

SMART TRAFFIC MANAGEMENT SYSTEM

<https://github.com/dullah-commits/Smart-traffic-management-system>

Instructor: Dr. Zunnurain Hussain

Authors: Muhammad Abdullah (BSSE23073) & Hira Ijaz (BSSE23103)

Course: Software Construction & Development

Platform: Amazon Web Services & React.js

Table of Contents

1. Abstract	1
2. Introduction	2
○ 2.1 Project Motivation	
○ 2.2 System Scope	
3. System Architecture	3
○ 3.1 Cloud Infrastructure (AWS VPC, EC2, RDS)	
○ 3.2 Load Balancing with ALB	
4. Front-End Development	5
○ 4.1 React & Vite Build Process	
○ 4.2 Interactive Mapping with Leaflet.js	
5. Data Management	7
○ 5.1 Relational Database Service (RDS) Configuration	
○ 5.2 Security Group Chaining & Connectivity	
6. Implementation Results	9
7. Conclusion & Future Work	10
8. References	11

List of Figures

- **Figure 1:** High-Level AWS 3-Tier Architecture Diagram
- **Figure 2:** Live Traffic Dashboard for Lahore City
- **Figure 3:** Security Group Inbound Rules for RDS Port 3306
- **Figure 4:** Vite Production Build Process and Asset Generation

List of Tables

- **Table 1:** AWS Resource Allocation and Service Utilization
- **Table 2:** Junction Flow Rate Thresholds for Signal Optimization

- **Table 3:** Security Group Chaining Matrix

List of Equations

- **Equation 1:** Traffic Flow Density Calculation:
 - **Equation 2:** Dynamic Green Light Duration Formula: $D = n G / L = (V_c / V_t) \times C_{cycle}$
-

Technical Analysis & Writing

1. Cloud Infrastructure and Scalability

The system is deployed on a **3-tier AWS architecture** designed for high availability. The **Application Load Balancer (ALB)** serves as the entry point, distributing traffic to an **EC2 instance** located within a private subnet to ensure security. This configuration prevents direct exposure of the application server to the public internet.

2. Data Integrity and Security

Data management is handled via **Amazon RDS (MySQL)**. To maintain a "Least Privilege" security model, we implemented **Security Group Chaining**. The database security group (**DatabaseSG**) only accepts inbound traffic from the web server's security group (**WebServerSG**) on port **3306**, effectively creating a secure internal tunnel for traffic metadata.

3. Geospatial Visualization

The frontend utilizes **Leaflet.js** to provide a real-time heat map of 26 major junctions in Lahore. By fetching flow rates from the RDS instance, Leaflet dynamically updates circle markers—representing congestion levels—on top of OpenStreetMap tiles.

Traceable References

APA Style (7th Edition)

- iamrishav111. (2021). *Smart-traffic-management-system: Deals with an idea to remove traffic congestion* [Computer software]. GitHub.
<https://github.com/iamrishav111/Smart-Traffic-Management-System>
- sunilkumarmaurya786693. (2020). *Intelligence-traffic-monitoring-system: ITMS using image processing and machine learning* [Computer software]. GitHub.
<https://github.com/sunilkumarmaurya786693/Intelligence-traffic-monitoring-system> **IEEE**

Style

- [1] iamrishav111, "Smart-Traffic-Management-System," GitHub, 2021. [Online]. Available: <https://github.com/iamrishav111/Smart-Traffic-Management-System>
- [2] sunilkumarmaurya786693, "Intelligence-traffic-monitoring-system," GitHub, 2020. [Online]. Available: <https://github.com/sunilkumarmaurya786693/Intelligence-traffic-monitoring-system> •
- [3] Amazon Web Services, "Amazon RDS connectivity and security," AWS Documentation, 2026. [Online]. Available: <https://aws.amazon.com/rds/>