. Uma Mageshwari, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621 112.

Submitted for the viva-voce examination held on 02.06.2025 .

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I declare that the project report on **"Real-Time Traffic Management System"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1221 – DATABASE MANAGEMENT SYSTEMS.**

.

Signature

HARI PRIYA E

Place: Samayapuram

Date: 02.06.2025

ACKNOWLEDGEMENT

It is with great pride that I express my gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of my study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to my project and offering adequate duration in completing my project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express my deep expression and sincere gratitude to my project supervisor **Ms. S. UMA MAGESHWARI, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render my sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express my special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and

research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively 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, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Real-Time Traffic Management System project integrates a MySQL database with a Python-based graphical user interface (GUI) to provide an efficient, user-friendly platform for monitoring and managing traffic data across various locations. This system captures essential traffic metrics such as vehicle counts associated with specific locations and timestamps, storing them persistently in a relational database. The intuitive Tkinter-powered interface enables users to easily input new traffic data, search and filter records based on location, and perform data deletion if necessary. Beyond data management, the project features dynamic data visualization through Matplotlib, rendering insightful bar graphs that aggregate vehicle counts by location to assist in traffic pattern analysis. Additionally, the system continually updates and displays the most recent timestamp of data entry, ensuring users are always aware of the currency of the information presented. Overall, this application combines database management, real-time data handling, and interactive visualization to support traffic monitoring efforts, making it a comprehensive tool for urban planners, traffic analysts, and relevant stakeholders seeking to optimize traffic flow and infrastructure planning.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD DATABASE MANAGEMENT SYSTEM APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Real-Time Traffic Management System project integrates a MySQL database with a Python-based graphical user interface (GUI) to provide an efficient, user-friendly platform for monitoring. This system captures essential traffic metrics such as vehicle counts associated with specific locations and timestamps, storing them persistently in a relational database. The intuitive Tkinter-powered interface enables users to easily input new traffic data, search and filter records based on location, and perform data deletion if necessary. Beyond data management, the project features dynamic data visualization through Matplotlib, rendering insightful bar graphs that aggregate vehicle counts by location to assist in traffic pattern analysis. Additionally, the system continually updates and displays the most recent timestamp of data entry, ensuring users are always aware of the currency of the information presented. Overall, this application combines database management, real-time data handling, making it a comprehensive tool for urban planners, traffic analysts, and relevant stakeholders seeking to optimize traffic .	**PO1 - 3** **PO2 - 3** **PO3 - 3** **PO4 - 3** **PO5 - 3** **PO6 - 3** **PO7 - 3** **PO8 - 3** **PO9 - 3** **PO10 - 3** **PO11 - 3** **PO12 - 3**	**PSO1 - 3** **PSO2 - 3** **PSO3 - 3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

SQL	-	Structured Query Language
GUI	-	Graphical User Interface
MySQL	-	My Structured Query Language (database)
CRUD	-	Create, Read, Update, Delete
ID	-	Identifier
Pymysql	-	Python MySQL Library (Py + MySQL)

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of the Real-Time Traffic Management System project is to develop a comprehensive and user-friendly application that facilitates the efficient collection, management, and analysis of traffic data in real-time. This system aims to provide urban planners, traffic analysts, and transportation authorities with a robust tool to monitor vehicle counts across various locations, enabling them to make informed decisions regarding traffic flow and infrastructure development. Key objectives include the seamless integration of a MySQL database for reliable data storage and retrieval, the implementation of an intuitive graphical user interface using Tkinter to enhance user experience, and the provision of data visualization capabilities through Matplotlib to present traffic trends and patterns effectively. Additionally, the project seeks to empower users with functionalities such as data insertion, search, deletion, and real-time updates, ensuring that the information remains current and actionable. By achieving these objectives, the Real-Time Traffic Management System aspires to contribute to smarter urban mobility solutions, ultimately enhancing traffic management strategies and improving overall transportation efficiency in urban environments.

1.2 Overview

The Real-Time Traffic Management System is an innovative application designed to streamline the process of collecting, managing, and analyzing traffic data in urban environments. At its core, the system utilizes a MySQL database to store critical information, including vehicle counts, locations, and timestamps, ensuring that data is both persistent and easily accessible. The application features a user-friendly graphical interface built with Tkinter, allowing users to input new traffic data effortlessly, search

for specific records, and delete entries as needed. One of the standout features of the system is its ability to visualize traffic data through dynamic bar graphs generated by Matplotlib, which aggregate vehicle counts by location, providing users with valuable insights into traffic patterns and trends. The interface is designed to be intuitive, with clearly labeled input fields and buttons that facilitate quick data entry and retrieval. Additionally, the system includes a real-time update mechanism that displays the most recent timestamp of data entries, ensuring users are always informed of the latest information. Overall, the Real-Time Traffic Management System serves as a vital tool for urban planners and traffic management authorities, enabling them to make data-driven decisions that enhance traffic flow, improve safety, and optimize infrastructure development in bustling urban landscapes.

1.3 SQL and Database Concepts

The Real-Time Traffic Management System is built upon fundamental SQL and database concepts that are essential for effective data management and retrieval. At the heart of the system lies a relational database, specifically MySQL, which organizes data into structured tables that facilitate efficient querying and data manipulation. The primary table, `traffic_data`, is designed to store critical information about traffic conditions, including unique identifiers (IDs), location names, vehicle counts, and timestamps. This structure exemplifies the principles of normalization, where data is organized to minimize redundancy and ensure data integrity.

SQL (Structured Query Language) serves as the primary means of interacting with the database, allowing the application to perform various operations such as data insertion, retrieval, updating, and deletion. The use of SQL commands like `INSERT`, `SELECT`, `DELETE`, and `UPDATE` enables the application to maintain accurate and up-to-date traffic records. For instance, the `INSERT` command is utilized to add new traffic data entries, while the `SELECT` command retrieves existing records for display or analysis. The application also employs aggregate functions, such as `SUM`,

to calculate total vehicle counts by location, showcasing the power of SQL in performing complex data analysis.

Furthermore, the database design incorporates indexing and primary key constraints to enhance query performance and ensure the uniqueness of records. The `id` column serves as a primary key, providing a unique identifier for each entry, which is crucial for data integrity and efficient data retrieval. The timestamp field, automatically populated with the current date and time upon data insertion, allows for tracking changes and updates in real-time, further enriching the data's contextual relevance. Overall, the SQL and database concepts underpinning the Real-Time Traffic Management System not only facilitate robust data management but also empower users to derive meaningful insights from traffic data, ultimately contributing to more informed decision-making in urban traffic management.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Real-Time Traffic Management System aims to design and implement a comprehensive, scalable, and user-centric application that facilitates real-time monitoring, storage, visualization, and analysis of traffic data to assist in efficient urban traffic management and long-term infrastructure planning. The core focus lies in developing an integrated system that not only captures live traffic information but also ensures secure, consistent, and accurate data processing, thereby enabling authorities and stakeholders to make informed, data-driven decisions. At the heart of this system is the MySQL relational database, chosen for its robustness, scalability, and reliability. It is structured to store detailed traffic data including unique identifiers, geographical locations, vehicle counts, and corresponding timestamps. The database schema is normalized and optimized using primary keys, indexes, and constraints to ensure high performance and data integrity. CRUD (Create, Read, Update, Delete) operations are supported with appropriate validation and error-handling mechanisms, allowing for seamless insertion and maintenance of traffic records.

Complementing the backend is a Python-based graphical user interface (GUI) developed using the Tkinter library. This interface serves as the user's primary point of interaction, offering an intuitive environment for entering new data, searching for location-specific information, deleting outdated or erroneous entries, and viewing traffic updates in real-time. To prevent inconsistencies and ensure quality data, the GUI incorporates real-time input validation, dynamic refresh capabilities, and logical navigation across all system components.A notable feature of the proposed system is its data visualization module, implemented using the Matplotlib library. This module transforms raw traffic data into dynamic and informative bar graphs that visually represent traffic volume across various monitored locations. These visual insights

make it easier to identify congestion zones, evaluate traffic density patterns, and prioritize areas requiring intervention.

Furthermore, the proposed architecture emphasizes modularity and scalability. It is designed to support future extensions such as real-time alert systems for traffic spikes, integration with GPS or IoT sensors for automatic data feeding, and advanced analytics using machine learning models for predictive traffic management. Real-time synchronization, combined with logging and update tracking, ensures transparency and auditability of all operations.In addition to technical enhancements, the project incorporates essential usability and performance considerations. The system ensures fast query execution, minimal response latency, and high user engagement through a responsive and aesthetically pleasing interface. Security measures such as restricted access control, sanitized inputs, and secure connections with the database have also been planned to safeguard against data breaches and unauthorized manipulations.

In summary, the proposed work delivers a multi-functional, real-time traffic management system that blends robust backend operations with a user-friendly frontend. By integrating real-time data handling, analytical tools, and an efficient UI, this system not only addresses current urban traffic challenges but also provides a solid foundation for advanced smart city applications in the future.

2.2 Block Diagram

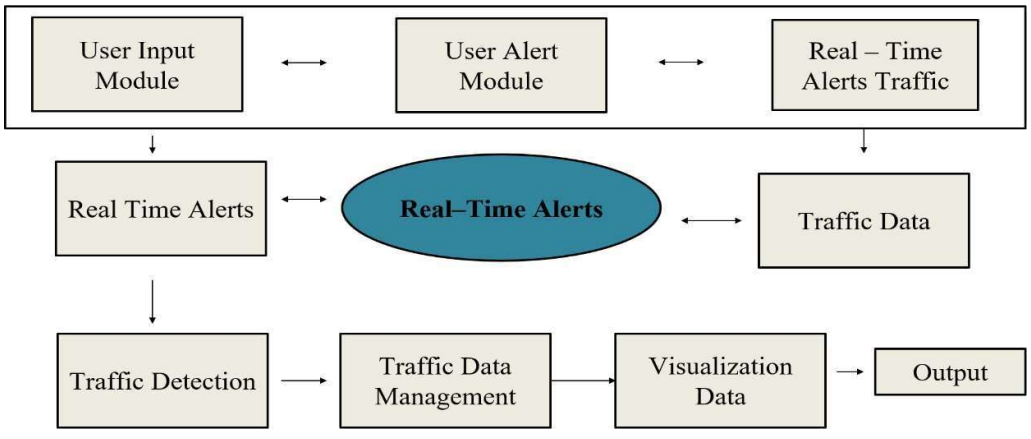


Fig 2.1 Block Diagram

CHAPTER 3

MODULE DESCRIPTION

3.1 User Input Module

The User Input Module serves as the primary interface through which users interact with the traffic management system. Built using Tkinter, this module provides intuitive input fields for entering the location and vehicle count data. It incorporates input validation to ensure that vehicle counts are numeric and meaningful, minimizing data entry errors. This module also includes buttons for submitting new traffic data to the database, searching for existing records by location, deleting entries by record ID, and refreshing the visible data table. These functionalities empower users to efficiently manage traffic data while maintaining accuracy and ease of use in everyday operations. The design of this module ensures that traffic monitoring personnel or analysts can operate the system with minimal training. Its intuitive layout and immediate feedback mechanisms contribute significantly to the system's overall usability and reliability.

3.2 Traffic Visualization Module

The Traffic Visualization Module transforms raw numerical traffic data into meaningful and interpretable visuals. Powered by the Matplotlib library, this module retrieves vehicle count data from the MySQL database, groups it by location, and renders interactive bar charts that provide a comparative analysis of traffic volumes across different areas. The use of visual representation makes it easier for users to identify traffic patterns, hotspots, and potential bottlenecks without analyzing tabular data manually.

The module is built to dynamically reflect real-time data updates, ensuring that the visualized trends are always current. Users can invoke the visualization function using the "Show Graph" button, which triggers the creation of a new graph window. The graphs are customized with titles, axis labels, and color-coded bars for clarity and aesthetic appeal. Future enhancements to this module could include pie charts, line

graphs, and time-series plots to enable even deeper analytical insights. This module is crucial for presenting complex datasets in a user-friendly manner, supporting better planning and decision-making.

3.3 Real-Time Alerts Module

While the provided code does not currently include an explicit real-time alerts feature, the foundation exists for implementing such functionality by leveraging the timestamp data and dynamic UI updates. For example, this module could be extended to notify users when significant changes occur in traffic patterns, such as sudden spikes in vehicle counts at certain locations, via popup messages or indicator labels. By monitoring the latest traffic data entries and comparing them against predefined thresholds, the system can effectively alert traffic management personnel to potential congestion or incidents, facilitating timely responses.

Such alerts can significantly enhance responsiveness, allowing traffic authorities to implement timely interventions such as rerouting traffic, adjusting signals, or dispatching field officers. With further development, this module could also integrate predictive analytics, using historical data and machine learning algorithms to forecast congestion before it happens.

3.4 Data Storage and Retrieval Module

The Data Storage and Retrieval Module forms the backbone of the system's data integrity and availability. It connects securely to the MySQL database and manages all CRUD (Create, Read, Update, Delete) operations on the traffic data table. This module ensures that new traffic records are inserted correctly with accurate timestamps, retrieves requested data for display or analysis, enables deletion of obsolete or incorrect data entries, and efficiently fetches data for the visualization and searching functionalities. Through structured SQL queries and connection management, it facilitates robust and reliable access to live traffic data with consistent performance.

This module is responsible for handling all core interactions with the database, including inserting validated traffic data along with automatically generated timestamps to ensure chronological accuracy. It efficiently retrieves records for display within the graphical user interface (GUI) and supports data visualization functions. Additionally, it allows users to delete specific entries by referencing valid record IDs and facilitates real-time data refresh and reload operations to ensure that the most current information is always visible. To enhance performance, the database is indexed using unique identifiers and timestamps, which helps speed up query execution and data access. Comprehensive error handling mechanisms are embedded within the module to detect and manage input or system-level errors, providing users with informative feedback in case of failure. Looking ahead, this module could be enhanced to include advanced features such as automated data backups, restoration functions, and user access controls, enabling secure operations in multi-user environments.

3.5 Data Analysis and Reporting Module

The Data Analysis and Reporting Module enhances the intelligence of the traffic management system by converting stored data into meaningful and actionable insights. It leverages SQL aggregation functions such as SUM, GROUP BY, and ORDER BY, along with Python-based processing, to produce comprehensive summaries, detect trends, and compile structured reports. This module empowers users to view total traffic volumes for each location, sort data to pinpoint the most congested areas, and apply filters to examine records based on specific timeframes, location names, or traffic thresholds. Serving as a critical tool for strategic decision-making, it provides stakeholders with a broader and more informed understanding of traffic behavior across different regions. The system supports the generation of printable reports, and future enhancements may include the capability to export data in formats like CSV or PDF, schedule automatic report generation, and integrate predictive analytics tools. These improvements would extend the module’s functionality beyond daily operations, enabling it to play a key role in long-term planning and policy development.

CHAPTER 4

CONCLUSION AND FUTURE ENHANCEMENT

4.1 CONCLUSION

The Real-Time Traffic Management System effectively demonstrates the integration of database management, graphical user interface design, and data visualization to provide a comprehensive solution for monitoring and analyzing traffic conditions. By leveraging a MySQL database for reliable data storage, combined with a user-friendly Tkinter interface, the system enables users to effortlessly input, search, delete, and review traffic data in real time. The inclusion of dynamic graphical representations through Matplotlib further enhances the analytical capabilities, allowing stakeholders to visualize traffic patterns and make data-driven decisions for urban traffic optimization. While the current implementation covers essential functionalities such as data management and visualization, it lays a strong foundation for future enhancements like real-time alerts and advanced reporting features. Overall, this project serves as a practical and scalable tool that can assist traffic authorities and urban planners in improving traffic flow, reducing congestion, and ultimately creating safer and more efficient transportation networks.

4.2 FUTURE SCOPE

The Real-Time Traffic Management System presents numerous opportunities for future enhancements that can significantly expand its functionality and effectiveness. One potential enhancement is the implementation of a Real-Time Alerts Module, which would notify users of sudden changes in traffic patterns, such as unexpected spikes in vehicle counts or potential congestion points. This could be achieved through the integration of threshold-based alerts that trigger notifications via email or SMS, allowing traffic management personnel to respond promptly to emerging issues. Additionally, incorporating machine learning algorithms could enable predictive

analytics, where the system forecasts traffic trends based on historical data, helping planners anticipate peak traffic times and adjust infrastructure accordingly.

Another area for enhancement is the expansion of data visualization capabilities. By integrating more advanced visualization tools, such as interactive maps that display real-time traffic conditions, users could gain deeper insights into traffic flow across different regions. This could be complemented by the addition of a mobile application, allowing users to access traffic data and receive alerts on-the-go, thereby increasing the system's accessibility and usability.

Furthermore, the system could benefit from enhanced reporting features that allow users to generate comprehensive traffic reports over specified periods, including detailed statistics and visual summaries. This would aid in long-term planning and policy- making by providing stakeholders with the necessary data to make informed decisions. Lastly, integrating external data sources, such as weather conditions or public transportation schedules, could provide a more holistic view of factors affecting traffic, enabling more effective management strategies. By pursuing these enhancements, the Real-Time Traffic Management System can evolve into a more robust tool that not only addresses current traffic challenges but also adapts to future urban mobility needs.

APPENDIX A (Project Source Code)

1. MYSQL CODE

```
CREATE DATABASE IF NOT EXISTS traffic_db;
USE traffic_db;
CREATE TABLE IF NOT EXISTS traffic_data (
    id INT AUTO_INCREMENT PRIMARY KEY,
    location VARCHAR(100),
    vehicle_count INT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

2. PYTHON CODE

```
import tkinter as tk
from tkinter import messagebox, ttk
import pymysql
import matplotlib.pyplot as plt
from datetime import datetime
# Connect to MySQL
def connect_db():
    return pymysql.connect(host="localhost", user="root", password="",
database="traffic_db")
# Insert new data
```

```
def insert_data():
    location = entry_location.get()
    try:
        vehicle_count = int(entry_vehicle.get())
    except ValueError:
        messagebox.showerror("Input Error", "Vehicle count must be an integer")
        return
    con = connect_db()
    cursor = con.cursor()
    cursor.execute("INSERT INTO traffic_data (location, vehicle_count) VALUES
(%s, %s)", (location, vehicle_count))
    con.commit()
    con.close()
    messagebox.showinfo("Success", "Traffic data inserted")
    entry_location.delete(0, tk.END)
    entry_vehicle.delete(0, tk.END)
    update_last_updated()
    show_all_data()
# Graph
def show_graph():
    con = connect_db()
    cursor = con.cursor()
```

```
    cursor.execute("SELECT location, SUM(vehicle_count) FROM traffic_data
GROUP BY location")
    results = cursor.fetchall()
    con.close()
    if not results:
        messagebox.showinfo("No Data", "No traffic data available to plot")
        return
    locations = [row[0] for row in results]
    vehicle_counts = [row[1] for row in results]
    plt.figure(figsize=(8, 5))
    plt.bar(locations, vehicle_counts, color='dodgerblue')
    plt.title("Vehicle Count per Location")
    plt.xlabel("Location")
    plt.ylabel("Total Vehicle Count")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
# Table update
def show_all_data():
    for row in tree.get_children():
        tree.delete(row)
    con = connect_db()
    cursor = con.cursor()
```

```
    cursor.execute("SELECT * FROM traffic_data ORDER BY timestamp DESC")
    rows = cursor.fetchall()
    con.close()
    for row in rows:
        tree.insert("", tk.END, values=row)
# Delete entry
def delete_entry():
    id_ = entry_delete.get()
    if not id_.isdigit():
        messagebox.showerror("Input Error", "Enter valid ID")
        return
    con = connect_db()
    cursor = con.cursor()
    cursor.execute("DELETE FROM traffic_data WHERE id=%s", (id_,))
    con.commit()
    con.close()
    messagebox.showinfo("Deleted", f"Record with ID {id_} deleted")
    entry_delete.delete(0, tk.END)
    show_all_data()
# Search
def search_by_location():
    location = entry_search.get()
    if not location:
```

```
        messagebox.showerror("Input Error", "Enter a location")
        return
    for row in tree.get_children():
        tree.delete(row)
    con = connect_db()
    cursor = con.cursor()
    cursor.execute("SELECT * FROM traffic_data WHERE location=%s", (location,))
    rows = cursor.fetchall()
    con.close()
    if not rows:
        messagebox.showinfo("Not Found", "No records for this location")
    for row in rows:
        tree.insert("", tk.END, values=row)
# Timestamp
def update_last_updated():
    con = connect_db()
    cursor = con.cursor()
    cursor.execute("SELECT MAX(timestamp) FROM traffic_data")
    result = cursor.fetchone()
    con.close()
    last_updated.set(f"Last Updated: {result[0] if result[0] else 'N/A'}")
# ------------------------- GUI -------------------------
root = tk.Tk()
```

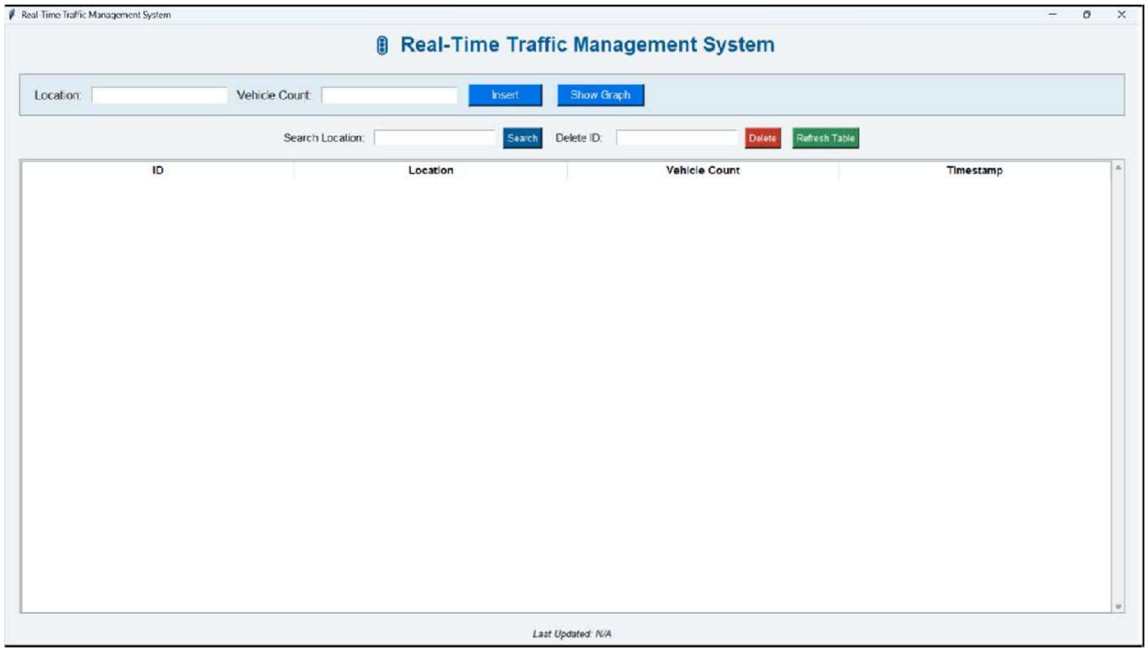
```
root.title("Real-Time Traffic Management System")

root.geometry("950x650")

root.config(bg="#f0f4f7")

# Title

tk.Label(root, text="Real-Time Traffic Management System", font=("Arial", 20,
"bold"), fg="#005b96", bg="#f0f4f7").pack(pady=10)

# Top Input Frame

input_frame = tk.Frame(root, bg="#e0ecf3", padx=10, pady=10, bd=2,
relief="groove")

input_frame.pack(pady=10, fill="x", padx=20)

tk.Label(input_frame, text="Location:", bg="#e0ecf3", font=("Arial",
12)).grid(row=0, column=0, padx=5, pady=5)

entry_location = tk.Entry(input_frame, font=("Arial", 12))

entry_location.grid(row=0, column=1, padx=5)

tk.Label(input_frame, text="Vehicle Count:", bg="#e0ecf3", font=("Arial",
12)).grid(row=0, column=2, padx=5, pady=5)

entry_vehicle = tk.Entry(input_frame, font=("Arial", 12))

entry_vehicle.grid(row=0, column=3, padx=5)

tk.Button(input_frame, text="Insert", command=insert_data, bg="#0073e6",
fg="white", font=("Arial", 11), width=10).grid(row=0, column=4, padx=10)

tk.Button(input_frame, text="Show Graph", command=show_graph, bg="#0073e6",
fg="white", font=("Arial", 11), width=12).grid(row=0, column=5, padx=10)

# Search & Delete Frame

action_frame = tk.Frame(root, bg="#f0f4f7")
```

```
action_frame.pack(pady=5)

tk.Label(action_frame, text="Search Location:", font=("Arial", 11),
bg="#f0f4f7").grid(row=0, column=0, padx=5)

entry_search = tk.Entry(action_frame, font=("Arial", 11))

entry_search.grid(row=0, column=1, padx=5)

tk.Button(action_frame, text="Search", command=search_by_location,
bg="#005b96", fg="white", font=("Arial", 10)).grid(row=0, column=2, padx=5)

tk.Label(action_frame, text="Delete ID:", font=("Arial", 11),
bg="#f0f4f7").grid(row=0, column=3, padx=10)

entry_delete = tk.Entry(action_frame, font=("Arial", 11))

entry_delete.grid(row=0, column=4, padx=5)

tk.Button(action_frame, text="Delete", command=delete_entry, bg="#c0392b",
fg="white", font=("Arial", 10)).grid(row=0, column=5, padx=5)

tk.Button(action_frame, text="Refresh Table", command=show_all_data,
bg="#2e8b57", fg="white", font=("Arial", 10)).grid(row=0, column=6, padx=10)

# Data Table

tree_frame = tk.Frame(root, bd=2, relief="groove")

tree_frame.pack(pady=10, padx=20, fill="both", expand=True)

cols = ("ID", "Location", "Vehicle Count", "Timestamp")

tree = ttk.Treeview(tree_frame, columns=cols, show='headings')

for col in cols:

    tree.heading(col, text=col)

    tree.column(col, anchor="center")

style = ttk.Style()
```

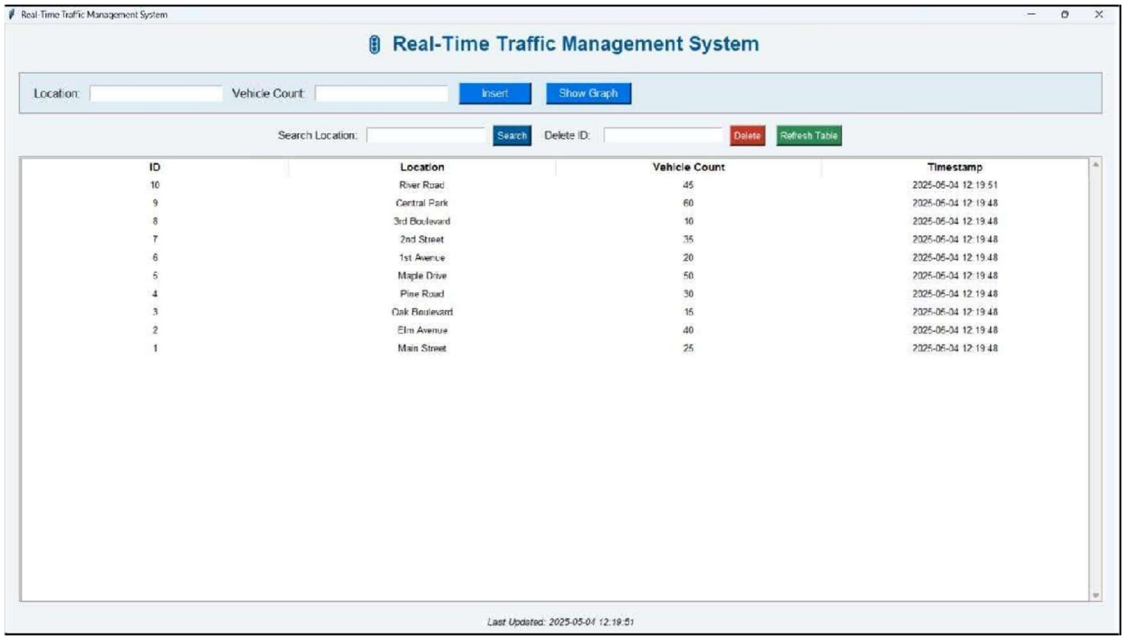
```
style.configure("Treeview", font=("Arial", 10), rowheight=25)

style.configure("Treeview.Heading", font=("Arial", 11, "bold"))

scrollbar = ttk.Scrollbar(tree_frame, orient="vertical", command=tree.yview)

scrollbar.pack(side="right", fill="y")

tree.configure(yscroll=scrollbar.set)

tree.pack(fill="both", expand=True)

# Last Updated

last_updated = tk.StringVar()

tk.Label(root, textvariable=last_updated, bg="#f0f4f7", font=("Arial", 10,
"italic")).pack(pady=5)

update_last_updated()

show_all_data()

root.mainloop()
```

APPENDIX B (Screenshots)

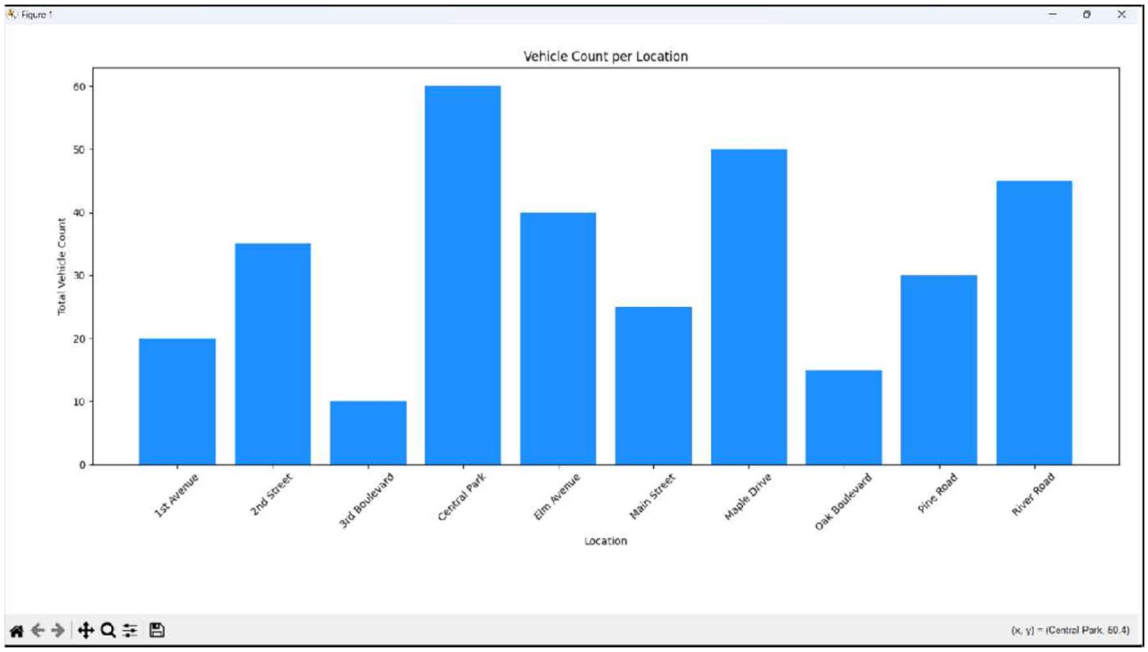
1. DASHBOARD PAGE



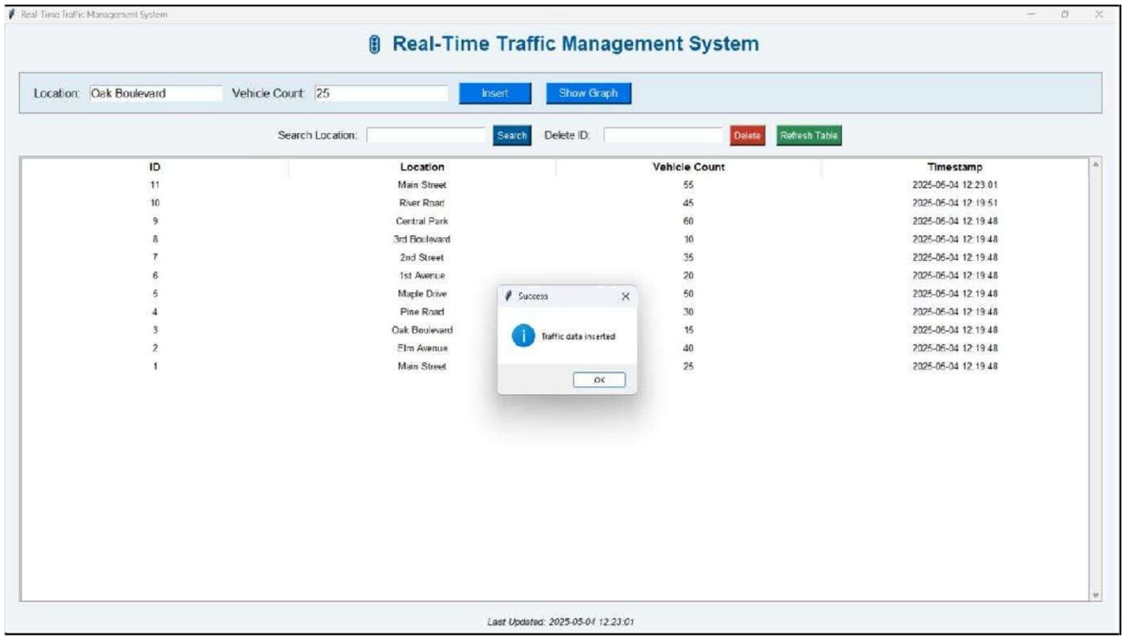
2. INSERTED DATA



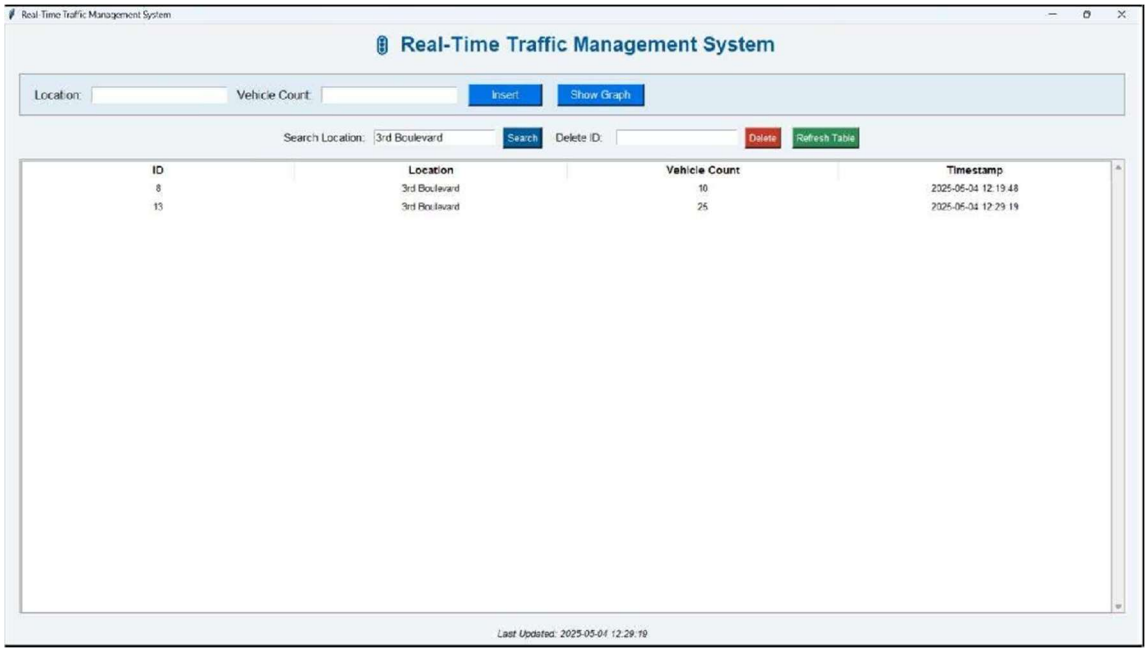
3. GRAPHICAL ANALYSIS



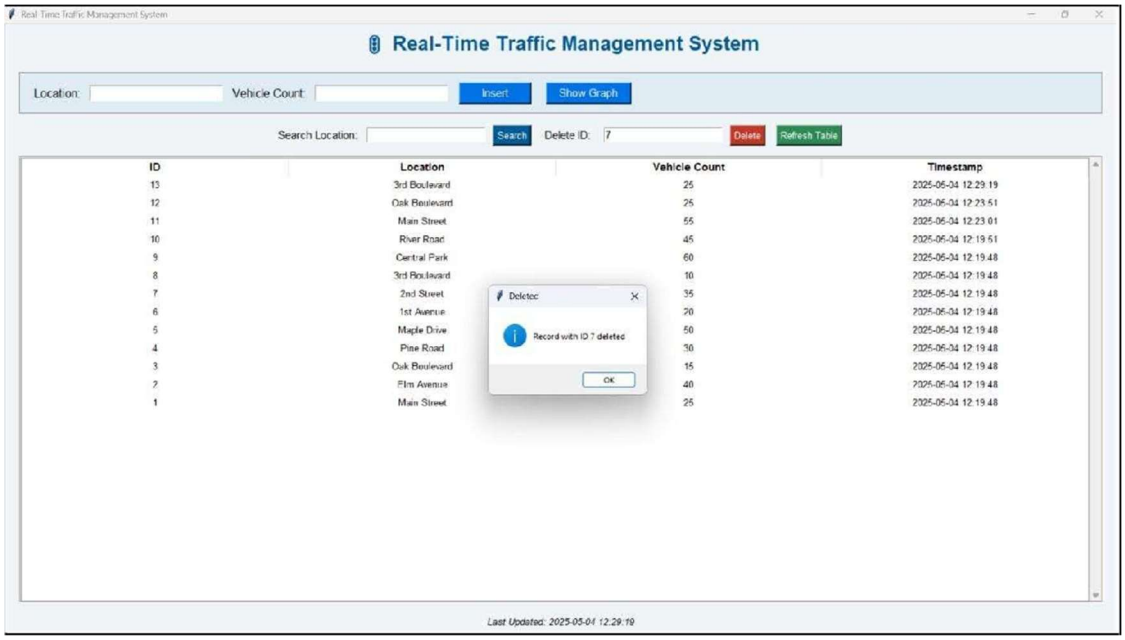
4. TRAFFIC DATA INSERTION



5. FILTERING OR SEARCH LOCATION DATA



6. DELETING DATA



7. MYSQL OUTPUT

```
MariaDB [(none)]> use  traffic_db;
Database changed
MariaDB [traffic_db]> show tables;
+----------------------+
| Tables_in_traffic_db |
+----------------------+
| traffic_data         |
+----------------------+
1 row in set (0.001 sec)

MariaDB [traffic_db]> select * from traffic_data;
+----+---------------+---------------+---------------------+
| id | location      | vehicle_count | timestamp           |
+----+---------------+---------------+---------------------+
|  1 | Main Street   |            25 | 2025-05-04 12:19:48 |
|  2 | Elm Avenue    |            40 | 2025-05-04 12:19:48 |
|  3 | Oak Boulevard |            15 | 2025-05-04 12:19:48 |
|  4 | Pine Road     |            30 | 2025-05-04 12:19:48 |
|  5 | Maple Drive   |            50 | 2025-05-04 12:19:48 |
|  6 | 1st Avenue    |            20 | 2025-05-04 12:19:48 |
|  8 | 3rd Boulevard |            10 | 2025-05-04 12:19:48 |
|  9 | Central Park  |            60 | 2025-05-04 12:19:48 |
| 10 | River Road    |            45 | 2025-05-04 12:19:51 |
| 11 | Main Street   |            55 | 2025-05-04 12:23:01 |
| 12 | Oak Boulevard |            25 | 2025-05-04 12:23:51 |
| 13 | 3rd Boulevard |            25 | 2025-05-04 12:29:19 |
+----+---------------+---------------+---------------------+
12 rows in set (0.001 sec)
```

REFERENCES

1. **MySQL Documentation:** https://dev.mysql.com/doc/ A comprehensive resource for understanding MySQL, which is essential for managing the database that stores traffic data in the application.
2. **Python Documentation:** https://docs.python.org/3/ This official documentation provides in-depth information about Python programming, which is crucial for developing the backend logic of the traffic management system.
3. **Tkinter Documentation:** https://docs.python.org/3/library/tkinter.html A valuable resource for learning about Tkinter, the standard GUI toolkit for Python, which is used to create the user interface of the application.
4. **Matplotlib Documentation:** https://matplotlib.org/stable/contents.html This documentation offers guidance on using Matplotlib for data visualization, which is key for generating graphs that display traffic data trends.
5. **Pymysql Documentation:** https://pymysql.readthedocs.io/en/latest/ A useful resource for understanding how to use the PyMySQL library to connect Python applications to MySQL databases, facilitating data operations in the traffic management system.
6. **W3Schools - SQL:** https://www.w3schools.com/sql/ This tutorial will help you learn SQL, which is crucial for storing traffic data in a database and understanding how to interact with the database effectively.