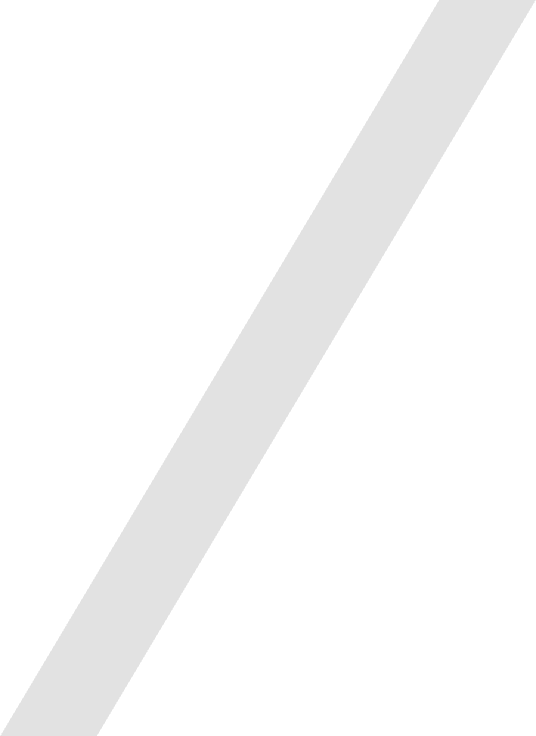
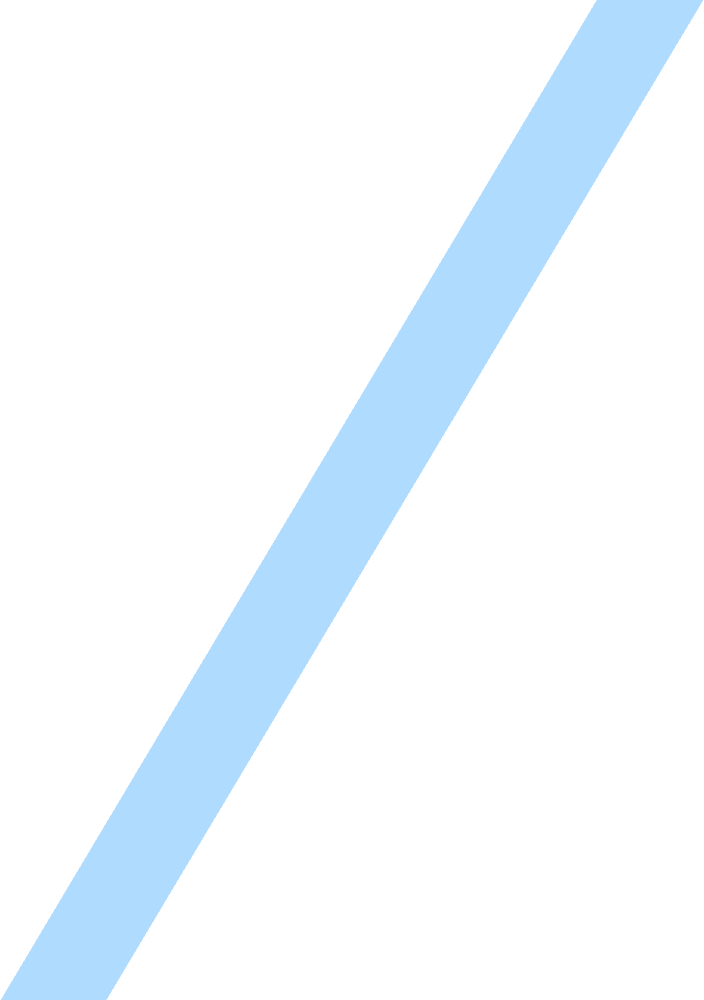
|  |
| --- |
| TECHNICAL REPORT TEMPLATE |

|  |
| --- |
| Electrical & Computer Engineering & Computer Science (ECECS) |

|  |  |
| --- | --- |
| SEMESTER |  |



Contents

[Project Name 2](#_Toc96341549)

[Executive Summary 2](#_Toc96341550)

[Technical Report 3](#_Toc96341551)

[Highlights of Project 3](file:///C:\Users\ardia\Desktop\TEACHING\SPRING%2022\DSCI6002%20Data%20Science\Projects\Technical%20report%20Template.docx#_Toc96341552)

[Submitted on: 3](file:///C:\Users\ardia\Desktop\TEACHING\SPRING%2022\DSCI6002%20Data%20Science\Projects\Technical%20report%20Template.docx#_Toc96341553)

[Abstract 4](#_Toc96341554)

[Methodology 5](#_Toc96341555)

[Results Section 5](#_Toc96341556)

[Discussion 6](#_Toc96341557)

[Conclusion 6](#_Toc96341558)

Project Title: Real-Time Ride-Sharing Platform

Submitted By: Group-2  
Course: Data Engineering  
Instructor: Dr. Sula

Technologies Used

* Django: Backend API and business logic
* Next.js: Interactive user interface
* PostgreSQL: User and ride data storage
* Apache Kafka: Real-time event streaming (Pub/Sub)
* ML Models: Random Forests (driver allocation) & Regression (fare/ETA prediction)
* Mailgun: Notifications and alerts
* AWS CloudWatch: Performance monitoring and logging

### ****Project Overview****

A cloud-based ride-sharing platform enabling real-time ride tracking, dynamic fare estimation, and efficient driver allocation. It integrates a robust backend with predictive ML models and an interactive frontend to deliver a smooth, real-time ride experience for users and drivers.

|  |
| --- |
| Project Name |

|  |  |  |
| --- | --- | --- |
|  | | |
| person at a table writing in a notebook with people around | | |
| **Team Members:**  **AMRUTH KUNTAMALLA**  **SANDEEP RAJ KATIPAGALA**  **NAGASAI JAJA** | **Questions?**  Contact :  [Akunt3@unh.newhaven.edu](mailto:Akunt3@unh.newhaven.edu)  [Skati6@unh.newhaven.edu](mailto:Skati6@unh.newhaven.edu) |  |

|  |
| --- |
| Technical Report |

|  |  |
| --- | --- |
| **Cost-Effective Real-time Ride Sharing Platform** |  |
| Highlights of Project  Real-Time Tracking: Live ride tracking and event streaming with Kafka.   Smart Predictions: ML models for fare, ETA, and driver allocation.   Scalable Backend: Django on AWS EC2 for API and business logic.   Secure Data: PostgreSQL for user data and ride history.   Interactive UI: Next.js web app for riders and drivers.   Automated Alerts: Notifications via Mailgun.   Performance Monitoring: AWS CloudWatch for logs and performance.   Cloud Integration: Seamless AWS-based deployment. Submitted on:08 December 2024 |

## Abstract

**This project focuses on developing a real-time, cloud-based ride-sharing platform with intelligent fare estimation, driver allocation, and ride tracking. The platform integrates key technologies to ensure seamless user experience and operational efficiency. Django serves as the backend, hosted on AWS EC2, managing API requests, data cleaning, and ride logic. PostgreSQL stores user profiles, ride history, and payment details, while Kafka facilitates real-time event streaming. Machine Learning models using Random Forests and Regression enable fare prediction, ETA calculation, and driver allocation. Next.js powers the user-friendly frontend, while Mailgun manages notifications. AWS CloudWatch tracks system logs and performance, and dashboards offer analytics for system monitoring. The architecture is designed for scalability, responsiveness, and real-time decision-making, ensuring a smooth ride-sharing experience for both users and drivers.**

Executive Summary

The Real-Time Ride-Sharing Platform is a cloud-based, end-to-end system designed to deliver seamless ride booking, live tracking, and dynamic fare estimation for users. Leveraging advanced technologies, the platform enables efficient driver allocation, accurate fare predictions, and real-time updates.

The backend, built with Django and hosted on AWS EC2, manages API requests, user authentication, and ride logic. PostgreSQL ensures secure storage of user data, ride history, and payment details. Apache Kafka serves as a real-time event streaming system for ride tracking and driver updates. Machine Learning models, including Random Forests and Regression, power smart predictions for driver allocation, fare estimation, and ETA calculations.

The frontend, being developed using Next.js, offers a responsive, user-friendly interface for riders and drivers. Mailgun facilitates automated notifications and alerts, while AWS CloudWatch ensures system reliability through log monitoring and performance tracking.

This platform integrates all these components into the AWS cloud environment to provide a scalable, real-time, and user-centric ride-sharing experience. By leveraging predictive analytics and real-time data processing, the platform enhances customer satisfaction, operational efficiency, and decision-making.

Introductory Section

UniRides is a real-time data streaming platform designed to optimize ride-sharing services for university students, in general, for anyone.

Our primary focus is to provide a platform that meets the transportation needs of students while keeping costs low and ensuring the platform remains sustainable.

Review of available research

The Real-Time Ride-Sharing Platform is built using research-driven insights from real-time processing, predictive analytics, and cloud-native development. Key highlights include:

1. Event-Driven Systems (Kafka): Research supports Kafka's Pub/Sub model for real-time ride tracking, driver updates, and event streaming.
2. Backend Development (Django on AWS EC2): Django’s ORM and API support streamline backend development, while AWS EC2 ensures scalable and secure deployment.
3. Data Storage (PostgreSQL): PostgreSQL is widely used for secure relational data storage, handling user data, ride history, and payments.
4. Predictive Analytics (ML Models): Random Forests predict driver allocation and detect fraud, while Regression models forecast fare and ETA.
5. Frontend (Next.js): Research supports Next.js for fast, responsive web apps, leveraging Server-Side Rendering (SSR) for better user experience.
6. Cloud Monitoring (AWS CloudWatch): CloudWatch provides system monitoring and alerting to maintain performance and minimize downtime.

## Methodology(CRISP-DM)

 Title: Real-Time Ride-Sharing Platform: Predictive Analytics for Fare, ETA, and Driver Allocation

 Business Understanding:

* Objective: Build a real-time ride-sharing platform for ride requests, fare prediction, and driver allocation.
* Goals: Provide accurate fares, reduce wait times, and enhance user experience.

 Data Understanding:

* Data Sources: Ride history from PostgreSQL and real-time event streams from Kafka.
* Key Variables: Rider info, driver availability, timestamps, location, ride distance, and fare.

 Data Preparation:

* Cleaning: Handle missing, duplicate, and inconsistent records.
* Feature Engineering: Create peak hour indicators, ride distances, and driver proximity.
* Tools: Pandas and NumPy for data processing and feature generation.

 Modeling:

* Random Forests: Classifies driver allocation and detects anomalies.
* Regression Models: Predicts ride fares and ETAs.
* Tools: Scikit-learn for model development and training.

 Evaluation:

* Metrics: MAE and RMSE for fare/ETA predictions, accuracy for driver allocation.
* Testing: Models validated using 80/20 split and cross-validation to avoid overfitting.

## 

## Results Section

Results and Data Engineering Pipeline

1. Data Engineering Pipeline

* Data Ingestion: Kafka for real-time ride requests; Postman API for testing; Webhooks for third-party data.
* Data Storage: PostgreSQL for structured data (user, ride, payment info) and AWS S3 for logs and backups.
* Data Processing: Spark (AWS EMR) for large-scale ETL, Pandas/NumPy for small-scale transformations, and Airflow for pipeline orchestration.
* Data Consumption: Next.js for the user app, Django API for API calls, and Power BI/Tableau for admin dashboards.

2. Model Deployment

* ML Models: Random Forest for driver allocation, Regression for fare/ETA prediction.
* Deployment Tools: AWS EC2 + Flask App for model hosting.

3. Data Visualization

* Tools: Matplotlib/Seaborn for illustrative graphics and Power BI/Tableau for dashboards.
* Visuals: Ride volume trends, driver allocation heatmaps, revenue analysis, and system usage analytics.

4. Deployment

* Tools: AWS EC2 for backend hosting, CloudWatch for performance monitoring, and GitHub CI/CD for code deployment.
* Process: Automated deployment pipelines ensure smooth updates, while CloudWatch tracks system health.

This pipeline enables real-time data flow, accurate predictions, and comprehensive insights for end users, drivers, and administrators.

## 

## Discussion

The Real-Time Ride-Sharing Platform successfully addresses key objectives, including fare prediction, ETA estimation, and driver allocation.

1. **Research Questions Addressed**
   * **Fare & ETA Prediction**: **Regression models** provided accurate fare and ETA estimates.
   * **Driver Allocation**: **Random Forest models** reduced rider wait times by prioritizing driver proximity.
2. **Key Insights**
   * **Real-Time Tracking**: **Kafka** enabled real-time ride updates and driver availability.
   * **Predictive Accuracy**: **ML models** delivered precise fare and ETA predictions.
   * **Operational Efficiency**: **AWS EC2 + Flask Server** and **CloudWatch** supported real-time system monitoring and updates.
3. **Limitations & Gaps**
   * **External Factors**: Traffic and weather data were not incorporated, affecting ETA accuracy.
   * **Dynamic Pricing**: Surge pricing logic is yet to be implemented.
   * **Data Quality**: Limited historical data impacted peak hour predictions.
4. **How We Filled the Gaps**
   * **Driver Allocation**: **Random Forest models** filled the gap in real-time driver assignment.
   * **Transparent Pricing**: Regression-based fare predictions enhance user trust.
5. **Future Enhancements**
   * Add **traffic/weather data** to improve ETAs.
   * Implement **dynamic surge pricing**.
   * Train models with **seasonal ride patterns** for better long-term accuracy.

## 

## Conclusion

The Real-Time Ride-Sharing Platform successfully delivers real-time tracking, almost accurate fare predictions, and optimal driver allocation. Leveraging Django, Kafka, PostgreSQL, and AWS services, the platform provides a scalable, cloud-based solution for ride-sharing.

Key Achievements:

* Accurate Predictions: Regression models predict fares and ETAs, improving user transparency.
* Driver Allocation: Random Forests ensure faster driver assignment and reduced wait times.
* Scalability: AWS EC2, Flask Application, and CloudWatch enable seamless scaling, model updates, and performance monitoring.

Future Enhancements:

* Traffic/Weather Data: Improve ETA accuracy.
* Surge Pricing: Implement dynamic pricing during peak hours.
* Seasonal Data: Train models to recognize long-term trends.
* Studies on event-driven systems and predictive modeling using Random Forests and Regression.

## Acknowledgement

Special thanks to Amruth for AWS integration and our instructors and mentors for guidance and support.

## References

* Official docs for Django, PostgreSQL, AWS EC2 + Flask applicatio, and CloudWatch.