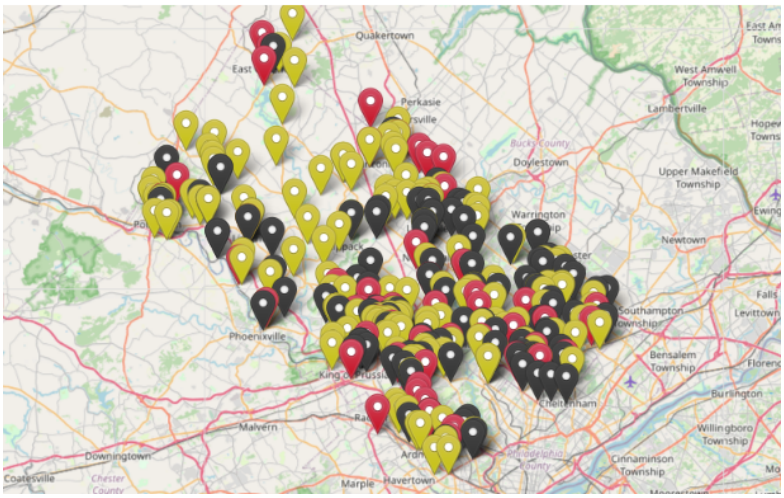
Development of an Emergency Response Dashboard and Associated Infrastructure - Final Report (Group 2)



Antoun, Fadi[1], Fishman, David[2], Haughton, Laurie[3], Jung, Myunghee[4], and McLennan, Dan[5]

Emails: [1]fadi.antoun.95@gmail.com, [2]davjfish@gmail.com, [3]eponapr@gmail.com, [4]mundlejung@gmail.com, [5]DanJMcLennan@gmail.com

December 2025

1 Summary

This project explores how large volumes of emergency response data can be transformed from tabular CSV logs into reliable operational intelligence for public safety planner and other stakeholders. Emergency call information is often fragmented across jurisdictions and systems, making it difficult to identify trends, coordinate resources, and support data-driven decision-making. To address this, the project evaluates approaches for structuring, aggregating, and disseminating emergency call data through a relational database, a REST-based application interface, and a mock emergency response dashboard.

A key focus of the work is the design and implementation of a SQL database schema capable of storing emergency call data efficiently and with strong data integrity guarantees. Using a structured dataset from Montgomery County, PA (containing over 600,000 records), the project develops a normalized five-table schema that incorporates foreign keys, uniqueness constraints, and targeted indexing. This schema reduces storage requirements relative to the raw CSV and provides a foundation compatible with modern business intelligence tools and application frameworks.

Building on the database layer, the project implements a mock dashboard application using Django and the Django REST Framework. The application provides data access through paginated and filterable API endpoints and integrates frontend components such as Vue.js, Leaflet, and Charts.js to deliver an interactive user experience. Example features include a map for visualizing call locations, a summary page for real-time aggregations, and an administration portal for uploading CSV data. The project emphasizes the need for paginated queries and aggregation endpoints to ensure responsiveness when dealing with hundreds of thousands of records.

The project concludes with recommendations for production-scale adoption. Future systems should integrate asynchronous task processing (e.g., Celery with Redis) to avoid long-running import operations on the main application thread. Additionally, the proof-of-concept SQLite backend should be replaced with a more robust system such as PostgreSQL, ideally with read replicas and Django Database Routers to support high-volume analytical queries. Overall, the project demonstrates a complete end-to-end pipeline for transforming emergency call logs into structured, analyzable, and operationally useful data.