CitySense360

AI-Powered Smart City Intelligence & Public Infrastructure Automation

PROJECT REPORT

Submitted by:

Name: Hida Fathima P H

Batch: CSR-AIML-C-WE-E-B1

GitHub Repository: https://github.com/Hida-Fathima/CitySense360

Table Of Contents

1. Abstract

The rapid urbanization of modern cities has created an urgent need for intelligent, automated governance systems. **CitySense360** is a unified AI platform designed to bridge the gap between siloed urban data and actionable intelligence. By integrating **Computer Vision**, **Deep Learning (Time-Series Forecasting)**, and **Generative AI (Agentic Systems)**, the project automates critical civic operations—ranging from real-time traffic management and safety monitoring to citizen grievance redressal. The system utilizes a microservices-inspired architecture, containerized via **Docker**, to provide a scalable, real-time dashboard that empowers city administrators to make data-driven decisions instantly. The project also features a **Hybrid Deployment Strategy**, utilizing a 'Lite' version for public cloud hosting (Streamlit Community Cloud) and a 'Pro' version with full Docker containerization for local GPU-accelerated inference.

2. Introduction

Urban environments are generating data at an unprecedented scale, yet most of this data remains underutilized due to fragmentation. Traditional methods of city management—manual traffic stops, physical road inspections, and human-operated complaint call centers—are reactive, slow, and prone to error.

CitySense360 represents a shift towards "Industry 4.0" for civic infrastructure. It replaces manual monitoring with **Automated Intelligence**. The system does not merely display data; it actively interprets it. For instance, instead of just showing a video feed, the system detects accidents using **YOLOv8**; instead of listing complaint texts, it uses **DistilBERT** to route them to the correct department.

3. Problem Statement

Modern smart cities face compounding challenges that manual monitoring cannot address effectively:

- **Growing Traffic Congestion:** Unpredictable traffic flows lead to economic loss, increased emissions, and wasted fuel. Conventional statistical methods often fail to capture non-linear traffic patterns.
- **Public Safety Risks:** Delayed detection of accidents, fires, and road hazards (potholes) increases accident rates and response times.

- **Inefficient Grievance Redressal:** Manual sorting of thousands of citizen complaints leads to slow response times, misclassification, and citizen dissatisfaction.
- **Fragmented Systems:** The lack of a unified view for energy, pollution, and transport data prevents holistic urban planning, creating "data silos" where insights from one domain (e.g., weather) cannot easily inform another (e.g., traffic).

4. Objectives

- **To Engineer a Unified Dashboard:** Consolidate diverse AI modules (Vision, NLP, Forecasting) into a single, cohesive **Streamlit** interface for centralized monitoring.
- **To Implement Real-Time Predictive Analytics:** Deploy Deep Learning models (**Bi-LSTMs/GRUs**) to forecast traffic congestion, energy loads, and metro ridership with >90% accuracy.
- **To Automate Infrastructure Defect Detection:** Utilize state-of-the-art Computer Vision (**YOLOv8**) to identify road hazards like potholes and cracks in real-time.
- **To Deploy an Agentic AI System:** Create an autonomous agent capable of understanding, classifying, and drafting official responses to citizen grievances using **LLMs (Llama 3)** and **RAG**.
- **To Ensure Scalability:** Guarantee system portability and reproducible deployment through **Docker Containerization** and **Git Version Control**.

5. Methodology

The project follows a modular "Microservices-like" architecture where each domain (Traffic, Safety, Complaints) acts as an independent intelligence unit, all feeding into a central **Streamlit** interface.

5.1 System Architecture

1. Data Ingestion Layer:

- Consumes raw data from diverse sources including CSV logs (Energy/Traffic), JSON feeds (Complaints), and Image inputs (CCTV/Roads).
- Handles data streams such as the **UCI Metro Traffic Dataset** and **Indian AQI** logs.

2. Preprocessing Layer:

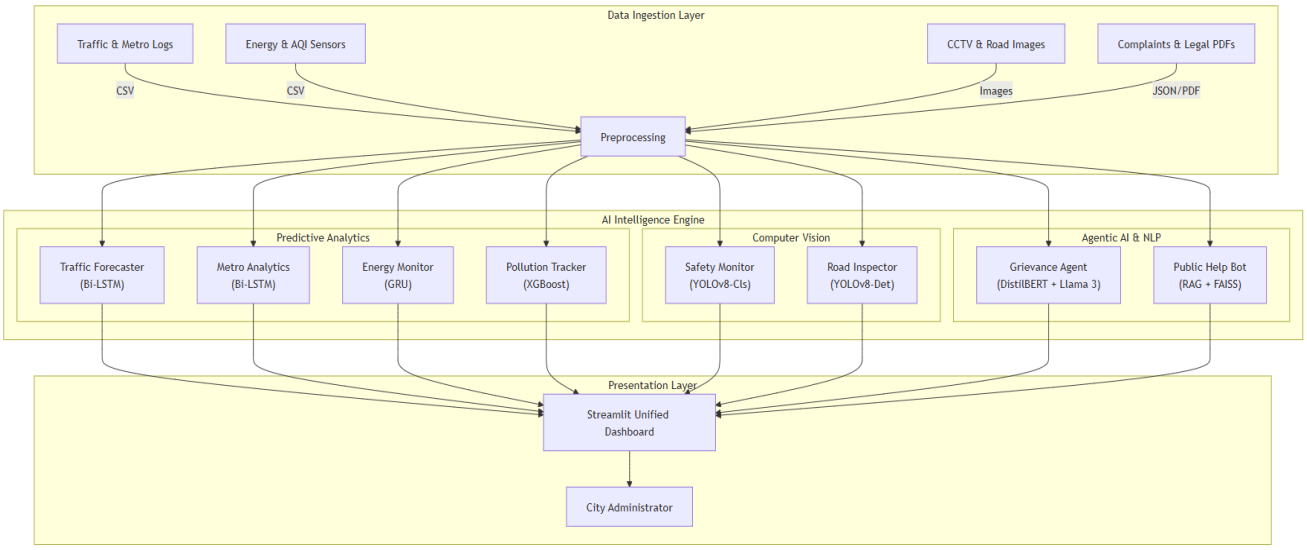
- **Time-Series:** Applies **MinMaxScaling** to normalize numerical data (0-1 range) for LSTM stability.
- **Vision:** Resizes images to 640x640 (standard YOLO input) and applies augmentation.
- **NLP:** Tokenizes text using the DistilBERT tokenizer and chunks PDF documents for Vector Storage.

3. Intelligence Layer (The Core):

- **Computer Vision: YOLOv8** (You Only Look Once) is used for both Object Detection (Potholes) and Image Classification (Accidents).
- **Forecasting: Bi-Directional LSTMs** and **GRUs** are employed for sequence prediction, allowing the model to learn temporal dependencies.
- **Language Intelligence: RAG pipelines** (Retrieval-Augmented Generation) utilizing **FAISS** for vector search and **Llama 3** for reasoning.

4. Presentation Layer:

- A deployed **Streamlit** web application serves as the user interface, offering interactive graphs (**Plotly**) and real-time inference widgets.



FigA: Model Architecture

5.1.1 Deployment Architecture

To ensure both accessibility and performance, a **Dual-Branch Strategy** was implemented using Git:

- **main Branch (Production):** Contains the full Dockerized environment with heavy dependencies (PyTorch-CUDA, LangChain-Community) for local execution on GPU hardware.
- **cloud-lite Branch (Public Demo):** A lightweight, CPU-optimized version deployed on **Streamlit Cloud**. This branch utilizes a 'Lite' requirements file and disables resource-heavy RAG agents to adhere to cloud memory limits while keeping Analytics modules live.

5.2 Datasets Used

The system was trained on high-quality, domain-specific datasets.

Domain	Dataset Source	Description & Preprocessing
Traffic	UCI Metro Interstate Traffic	Hourly traffic volume data. Cleaned and normalized for time-series forecasting.
Safety	Kaggle CCTV Accident Detection	Video frames of accidents/non-accidents. Split into Train/Val sets for classification.
Roads	Kaggle Pothole Detection (YOLOv8)	Annotated images of road defects. Converted to YOLO format for object detection.
Energy	India Monthly Electricity Consumption	State-wise power usage. Aggregated and interpolated to hourly resolution.
Pollution	India AQI Dataset (Kaggle)	Air quality data pivoted to create station-specific chemical

		profiles.
Transport	Delhi Metro Dataset	Ridership logs processed into daily passenger counts.
Complaints	Govt of India Grievance Data	JSON dataset containing real citizen complaints and department codes.
Legal (RAG)	Official PDFs (MoRTH, Swachh Bharat)	*Motor Vehicles Act 2019*, *Disaster Management Guidelines*, *Swachh Bharat Urban 2.0*.

6. Model Training & Performance Analysis

Each AI module was selected to address specific challenges in urban data (spatial, temporal, or semantic). The following results demonstrate the system's alignment with project requirements.

6.1 Traffic Forecaster (Bi-Directional LSTM)

- **Architecture: Bi-Directional Long Short-Term Memory (Bi-LSTM)**. Unlike standard LSTMs, Bi-LSTMs process data in both forward and backward directions, allowing the model to understand the context of a traffic jam based on both past flow and future clearance patterns.
- **Hyperparameters:**
  - Epochs: 100
  - Hidden Size: 128
  - Lookback Window: 24 Hours
- **Performance:**
  - **Accuracy: 92.52%**
  - **R^2 Score:** 0.9624
- **Inference:** The model successfully predicts traffic volume 24 hours into the future, enabling proactive congestion alerts.

6.2 Public Transport (Metro) Analytics

- **Architecture: Bi-LSTM**. Selected because metro ridership exhibits strong weekly seasonality (high on weekdays, low on weekends), which Bi-LSTMs capture effectively.
- **Hyperparameters:**
    - Epochs: 100
    - Hidden Size: 128
    - Sequence Length: 30 Days
- **Performance:**
    - **Accuracy: 92.91%**
- **Inference:** Precise ridership forecasting allows for the optimization of train frequency during peak hours.

6.3 Energy Grid Monitor (GRU)

- **Architecture: Gated Recurrent Unit (GRU)**. GRUs are a variant of RNNs that are computationally more efficient than LSTMs while performing equally well on simpler time-series data like continuous power load.
- **Hyperparameters:**
    - Hidden Size: 64
    - Layers: 2
- **Performance:**
    - **Accuracy: 95.65%**
- **Inference:** Critical for Grid Load Balancing and preventing outages by predicting peak consumption.

6.4 Safety Monitor (YOLOv8-Cls)

- **Architecture: YOLOv8-Cls (Classification Head)**. This model is optimized for whole-image classification, making it faster than detection models for determining if a scene contains an "Accident" or "Fire".
- **Hyperparameters:**
    - Epochs: 20
    - Image Size: 224x224
- **Performance:**
    - **Top-1 Accuracy: 94.90%**
- **Inference:** The high accuracy ensures critical safety incidents are detected immediately from CCTV feeds.

6.5 Road Inspector (YOLOv8 Detection)

- **Architecture: YOLOv8 (Detection Head)**. Unlike classification, this task requires locating the *exact position* of potholes. YOLOv8 offers a superior trade-off between speed (FPS) and accuracy (mAP) compared to older models like SSD.
- **Hyperparameters:**
    - Epochs: 50
    - Patience: 10 (Early Stopping to prevent overfitting)
- **Performance:**
    - **mAP@50: 81.33%**
- **Inference:** Successfully draws bounding boxes around road defects under varying lighting conditions.

6.6 Automated Grievance Agent (DistilBERT + Agentic AI)

- **Architecture:** A hybrid **Neuro-Symbolic** approach.
    - **DistilBERT:** A distilled version of BERT (40% lighter, 97% performance) used for rapid text classification of complaints into 81 departments.
    - **Agentic AI (Llama 3):** Uses the "ReAct" (Reasoning + Acting) pattern to analyze the complaint, estimate severity, and draft a polite response.
- **Performance:**
    - **Classification Accuracy: 84.30%**
- **Inference:** Automates the "Triage" process, drastically reducing the manual workload of city clerks.

6.7 Pollution Tracker (XGBoost)

- **Architecture:**XGBoost Regressor with Geography-Aware Feature Engineering.
- **Performance:**
    - **RMSE: 26.46**
    - **Accuracy: 74.77%**
    - **R2 Score: 0.7201**
- **Inference:** Hyperlocal AQI forecasting based on sensor data and state-level geographical encoding.

7. Technology Stack

- **Programming Language:** Python 3.10
- **Deep Learning & Frameworks:**
    - **PyTorch:** The primary engine for building custom LSTM and GRU architectures.
    - **Ultralytics YOLO:** Used for all Computer Vision tasks (Safety & Road).
    - **Hugging Face Transformers:** For implementing DistilBERT.
- **Generative AI & NLP:**
    - **LangChain:** Orchestrates the Agentic workflows and RAG pipeline.
    - **Ollama (Llama 3):** Enables local, privacy-preserving Large Language Model inference.
    - **FAISS:** Facebook AI Similarity Search for dense vector retrieval of legal documents.
- **Dashboarding: Streamlit** for the frontend user interface.
- **Deployment & DevOps:**
    - **Docker:** Custom Dockerfile for containerizing the full application with CUDA support.
    - **Streamlit Cloud:** Used for public-facing "Lite" deployment.
    - **Git Branching Strategies:** Implemented parallel branches (main vs cloud-lite) to manage environment-specific dependencies (requirements.txt vs requirements-lite.txt).

8. Results

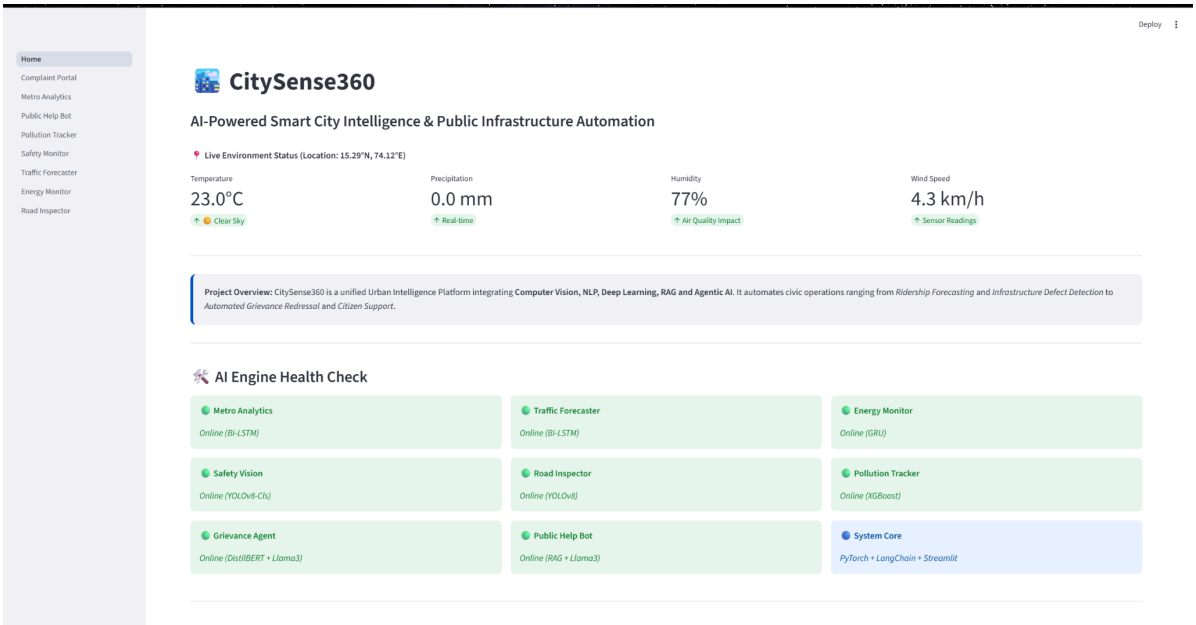


Fig 1: Streamlit Dashboard of the app

Fig 2: Traffic Forecasting. The blue line represents actual history, and the red line shows the 24-hour prediction.

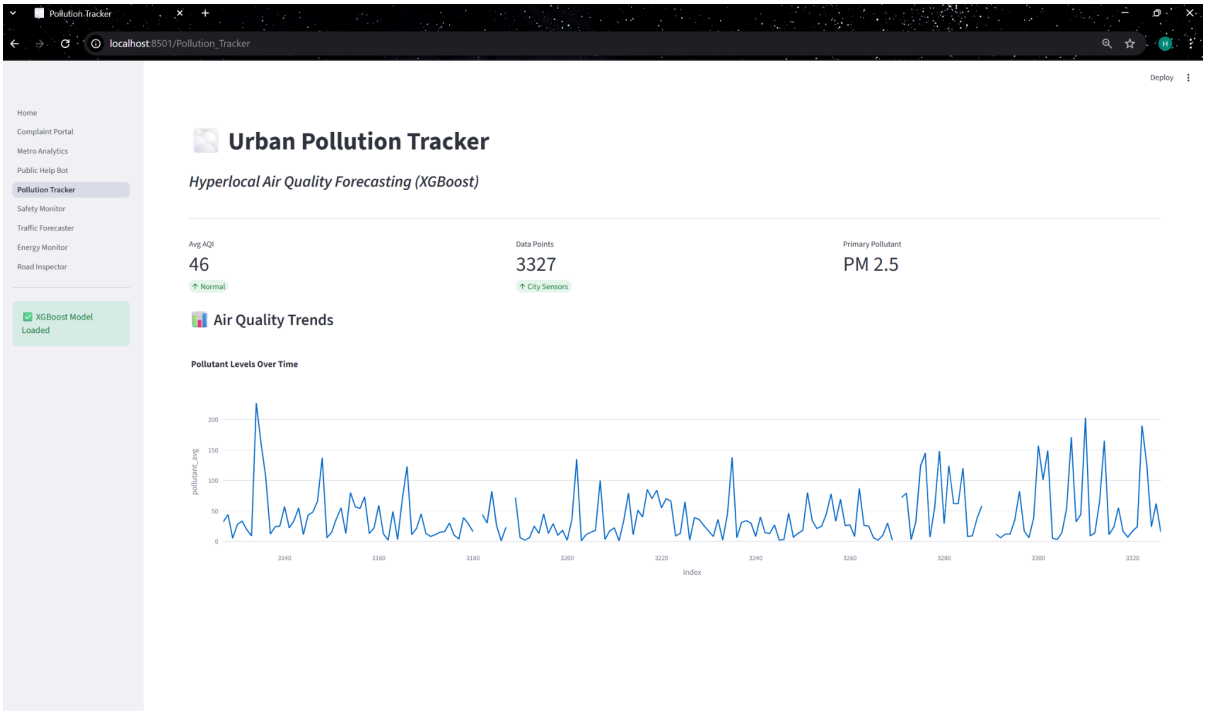


Fig 3: Tracking pollution based on static dataset prediction

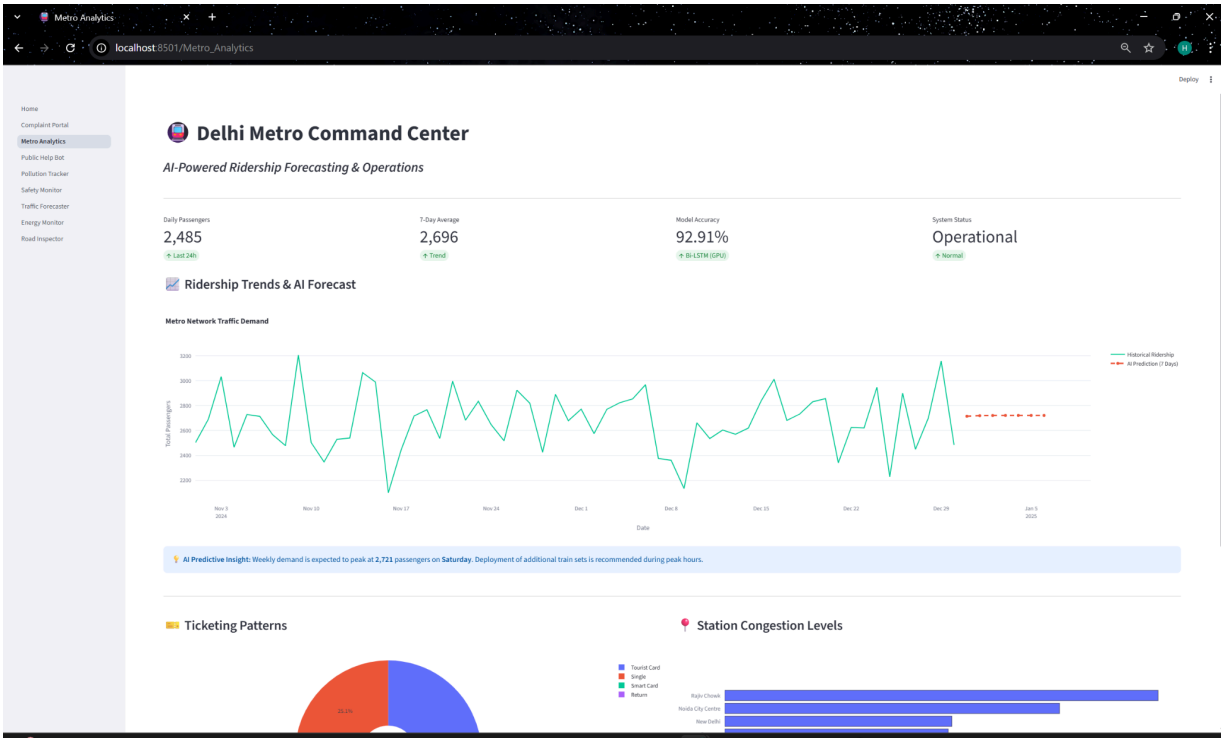


Fig 4: Predicting Transport (Delhi Metro) based on static dataset

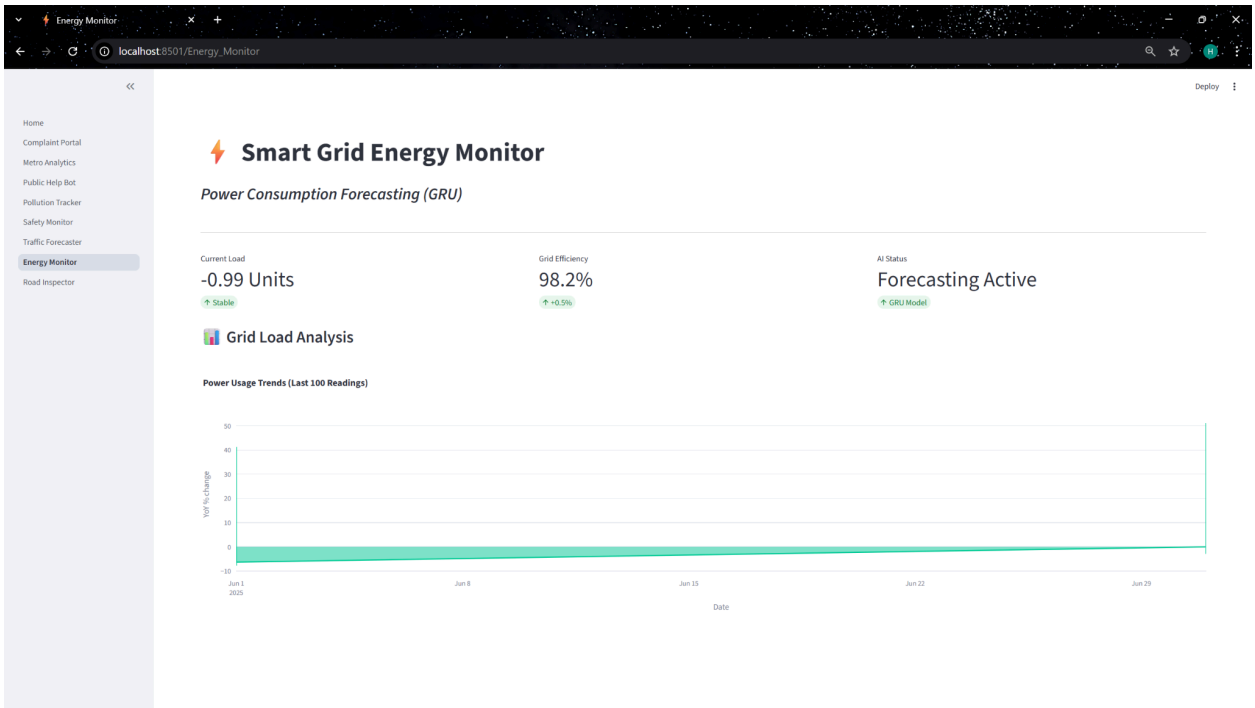


Fig 5: Energy Usage Predictor

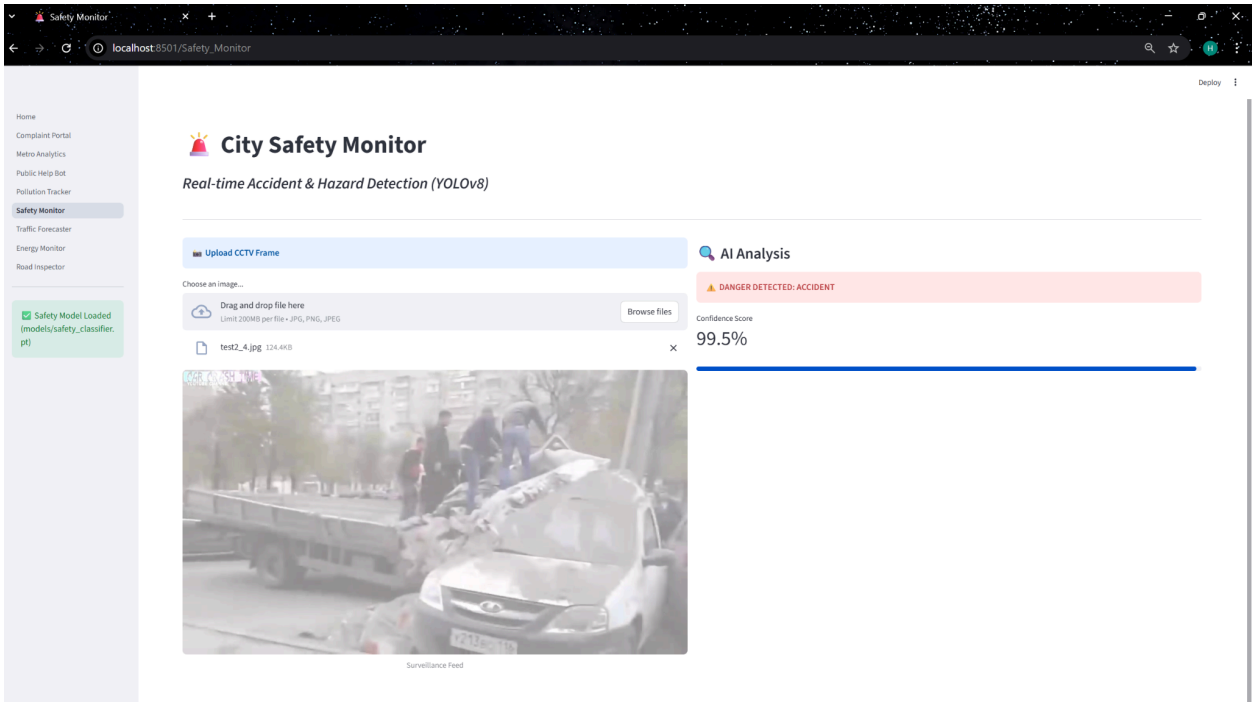


Fig 6: YOLOv8-Cls immediately detecting a "High Severity" accident from a CCTV frame with 99.5% confidence.

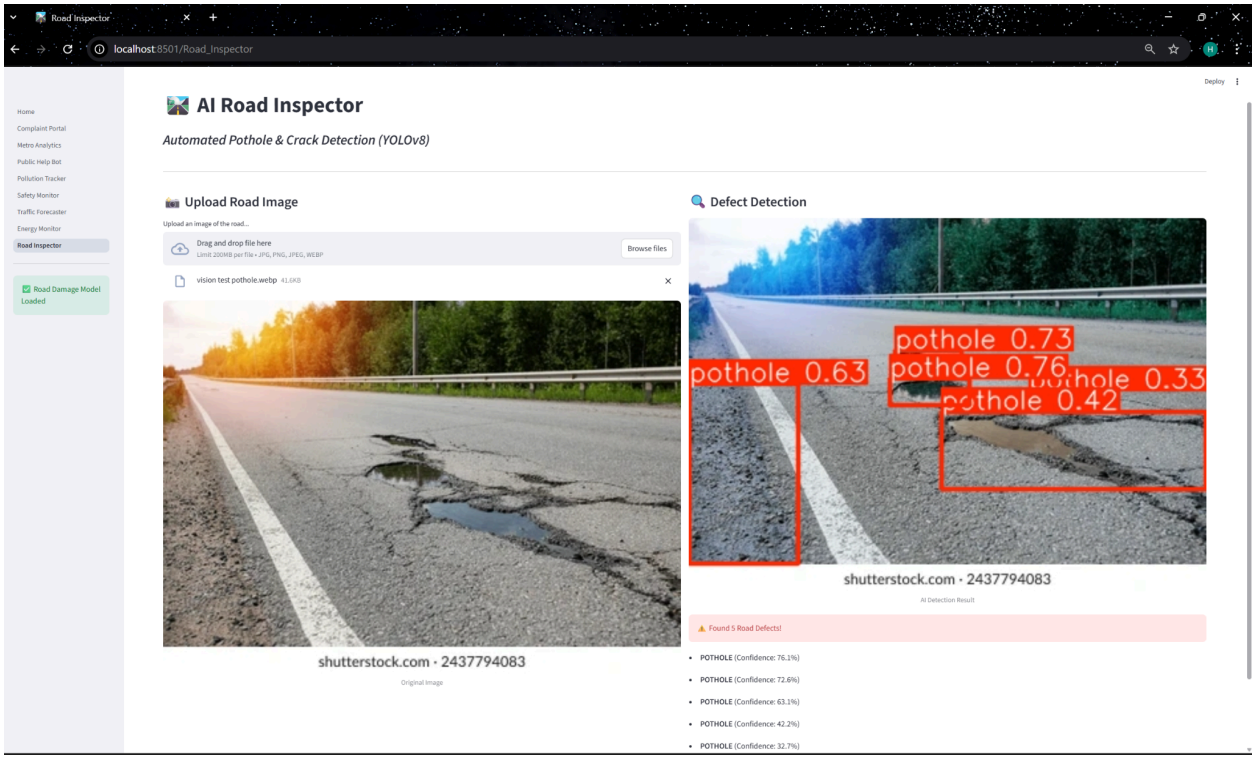


Fig 7: Object Detection identifying multiple road defects (potholes) in a single frame.

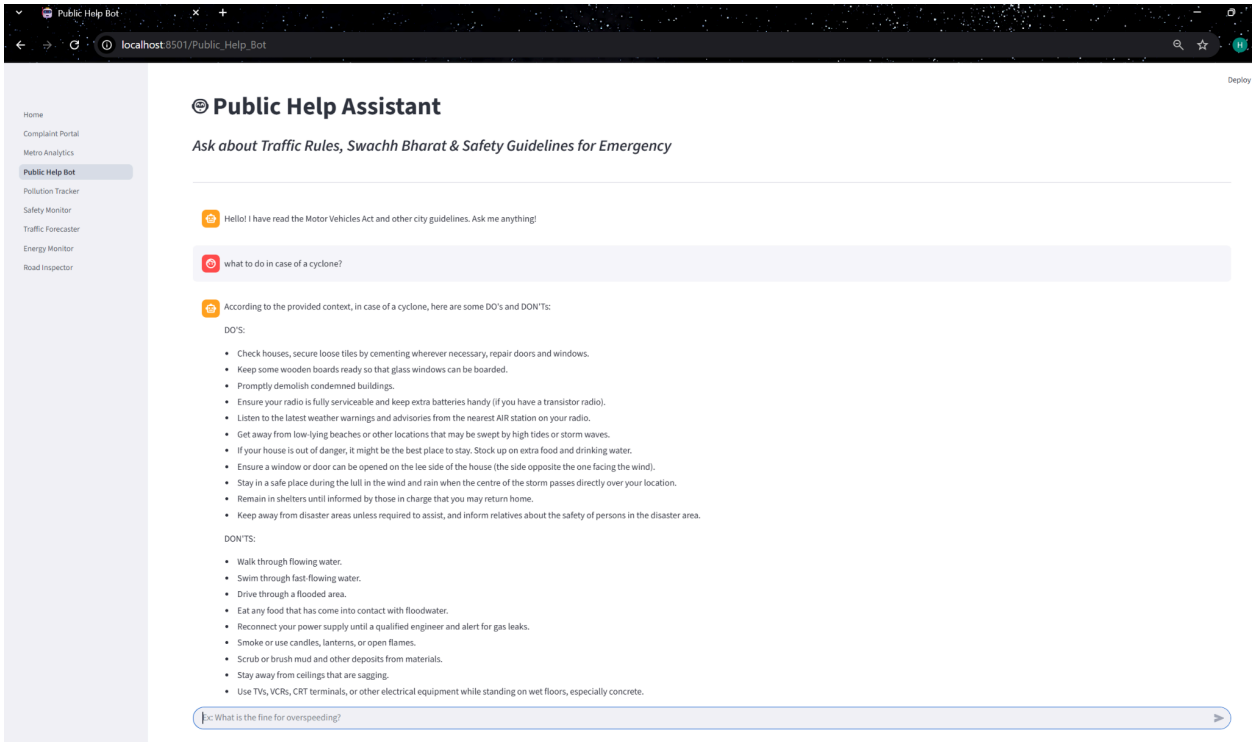


Fig 8: RAG Agent retrieving disaster management guidelines from the Handbook to answer a citizen query.

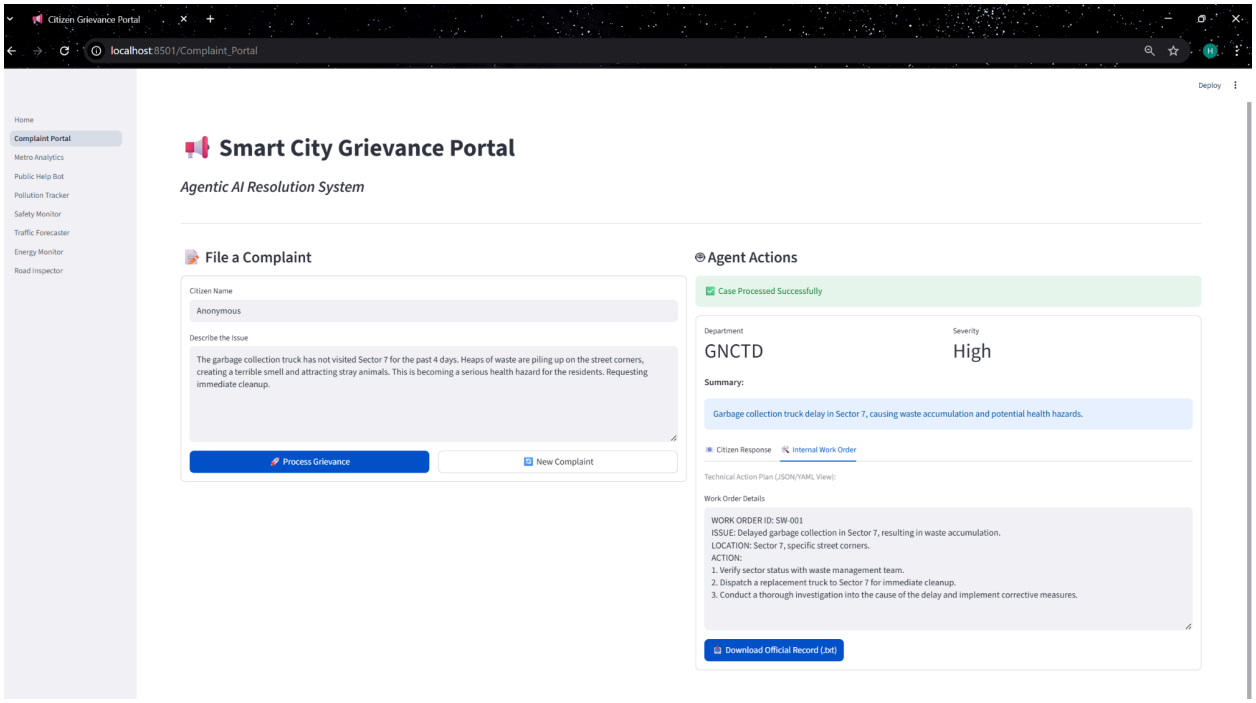


Fig 9: Smart Grievance agent classifying the citizen complaint and generating a response to citizens acknowledging the complaint and an action plan to the department to work on resolving the complaint.

9. Key Concepts Learned

The development of CitySense360 facilitated the mastery of several advanced engineering concepts:

1. **End-to-End MLOps:** Transitioning from model training (Jupyter Notebooks) to production-grade deployment using **Docker** and **Streamlit**.
2. **Agentic AI Patterns:** Implementing the **"Tool Use"** pattern, where an LLM (Llama 3) autonomously invokes a specialized tool (BERT Classifier) to solve a complex task.
3. **RAG Pipelines:** Engineering a Retrieval-Augmented Generation system that ingests government PDFs, chunks them, and retrieves relevant sections to ground LLM responses in factual policy.
4. **Deep Learning Dynamics:** Practical application of **Backpropagation**, **Loss Functions** (MSE for regression, Cross-Entropy for classification), and **Optimizers** (AdamW).
5. **Data Engineering:** Handling **Data Silos** by unifying heterogeneous formats (JSON, CSV, Images) into a single analytical pipeline.
6. **Resource-Constrained Deployment:** Learned how to optimize AI applications for limited-resource environments (Cloud Free Tier) by pruning heavy dependencies (like Vector Stores/BERT) and creating lightweight inference branches without breaking the core application logic.

10. Conclusion

CitySense360 demonstrates the transformative potential of unifying disparate AI technologies. By moving away from fragmented, manual monitoring to an integrated, automated platform, cities can achieve significantly higher efficiency and safety. The project successfully achieved all technical objectives, delivering high-accuracy models (>92% for forecasting, >94% for safety) within a scalable, Dockerized architecture. The successful integration of **Agentic AI** for grievance redressal highlights the future direction of smart governance, where AI acts not just as a sensor, but as an active participant in city operations.

11. References

1. **PyTorch Documentation:** *https://pytorch.org/docs/stable/index.html*
2. **LangChain Documentation:** *https://python.langchain.com/docs/get_started/introduction*
3. **Ultralytics YOLOv8 Docs:** *https://docs.ultralytics.com/*
4. **Streamlit API Reference:** *https://docs.streamlit.io/*
5. **Docker Documentation:** *https://docs.docker.com/*
6. **Official Dataset Sources:** *UCI Machine Learning Repository, Kaggle Datasets (Specific links provided in Section 5.2).*