**🧾 Technical Solution Document**

**Project:** Real-time Trip Data Processing System for NSP Bolt Ride  
**Domain:** Event-Driven Architecture | Serverless Computing | Real-time Analytics

**📘 Introduction**

**📌 Purpose**

This document describes the architecture and design of a real-time data processing system developed for NSP Bolt Ride. The system ingests trip event data (start/end), processes it using AWS-native services, and produces daily KPIs for analytics.

**📌 Scope**

The system:

* Processes streaming trip events (start and end)
* Stores and updates trip records in DynamoDB
* Aggregates daily KPIs for completed trips
* Outputs metrics to S3 as structured JSON files

**Excludes**: Real-time dashboards, ML models, frontend integrations

**📌 Definitions, Acronyms, and Abbreviations**

| **Term** | **Definition** |
| --- | --- |
| KPI | Key Performance Indicator |
| AWS | Amazon Web Services |
| Lambda | AWS serverless function |
| Kinesis | Real-time stream processing |
| DynamoDB | AWS NoSQL database |
| S3 | Amazon Simple Storage Service |

**📌 References**

* [README](https://github.com/Zukizuk/real-time-trip-processing-project/blob/main/README.md) — System overview and setup
* [PDF: Event-Driven Data Processing Using Lambda & Kinesis](https://github.com/Zukizuk/real-time-trip-processing-project/blob/main/problem/Event-Driven%20Data%20Processing%20Using%20Lambda%20%26%20Kinesis.pdf)

**📌 Overview**

This system leverages event-driven architecture and AWS serverless services to support scalable, modular, and fault-tolerant real-time analytics on ride-hailing trip data.

**Description:**  
The system uses Kinesis streams for ingestion, Lambda for processing, DynamoDB for storage, and daily KPI aggregation is persisted to S3.

**🎯 Architectural Goals and Constraints**

**Goals:**

* Modular, scalable, real-time data pipeline
* Ensure at-least-once event processing
* Accurate aggregation of completed trips only

**Constraints:**

* Trip data can arrive out of order
* System must handle unordered, late-arriving data
* Must be cost-effective (serverless-first)

**👤 Use-Case View**

**Use-Case Realizations:**

* **Trip Start Event Ingestion**
* **Trip End Event Ingestion**
* **Trip Completion Detection**
* **KPI Aggregation and Output**

**🧠 Logical View**

**Major Components:**

* Lambda (trip\_start.py, trip\_end.py, aggregate\_kpi.py)
* DynamoDB (trip table)
* Kinesis (trip\_event\_stream)
* S3 (analytics output)

**🔄 Process View**

**Concurrent Processes:**

* Independent Lambda functions for start and end event processing
* Scheduled daily Lambda for KPI aggregation

**Synchronization:**

* DynamoDB conditional writes and updates using trip\_id

**🖥️ Deployment View**

**Mapping:**

* AWS Lambda deployed per function via Terraform
* Stream and DB connections provisioned in Terraform

**💻 Implementation View**

**Development Environment:**

* Python 3.9
* Boto3 SDK
* Terraform for infrastructure

**Versioning:**

* GitHub repository with IaC and Lambda scripts
* Tagging and release pipelines can be integrated with GitHub Actions

**🗄️ Data View (Optional)**

**Schema:**

* DynamoDB Table:
  + trip\_id (PK)
  + start\_time, end\_time
  + fare\_amount, status

**Data Integrity:**

* Validation in Lambda functions
* Update logic ensures only completed trips are used for aggregation

**Scalability:**

* Designed to auto-scale via Lambda concurrency and Kinesis shard scaling

**✅ Quality**

**Key Attributes:**

| **Attribute** | **Implementation Strategy** |
| --- | --- |
| Reliability | Retry logic in Lambda, DynamoDB transactional writes |
| Maintainability | Modular codebase, IaC for infra |
| Security | IAM roles per function, no hardcoded secrets |
| Performance | Real-time with event-driven triggers |
| Usability | Structured outputs for downstream analytics |

**📎 Appendices**

**Revision History**

| **Version** | **Date** | **Author** | **Changes** |
| --- | --- | --- | --- |
| 1.0 | 2025-04-24 | Marzuk | Initial Draft |