# Operational Efficiency and Automation

## Use Case: Reducing Dasher Wait Times

**Purpose:**

To minimize delivery driver wait times by accurately predicting order preparation and delivery times.

**Data Sources:**

1. **Order Data:** Order timestamps, items ordered, restaurant details.
2. **Historical Wait Times:** Past data on driver wait times at various restaurants.
3. **Restaurant Preparation Times:** Average and variance in preparation times.
4. **Traffic Conditions:** Real-time and historical traffic data.
5. **Real-Time Location Data:** GPS coordinates of drivers and restaurants.

**Feature Engineering:**

1. **Temporal Features:** Time of day, day of week, holidays, peak hours.
2. **Restaurant Features:** Average preparation time, menu complexity, historical performance.
3. **Order Features:** Number of items, item preparation complexity.
4. **Traffic Features:** Current traffic conditions, historical traffic patterns.
5. **Driver Features:** Historical wait times, experience level, distance to restaurant.

**Model Choice:**

1. **Regression Models:**
   - **Linear Regression:** For initial baseline predictions.
2. **Machine Learning Models:**
   - **Gradient Boosting Machines (GBMs):** XGBoost, LightGBM for capturing non-linear relationships.
   - **Random Forests:** For ensemble-based predictions with feature importance insights.
3. **Deep Learning Models:**
   - **LSTMs:** For capturing temporal dependencies in sequential data.
   - **GRUs:** Similar to LSTMs but more computationally efficient.
4. **Ensemble Methods:**
   - **Combining Models:** Use ensemble methods to aggregate predictions from multiple models for improved accuracy.

**System Design:**

1. **Data Pipeline:**
    - **Ingestion:** Real-time data ingestion using Apache Kafka or AWS Kinesis.
    - **Preprocessing:** Data cleaning, normalization, and feature generation using Apache Spark or AWS Glue.
    - **Storage:** Use a data lake (e.g., AWS S3) for raw data and a data warehouse (e.g., Amazon Redshift) for structured data.
2. **Feature Store:**
    - **Management:** Centralized feature store (e.g., Feast) for consistent feature engineering and retrieval.
3. **Model Training:**
    - **Batch Processing:** Use distributed computing frameworks (e.g., Apache Spark) for scalable training.
    - **Cross-Validation:** K-fold cross-validation to ensure model robustness.
    - **Hyperparameter Tuning:** Bayesian optimization (e.g., Hyperopt, Optuna) for efficient hyperparameter search.
4. **Real-Time Inference:**
    - **Deployment:** Deploy models as microservices using Docker and Kubernetes for scalability.
    - **Inference Engine:** Use frameworks like TensorFlow Serving or NVIDIA Triton Inference Server for low-latency predictions.

**Operational Integration:**

1. **Integration:** Use predictions to optimize driver dispatch, route planning, and delivery times.

**Scalability:**

1. **Cloud Infrastructure:** AWS EC2 for compute, Lambda for serverless functions, S3 for storage.
2. **Distributed Processing:** Kubernetes for orchestrating containerized applications and managing distributed workloads.

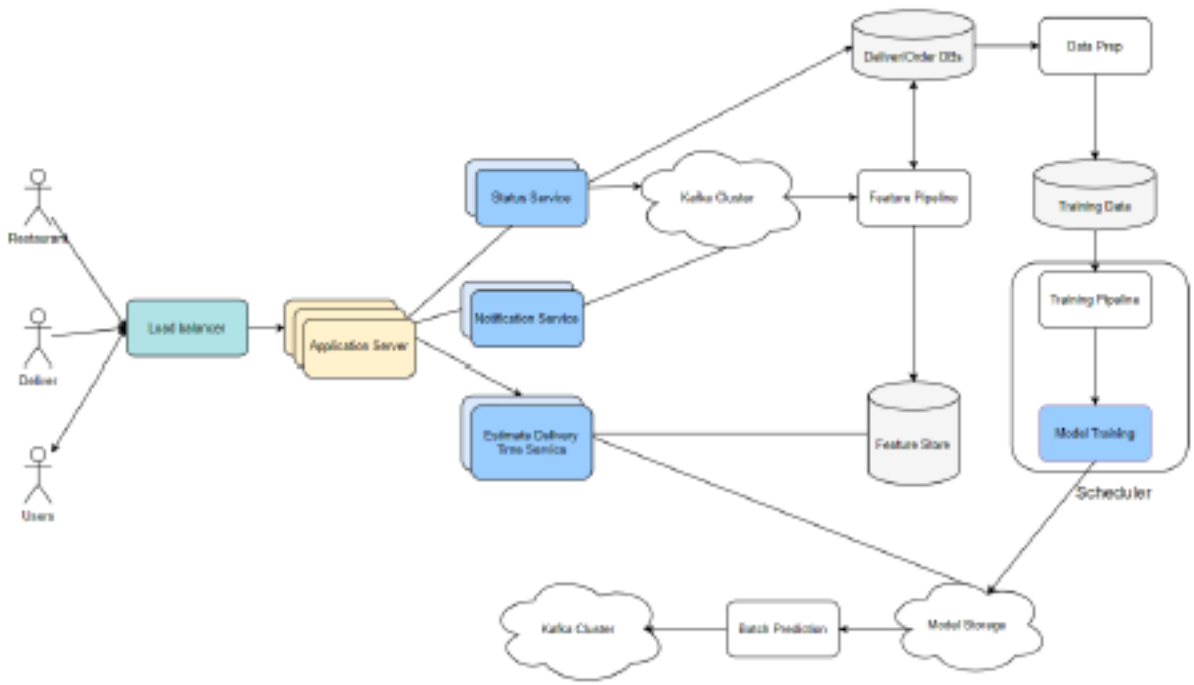
**Monitoring and Feedback Loop:**

1. **Monitoring:** Real-time monitoring of model performance and system health using Prometheus and Grafana.
2. **Feedback Loop:** Continuous learning pipelines to retrain models with new data and adapt to changing conditions.

**Evaluation Metrics:**

1. **Mean Absolute Error (MAE):** Measure of prediction accuracy.
2. **Root Mean Squared Error (RMSE):** Evaluate the magnitude of prediction errors.
3. **Mean Absolute Percentage Error (MAPE):** Understand relative accuracy.

4. **Business Impact Metrics:** Reduction in wait times, delivery efficiency, customer satisfaction.
5. **Latency:** Ensure real-time prediction capabilities within acceptable limits.




Food Delivery System Design at scale