Trip Data Dashboard – Technical Report

1. Problem Framing and Dataset Analysis

The dataset contains urban taxi trips with fields including `pickup_datetime`, `dropoff_datetime`, `trip_duration`, `trip_distance`, `trip_speed_kmh`, `fare_amount`, and `passenger_count`. It represents city mobility patterns.

Data Challenges:

- Missing values for `fare_per_km` and `trip_duration`.
- Outliers: zero duration, zero distance, or unrealistic speeds.
- Anomalies: inconsistent timestamps.

Assumptions:

- Non-finite or negative trips were excluded from charts and stats.
- Hourly analysis uses pickup hour; filters apply inclusive date ranges.

Unexpected Observation:
Some trips had extremely high fares for short distances. This led to including **fare and distance filters** in the dashboard for focused exploration.

2. System Architecture and Design Decisions

Architecture Diagram:

```
[Frontend: HTML/CSS/JS + Chart.js] → [Backend: Node.js/Express API] → [Data: CSV/JSON]
```

Design Choices:

- **Frontend:** Chart.js for interactive visualisation, plain JS for simplicity.

- **Backend:** Node.js/Express serves API endpoints efficiently.
- **Data:** CSV/JSON sufficient for 50,000 trips.

Trade-offs:

- Flat files simplify deployment but limit query speed for larger datasets.
- Avoided heavy frontend frameworks to maintain lightweight performance.
- Manual algorithms implemented for heatmap and top routes, emphasizing algorithmic thinking.

3. Algorithmic Logic

Top-Routes Aggregation:

- Round lat/lon to a given precision.
- Use a map with key `pickup_lat,pickup_lon,dropoff_lat,dropoff_lon` → count.
- Increment counts per trip; extract top k routes.

Pseudo-code:

```
routeCount = {}
for trip in trips:
    key = round(trip.pickup_lat,3)+","+round(trip.pickup_lon,3)+
          ","+round(trip.dropoff_lat,3)+","+round(trip.dropoff_lon,3)
    routeCount[key] = routeCount.get(key,0)+1
topRoutes = select k keys with highest counts
```

Complexity: Time O(n log k), Space O(n).
This manual approach avoids library functions like `sort_values` or `Counter`.

4. Insights and Interpretation

1. Busiest Hours:

- Counted trips per hour; top 5: 8–9 AM, 5–7 PM.
- **Interpretation:** Aligns with peak commuting times; helps optimize fleet allocation.
- *Visualization:* Bar chart.

2. Average Fare Trend:

- Daily average `fare_per_km`; higher on weekends.
- **Interpretation:** Reflects demand-driven pricing.
- *Visualization:* Line chart.

3. Trip Duration Distribution:

- Most trips last 5–15 minutes, long tail beyond 45 minutes.
- **Interpretation:** Short trips dominate urban areas; long trips indicate congestion or anomalies.
- *Visualisation:* Histogram.

5. Reflection and Future Work

Technical Challenges: Handling missing/invalid data and performance with 50,000+ trips.
Team Challenges: Coordinating chart/filter updates and agreeing on data cleaning assumptions.

Future Enhancements:

- Real-time trip data streaming.
- Export filtered trips for analysis.
- Predictive models for fare or duration.