PROJECT PROPOSAL

"StreamTide: Big Data NYC Taxi fare, Demand & System Dynamics"

Group 9 - Shristi Kumar, Vidushi Verma, Shreya Akotiya, Venkata Ramireddy

Section 1 : Abstract

The aim of this project, **StreamTide: Big Data NYC Taxi Fare, Demand & System Dynamics**, is to build a real-time analytics system for taxi demand modeling and dynamic fare optimization using the New York City Taxi dataset as a reference. Historical Parquet data will be ingested through **Apache Kafka** and processed by **Apache Spark Structured Streaming**. To handle the challenges of data **volume and velocity**, we implement **Reservoir Sampling** for memory-efficient representative sampling of trips and **Bloom Filters** for fast duplicate detection across billions of records. These algorithms, with a scalable Big Data infrastructure, let us detect demand surges, estimate trip distributions, and also lets us know insights into fare dynamics and system behavior. In the end , we are building a system showing how NYC taxis can adopt dynamic pricing strategies to balance supply and demand more efficiently.
Data source: NYC Taxi & Limousine Commission (TLC) website.

Section 2: Motivation

New York City, as the most populated city in the United States and a global hub for tourism and finance, has one of the busiest urban mobility ecosystems in the world. With high parking costs and heavy congestion, millions of residents and visitors rely on taxis and public transportation. Ride-sharing platforms such as Uber and Lyft already employ **dynamic surge pricing** to balance supply and demand in real time, but traditional NYC taxis still operate under **static fare structures**, leading to inefficiencies during peak and off-peak periods.

Our project, **StreamTide**, is motivated by this **business and societal challenge** of modernizing the legacy taxi system through **Big Data streaming analytics**. By applying **Reservoir Sampling** to provide statistically valid demand signals and **Bloom Filters** to ensure accurate, duplicate-free surge detection, we design a **scalable pipeline** that continuously monitors demand and fare dynamics across the city. This approach not only enables fairer and more adaptive fare strategies but also provides a broader view of system-level behavior, supporting both **driver profitability and commuter accessibility**.

Section 3: Literature Survey

Researchers have been diving deep into dynamic pricing and demand forecasting for ride services, especially for platforms like Uber and Lyft. Xu, Chen, and Zhou (2019) took a close look at how surge pricing works in mobility-on-demand services—they found it does help balance supply and demand, but it also creates some real fairness issues that passengers aren't happy about [1]. Meanwhile, Liu, Ferreira, and Ratti (2017) studied ride patterns across New York City and made a compelling case that the city desperately needs systems that can actually adapt when demand suddenly shifts [2].

What's interesting is that most of this research focuses on ride-sharing apps, while traditional NYC taxis are still stuck with static pricing from decades ago. That's exactly what motivates our project—instead of just building another predictive model, we're creating a real-time streaming system that can actually see what's happening right now with traditional yellow cabs.

On the technical side, Babcock et al. (2002) laid out the foundational principles for handling massive streaming data systems, giving us the architectural blueprint for building something that won't crash under the weight of billions of records. Their work directly guides how we're implementing Reservoir Sampling to get representative demand snapshots and Bloom Filters to catch duplicate entries without burning through memory [3].

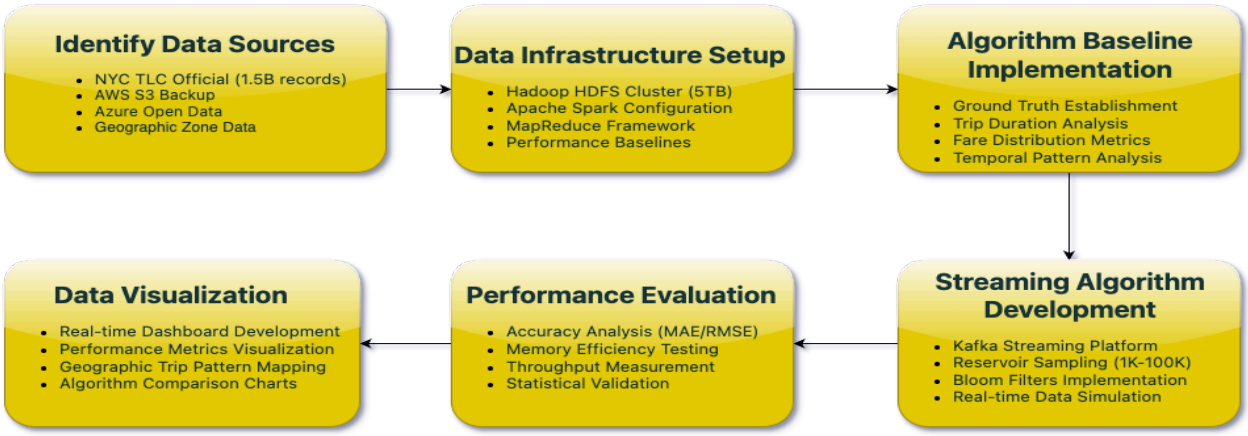
Chen, Jin, and Zhang (2020) also showed us how to optimize storage and processing for huge datasets using Parquet with deduplication—which is perfect for handling all those TLC trip records. When you put it all together, these studies give us both the transportation economics background and the computer science tools we need to build something that actually works at NYC's scale [4].

References :

1. W. Zhang, D. Kumar, and S. V. Ukkusuri, “Exploring the dynamics of surge pricing in mobility-on-demand taxi services,” 2021 IEEE International Conference on Big Data (Big Data), pp. 1375–1380, Dec. 2017, doi: 10.1109/bigdata.2017.8258070. Available: https://doi.org/10.1109/bigdata.2017.8258070
2. D. Correa and C. Moyano, “Analysis & Prediction of New York City Taxi and Uber Demands,” Journal of Applied Research and Technology, vol. 21, no. 5, pp. 886–898, Oct. 2023, doi: 10.22201/icat.24486736e.2023.21.5.2074. Available: https://doi.org/10.22201/icat.24486736e.2023.21.5.2074
3. Z. Yuan, Y. Chen, Y. Jia, and S. Yang, “Counting Evolving data stream based on hierarchical counting bloom filter,” 2017 IEEE International Conference on Big Data (Big Data), vol. 30, pp. 290–294, Dec. 2008, doi: 10.1109/cis.2008.216. Available: https://doi.org/10.1109/cis.2008.216
4. L. Kuhring and Z. Istvan, “Storing Parquet Tile by Tile: Application-Aware Storage with Deduplication,” 2019 29th International Conference on Field Programmable Logic and Applications (FPL), pp. 415–416, Sep. 2019, doi: 10.1109/fpl.2019.00073. Available: https://doi.org/10.1109/fpl.2019.00073.

Section 4: Methodology

Our approach involves designing a **scalable big data analytics system** focused on showcasing **memory-efficient streaming algorithms** using the **NYC taxi dataset**. The project contains every stage, including data collection, distributed processing, algorithm implementation, and a thorough performance evaluation, to ensure robust and efficient analytics at scale.



Identify Data Sources:

We begin by identifying and acquiring comprehensive data sources to ensure robust analysis coverage:

Primary Data Sources:

- **NYC TLC Official Dataset**: 1.5+ billion taxi trip records (2009-2025) in optimized Parquet format
- **AWS S3 Backup Repository:** Complete historical backup accessible via S3 API (s3://nyc-tlc/)
- **Azure Open Data Platform:** Integrated datasets with SDK access for time-range filtering
- **Geographic Zone Data:** NYC taxi zone lookup files for spatial analysis and mapping

We will also build a data **validation Strategy** to ensure primary data integrity via automated checks, cross-source consistency (NYC TLC/AWS/Azure), temporal alignment, and documented lineage for reproducibility.

Data Infrastructure Setup:

Our system implements a **hybrid batch-streaming architecture** optimized for big data volume and velocity challenges

Distributed Storage Architecture:

- **Hadoop HDFS Cluster:** Multi-node deployment with 5+ TB capacity and 3x replication factor
- **Apache Spark:** Optimized for Parquet processing with DataFrame API and Catalyst optimizer
- **MapReduce Framework:** Baseline implementation for ground truth establishment
- **Performance Baselines:** Initial metrics collection for system optimization

For **infrastructure optimization**, we'll tune memory for billion-records, optimize inter-node networking, **add fault tolerance** with auto-failover, and allocate resources to **maximize cluster utilization**.

Algorithm Baseline Implementation:

We will establish the ground truth metrics essential for streaming algorithm validation:

Ground Truth Development:

- **Trip Duration Analysis:** Statistical distribution computation across all pickup/dropoff
- **Fare Distribution Metrics:** Complete fare analysis including surge pricing and tip calculations
- **Temporal Pattern Analysis**: Hourly, daily, weekly, and seasonal aggregation patterns
- **Spatial Analytics:** Zone-based demand analysis using geographic clustering techniques

Streaming Algorithm Development:

We'll use Kafka to replay historical taxi trips as realistic live streams and run reservoir sampling and Bloom filters in parallel to test algorithms at scale.

Kafka Streaming Platform:

- We run a **multi-broker** & **high availability** Kafka cluster that streams real time, state-preserved data from history.
- **Topics** will be partitioned by time and location, and tuned producers/consumers run algorithms in parallel.

Algorithms:

Reservoir Sampling: Configurable sample sizes (1K, 10K, 100K) with temporal stratification

- **Sample Sizes:** 1K, 10K, 100K, 1M records for trade-off analysis
- **Temporal Stratification:** Handle time-based biases in taxi trip patterns
- **Spatial Distribution:** Account for geographic clustering in NYC boroughs
- **Real-time Accuracy Monitoring:** Compare sample statistics against known population parameters

Bloom Filters: Optimized hash functions with tunable false positive rates for duplicate detection

- **Bit Array Sizing:** Varying sizes (1MB, 10MB, 100MB) for false positive rate analysis
- **Hash Functions**: Multiple algorithms (MurmurHash, CityHash, SHA-based) performance comparison
- **Duplicate Detection**: Trip ID validation across multiple data sources
- **Dynamic Parameter Tuning**: Automatic adjustment based on data velocity and accuracy requirements

Real-time Validation: Continuous accuracy monitoring against **MapReduce** ground truth

Performance Evaluation And Data Visualization:

performance analysis validates our streaming algorithm implementations and Dashboards will help in data visualization

Accuracy Analysis & Statistical Validation:

- **Error Metrics:** MAE, RMSE calculations with 95% confidence intervals
- **Temporal Analysis:** Accuracy trends across different time periods and seasonal patterns
- **Edge Case Evaluation:** Performance during data spikes, rush hours, and special events

Memory Efficiency & Throughput Testing:

We'll profile peak memory and allocations, scale from 1M to 1B+ records, measure throughput and end-to-end latency, and surface bottlenecks across CPU, memory, network, and I/O.

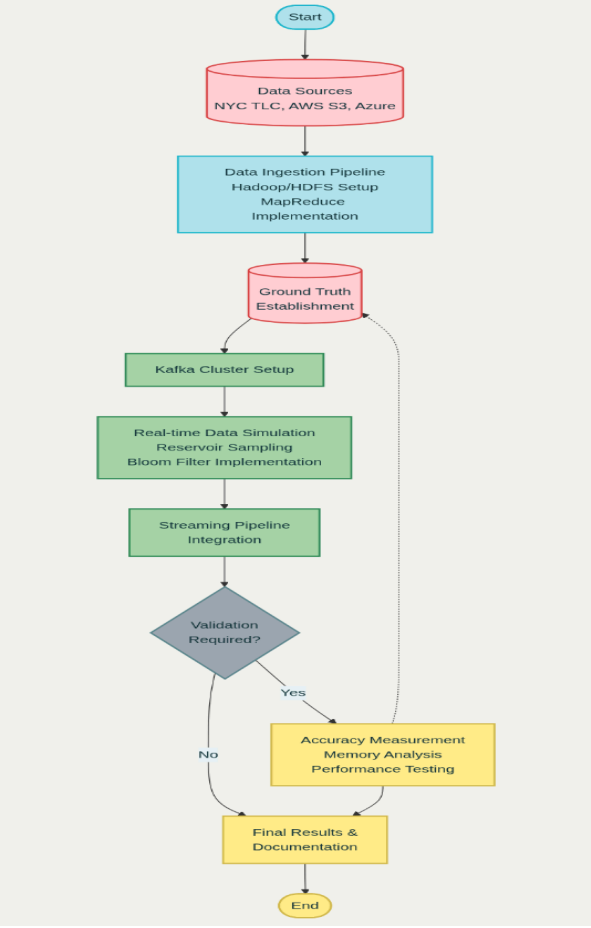
Data Visualization

Our Real-time Analytics Dashboard will include:

- **Performance Monitoring:** Live tracking of memory usage, throughput, and processing latency
- **Algorithm Accuracy Tracking:** Dynamic comparison between streaming and ground truth
- **Geographic Visualization**: Interactive NYC maps showing taxi trip patterns and route density
- **System Health Monitoring**: Cluster resource utilization with alerting capabilities

Visualization Technology: Grafana/Tableau for interactive dashboards and real-time charts.

Overall workflow:



Section 5: Deliverables

1. Deliverable 1: Data ingestion pipeline + validation results.
2. Deliverable 2: Spark/Kafka/HDFS cluster setup documentation.
3. Deliverable 3: Reservoir Sampling + Bloom Filters implemented.
4. Deliverable 4: MapReduce ground-truth results.
5. Deliverable 5: Performance & accuracy evaluation tables
6. Deliverable 6: Real-time analytics dashboards
7. Deliverable 7: Final IEEE-format report (LaTeX), presentation, demos

Section 6: Team members and their roles

1. **Shristi Kumar :** Data acquisition and validation
1. **Vidushi Verma:** Infrastructure setup and streaming pipelines
2. **Shreya Akotiya :** Algorithm implementation (Reservoir Sampling and Bloom Filters)
3. **Venkata Ramireddy :** Lead evaluation, visualization, and final presentation deliverables.

Section 7: Relevance to the course

This project illustrates how distributed frameworks, streaming algorithms, and analytical models are used to transform large-scale data into understandable insights. The challenges of scale, velocity, and complexity demonstrate how the theoretical concepts from the course translate into a real-life application.

Concepts and Methods (CLO 1 & CLO3):
The project applied core big data algorithms such as **reservoir sampling and Bloom filters** but also adopted frameworks such as **Kafka, Spark, and HDFS** to showcase his strong command of fundamental concepts and practices put forward during the course.

Tools and Techniques (CLO 3):
Hadoop HDFS and Apache Spark were employed directly in the system for distributed storage and computation, thus demonstrating mastery of the same tools focused on in lectures and assignments.

Problem Design and Implementation (CLO 4):
The project was done end-to-end-from ingestion and cleaning of raw data to streaming algorithm development to visualization and evaluation. This full-cycle design is in line with the course's intent to prepare students to design and implement complete big data solutions.

Emerging Big Data Systems (CLO 8):
By showcasing that the state-of-the-art big data systems could abstract value from raw/high volumes of messy datasets such as the NYC taxi records, the project indicated the development of extracting

value from high rates of new incoming data via stream processing frameworks and memory-efficient algorithms.

Analytical Solutions (CLO 9):
The project successfully illustrated the design of analytical solutions across big data, and streaming systems. By integrating large-scale storage and computation through HDFS and Spark, establishing ground-truth baselines through MapReduce, and utilizing Kafka pipelines for real-time processing, the system would be able to interleave batch and streaming workflows for longer-term analytics and real-time insight applications.

Section 8: Technical Difficulty

Data Challenges

- Designing ingestion pipelines to process over 1.5 billion taxi records (5TB+) and avoid bottlenecks.
- Cleaning and schema alignment became the most critical challenges as the data collected from different sources: NYC TLC, AWS S3, and Azure were inconsistent, with missing fields and noisy geospatial coordinates.

Distributed Infrastructure Tuning

- Setting up **HDFS clusters and Spark jobs** introduced difficulties in partitioning, shuffle optimization, and memory utilization.
- Establishing **reliable performance baselines was complex,** as throughput and latency fluctuated under varying workloads and node failures.

Algorithmic Complexities

- The implementation of the **reservoir sampling and Bloom filter** demanded a trade-off between memory efficiency, computational speed, and statistical accuracy.
- Building the **ground truth baseline for trip duration and fare distribution** was hampered by anomalies and missing records in the real-world data.
- Creating an iterative tuning process for a real-time data simulator that would produce a realistic high-velocity stream while not overloading system resources was indeed a challenging task.

Streaming Pipeline Integration

- **Tuning of brokers, partitions, and consumer offsets** needed precision to accomplish the desired high throughput with fault tolerance in data delivery for Kafka clusters.
- While integrating **batch and streaming** (Kafka), synchronization issues and fault recovery challenges emerged.

Section 9: Novelty

This project proves that an old traditional domestic NewYork cab infrastructure can also survive and thrive in the new Digital age. We are giving NewYork cab network a major tech upgrade by using highly reliable distributed tech stack like Kafka and Spark streaming. Our project allows a simple guide to switch to modern dynamic fares. This helps them to balance how many people want rides versus how many cabs are available. **Instead of using old static data or just guessing with simple averages**, **we use a smart trick called Reservoir Sampling to constantly update and keep up to date on demands and supply and more easily balance cab placements.**

Section 10: Impact

This project not only provides a reliable distributes fast pipeline to process huge cab data, but also provides actionable steps for ***NewYork cab industry to run efficiently*****,** which also helps city as well as residents or tourists. Our system doesn't just focus on busy or tourist friendly areas, but its advanced location-aware sampling will take care of all city areas irrespective of traffic or demand. Thus all residents benefited from it. This is going to be a very ***cost effective system for the city or cab industry to upgrade to without advanced memory and compute efficient algorithms and data structures***. We are using Bloom filters which help us to check for duplicate data points without having to utilize too much memory or compute. ***This doesn't require cab service or the city to spend massive amounts of money*** like Uber or Lyft. Sophisticated structured streaming systems make sure that they handle billions of data events with ***high fault tolerance allowing the system to recover easily*** from failures and other unknown issues. Also Apache kafka gives us smooth **real-time flowing data in distributed fashion** for spark to process quickly. **T*his also helps to plan for big events in the city or special dates*** where traffic patterns go to all together different levels.

Section 11: Heilmeier Catechis

1. What are you trying to do?(Articulate your objectives clearly, without using technical jargon.)

- We are trying to build a system that tracks **real-time demand** so NYC taxis can adjust prices fairly and be available for people when and where it is needed,
- It helps drivers and companies make smarter decisions on based on cost , demand supply surge

2. How is it done today, and what are the limitations of current practice? (Explain the existing methods and their shortcomings)

NYC's traditional yellow cabs are using the same pricing model that is fixed rates that do not change on demand and supply surge. There is real time data about rides where they are actually needed so drivers end up moving around common spots like airports and tourist areas while leaving entire neighborhoods underserved.

3. What is new in your approach, and why do you think it will be successful?(Highlight the uniqueness of your solution and its potential advantages.)

Our approach uses **real-time streaming** of taxi trip data instead of slow monthly reports. By replaying **historical TLC** records as live streams with Kafka and processing them with Spark, we can create a real-time, city-wide view of demand and taxi operations, enabling faster and smarter responses.

4. Who cares? (Identify the stakeholders who will benefit from your work.)

- **NewYork cab and city planners:** optimizing their cab network along with providing high quality services as well as making a profit.
- **Passengers:** Residents or Tourists get more realistic prices for their journey as well as more easily available cabs. And more options.
- **Data Engineers and Scientists:** They can learn and/or help improve a fully functional big data problem in an efficient manner.

5. If you're successful, what difference will it make? (Describe the real-world impact and significance of your project.)

Smart data driven decisions for cabs or city planners, better availability, more profit.All neighborhoods get services available, not just top busy areas as well as realistic prices for people.It will prove that legacy traditional systems can be efficiently upgraded to data driven new age.

6. What are the risks? (Discuss the potential challenges, uncertainties, or obstacles.)

To make sure that smart efficient data structures like bloom filter and reservoir sampling runs perfectly with limited false positives as we are dealing with very big data.With the super fast moving digital age, we have to keep our infrastructure components like kafka, spark, NOSQL database upto date and secure also.

7. How much will it cost? (Estimate the financial, resource, and time investments required.)

Costs include (AWS, Google Cloud, or on-premise), software tools (mostly open-source but some may need subscriptions).The total depends on infrastructure and project size.

8.How long will it take? (Provide a realistic timeline for development and implementation.)

The project will take about 3-5 weeks:

- **Phase 1** (1 week) for data setup and cleaning,
- **Phase 2** (1-2 weeks) to build the core algorithm and streaming,

- **Phase 3** (1-2 weeks) for dashboard development and optimization.Here is a shortened version of

9. What are the midterm and final “exams” to check for success?

Midterm: Stream & process taxi data, validate Reservoir Sampling & Bloom Filters.

Final: Detect demand surges, prototype pricing logic, deliver report & real-time dashboards.