Project 2 - Urban Development & Public Sentiment Analytics

Problem Statement: City planners and municipal governments often rely on outdated census data and anecdotal complaints to allocate resources for public services like transport, sanitation, and infrastructure maintenance. This leads to inefficient spending and a disconnect from real-time citizen needs.

Core concept —

Create a live urban-analytics system that fuses city service requests, transit telemetry, and social-media sentiment into geospatial KPIs and prioritized "need" scores so city planners can see problem hotspots, understand public opinion, and allocate resources quickly and transparently.

Why this matters

1. **Evidence-driven resource allocation** — focus crews where incidents + negative sentiment converge.
2. **Faster situational awareness** — real-time feeds reveal emerging problems before complaints flood inboxes.
3. **Transparency & trust** — public-facing summaries show the city is listening and acting.

High-level architecture (layers)

1. **Ingest layer** — poll 311 APIs, GTFS/GTFS-RT or transit APIs, scrape social media (keyword filters).
2. **Raw storage** — append-only files (data lake) or a "landing" schema.
3. **Processing & enrichment** — cleaning, geocoding, spatial joins, sentiment scoring, demographic join (Postgres/PostGIS or Spark).
4. **Analytical store** — cleaned normalized tables + materialized aggregates (Postgres/PostGIS, Azure SQL, or parquet tables).
5. **BI & Visualization** — Power BI with Azure Maps / ArcGIS layer, Key Influencers & AI visuals.
6. **Alerts & Ops** — scheduled jobs, dashboard alerts, exportable reports.

Core data model

- service_requests (id, created_at, resolved_at, category, lat, lon, neighborhood_id, text, status)
- transit_events (event_time, route_id, vehicle_id, delay, stop_id, lat, lon, neighborhood_id)

- social_posts (post_id, created_at, text, lat, lon, neighborhood_id, sentiment_score, sentiment_label, confidence)
- neighborhoods (neighborhood_id, name, polygon, population, socioeconomics)
- aggregates (daily/hourly precomputed KPIs by neighborhood)

Phase 1 — Basic ingestion & geospatial base (Week 1)

1. **Fetch 311 sample** — pull one week of data and save raw JSON/CSV.
2. **Neighborhood polygons** — import shapefile → GeoJSON. Confirm CRS = EPSG:4326.
3. **Clean 311** — timestamps to UTC, unify category names, remove duplicates. Geocode missing coords if needed.
4. **Power BI mock** — import neighborhoods + sample 311, make base map heatmap.

Tools/snippets

- Geo convert (Python/geopandas):

```
import geopandas as gpd

nb = gpd.read_file("neighborhoods.shp").to_crs(epsg=4326)

nb.to_file("neighborhoods.geojson", driver="GeoJSON")
```

Deliverable: Power BI file: map + 311 heatmap.

Phase 2 — Social ingestion & sentiment (Week 2)

1. **Collect posts** — implement simple ingestion (keyword filters). Store raw text + metadata.
2. **Sentiment scoring** — use Azure Cognitive Services Text Analytics or Power BI Cognitive connector. Save sentiment_score, label, confidence.
3. **Geolocate posts** — assign neighborhood with spatial join; mark low-confidence geolocations.
4. **Overlay layer** — in Power BI, add neighborhood-level average sentiment choropleth.

Azure sentiment quick call (concept):

```
// call Text Analytics v3.1: send documents and store response (score + label)
```

Deliverable: social_posts table with sentiment + Power BI layer showing sentiment by neighborhood.

Phase 3 — Transit data & performance metrics (Week 2–3)

1. **Ingest GTFS & GTFS-RT** — parse schedules and realtime positions/delays.
2. **Compute KPIs** — ridership (counts), avg delay, % on-time per route/stop/neighborhood.
3. **Visuals** — Transit Performance Dashboard: ridership trends, delay heatmap, route-level drilldowns.

Deliverable: Transit dashboard with drillthrough by route and hot-route detection.

Phase 4 — Correlation, scoring & resource allocation (Week 3)

1. ***Aggregate by neighborhood*** — request counts, negative sentiment pct, avg resolution time, transit reliability.
2. **Normalize and weight** → produce NeedScore (example weights: Requests 50%, Negative Sentiment 30%, Resolution Time 20%). Use min-max or z-score.
3. ***Gap analysis*** — join allocated resources (staff/ budget) if available → produce ResourceGap = Allocated - NeedScore.
4. **Key Influencers** — in Power BI use Key Influencers to find drivers of negative sentiment (e.g., delays, lighting, category).

SQL aggregation concept (PostGIS):

```
-- spatial join points to polygons and aggregate

SELECT n.id, COUNT(sr.*) AS requests, AVG(sp.sentiment_score) AS avg_sent

FROM neighborhoods n

LEFT JOIN service_requests sr ON ST_Contains(n.geom,
ST_SetSRID(ST_Point(sr.lon,sr.lat),4326))

LEFT JOIN social_posts sp ON ST_Contains(n.geom,
ST_SetSRID(ST_Point(sp.lon,sp.lat),4326))

GROUP BY n.id;
```

Deliverable: Ranked neighborhoods by NeedScore and gap list.

Phase 5 — Optimization, UX, deploy & public summary (Week 4)

1. **Performance** — materialize daily/hourly aggregates; use partitions by date; for heavy maps use imported datasets in Power BI.
2. **Polish visuals** — consistent colors, clear tooltips, bookmark-guided narratives (policymaker view vs internal ops).
3. **Scheduling & alerts** — set dataset refresh cadence and threshold alerts (NeedScore spike).
4. **Public summary** — single-page, non-technical map + top 3 insights + actions. Aggregate only, no PII.
5. **Handoff docs** — data dictionary, ETL schedules, maintenance notes.

Deliverable: Published Power BI report (service), scheduled refresh, alert rules, docs.

Quick technical tips & gotchas

- Use **Postgres + PostGIS** for spatial joins if you want simple SQL-based geography operations.
- When posts lack geo, mark them geo_confidence=low and exclude from neighborhood choropleth unless you have reliable inference.
- For Power BI performance: import heavy aggregates and use DirectQuery only for small live tables.
- Keep raw data (landing) immutable so you can re-run enrichment without losing provenance.

Minimal viable product (MVP) — what to deliver fast (2 weeks)

1. Ingest & clean: 1 week of 311 + sample transit + 1 week of social posts (keyword-based).
2. Geospatial baseline: neighborhoods geojson + heatmap.
3. Sentiment overlay by neighborhood.
4. One Power BI dashboard page: hotspots, sentiment, top 5 neighborhoods by NeedScore.

Immediate 5-step quickstart checklist

1. Create repo and folder structure.

2. Download neighborhood shapefile and convert to neighborhoods.geojson.
3. Pull 1 week of 311 data into /data/raw/311/.
4. Run a small Python notebook to clean and geocode missing lat/lon.
5. Open Power BI, import neighborhoods + cleaned 311, make a heatmap — this proves the loop.

Another way you can think

Week 1 — Data integration, geospatial base, Power BI mockup

Goals: Ingest 311 data, load city neighborhoods shapefile, set up transit data collection, Power BI base map + heatmap mock.

Tasks

1. **Get sample 311 data**
   - Pull a week of historical 311 via API or CSV. Save raw JSON/CSV to /data/raw/311/.
2. **Load neighborhood shapefile**
   - Convert to GeoJSON or keep shapefile. Tools: QGIS or geopandas.
   - Example (python/geopandas):
   - import geopandas as gpd
   - nb = gpd.read_file("city_neighborhoods.shp")
   - nb.to_file("neighborhoods.geojson", driver="GeoJSON")
3. **Set up transit data collection**
   - If GTFS-RT: set up a small script to poll and persist. If CSV GTFS for schedules, ingest schedule and stops.
4. **Create simple ETL (Power Query / Python) to clean 311**
   - Normalize timestamps, standardize categories, geocode missing addresses (batch using GIS or Azure Maps).
5. **Power BI mockup**
   - Create a base map visual (Azure Maps visual or ArcGIS). Load neighborhoods.geojson as map layer.

- Import service_requests sample, plot points or heatmap (Power BI Map > Heat map or ArcGIS hot spot).

Deliverable: Power BI file with base map + plotted 311 points; cleaned service_requests table.

Week 2 — Sentiment pipeline + Service Request Dashboard

Goals: Build social media pipeline, apply sentiment analysis, create Service Request dashboard, overlay sentiment.

Tasks

1. **Social media ingestion**
    - Choose source(s): platform API (X/Twitter), Reddit API, public Facebook pages, Mastodon, or streaming via third-party.
    - Filter by keywords: e.g., ["pothole","subway","bus","park safety","noise"]. Store raw text + metadata.
2. **Sentiment analysis**
    - Option A (Power BI AI): Use Power BI's Cognitive Services connector or built-in Text Analytics in Power Query.
    - Option B (Azure Cognitive Services Text Analytics with Python) — snippet:
    - import requests, json, os
    - endpoint = os.environ["TEXT_ANALYTICS_ENDPOINT"]
    - key = os.environ["TEXT_ANALYTICS_KEY"]
    - url = endpoint + "/text/analytics/v3.1/sentiment"
    - docs = {"documents":[{"id":"1","language":"en","text":"This bus service is terrible"}]}
    - r = requests.post(url, headers={"Ocp-Apim-Subscription-Key":key,"Content-Type":"application/json"}, json=docs)
    - print(r.json())
    - Store sentiment_score (numeric) and sentiment_label (positive/neutral/negative).
3. **Geolocate posts**

- If geo-tag available assign neighborhood via spatial join against neighborhoods polygons. If not, try user profile location or text-based geocoding (lower confidence).

4. **Power BI Service Request Dashboard**
   - Visuals: total open requests, requests by category, avg resolution time, trend line, neighborhood heatmap.
   - Overlay: add a layer showing average sentiment per neighborhood (choropleth or bubble size).
5. **Power Query measures & sample DAX**
   - Example DAX: avg resolution time
   - AvgResolutionHours = AVERAGEX(
   -   FILTER(service_requests, NOT(ISBLANK(service_requests[resolved_at]))),
   -   DATEDIFF(service_requests[created_at], service_requests[resolved_at], HOUR)

Deliverable: Power BI file with Service Request Dashboard and sentiment overlay; social_posts table with sentiment.

Week 3 — Correlation, demographics, Transit Performance Dashboard

Goals: Correlate requests & negative sentiment with demographics, analyze resource allocation, build transit dashboard and use Key Influencers.

Tasks

1. **Join demographics**
   - Spatially join service_requests and social_posts to neighborhoods and aggregate per neighborhood (counts, negative_sentiment_pct).
   - Example SQL for aggregation:
   - SELECT n.neighborhood_id,
   -        COUNT(sr.id) AS req_count,
   -        AVG(sp.sentiment_score) AS avg_sentiment,
   -        SUM(CASE WHEN sp.sentiment_label='negative' THEN 1 ELSE 0 END)*1.0/COUNT(sp.post_id) AS neg_pct,

- n.population
- FROM neighborhoods n
- LEFT JOIN service_requests sr ON ST_Contains(n.geom, ST_SetSRID(ST_Point(sr.lon, sr.lat),4326))
- LEFT JOIN social_posts sp ON ST_Contains(n.geom, ST_SetSRID(ST_Point(sp.lon, sp.lat),4326))
- GROUP BY n.neighborhood_id, n.population;

2. **Correlation & scoring**
   - Compute a **Need Score** per neighborhood, e.g.:
   - NeedScore = normalize(req_count) * 0.6 + normalize(neg_pct) * 0.3 + normalize(1/response_time) * 0.1
   - Use z-score or min-max normalization to combine metrics.
3. **Resource Allocation vs Need**
   - Compare budgeted resources or staff assigned per neighborhood (if available) against NeedScore to flag gaps.
4. **Transit performance**
   - Create Transit Performance Dashboard with KPIs: ridership (per route), average delay, % on-time, downtime events.
   - Use Power BI Key Influencers visual to find what drives negative sentiment (e.g., long delays, lack of lighting, specific lines).
5. **Model improvements**
   - Create materialized aggregates (daily/hourly) for performance (to keep dashboards responsive).

Deliverable: Power BI: Transit Dashboard + Correlation report; NeedScore table; recommendations list.

Week 4 — Optimization, polish, public summary & deployment

Goals: Optimize queries, finalize data model, polish visuals, prepare public summary page and admin dashboard for policymakers.

Tasks

1. **Performance**

- Materialize heavy queries (precompute daily aggregates). Use partitions by date.
- In Power BI: use Import mode for heavy visuals, DirectQuery for smaller live tables.

2. **UX polish**
   - Use consistent color palette, clear legends, tooltips with actionable text, bookmarks to create guided narrative pages.
   - Create a concise public summary page (one page), non-technical, with maps, top 3 insights, actions taken/required.
3. **Export & sharing**
   - Publish to Power BI Service. Configure row-level security if needed (city departments).
   - Set up scheduled refresh (daily/hourly depending on data).
4. **Monitoring & alerts**
   - Create alerts in Power BI or in backend for thresholds (e.g., neighborhoods where NeedScore rises > X).
5. **Documentation & Handoff**
   - Provide README: ETL schedules, data dictionary, how to re-run sentiment model, and dashboard guide for non-technical users.

Deliverable: Published Power BI report, refresh schedule, final documentation, slide deck for policymakers.