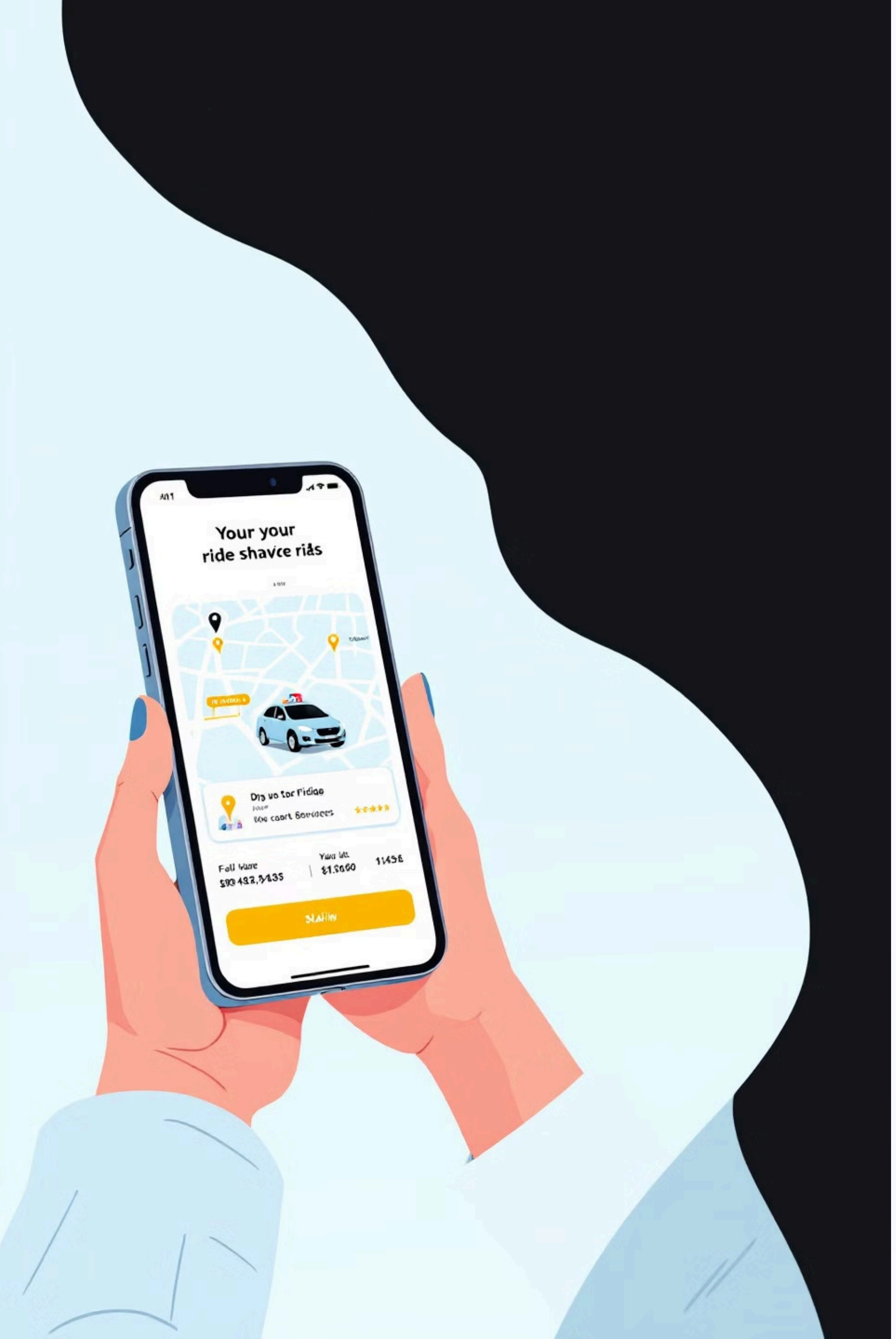
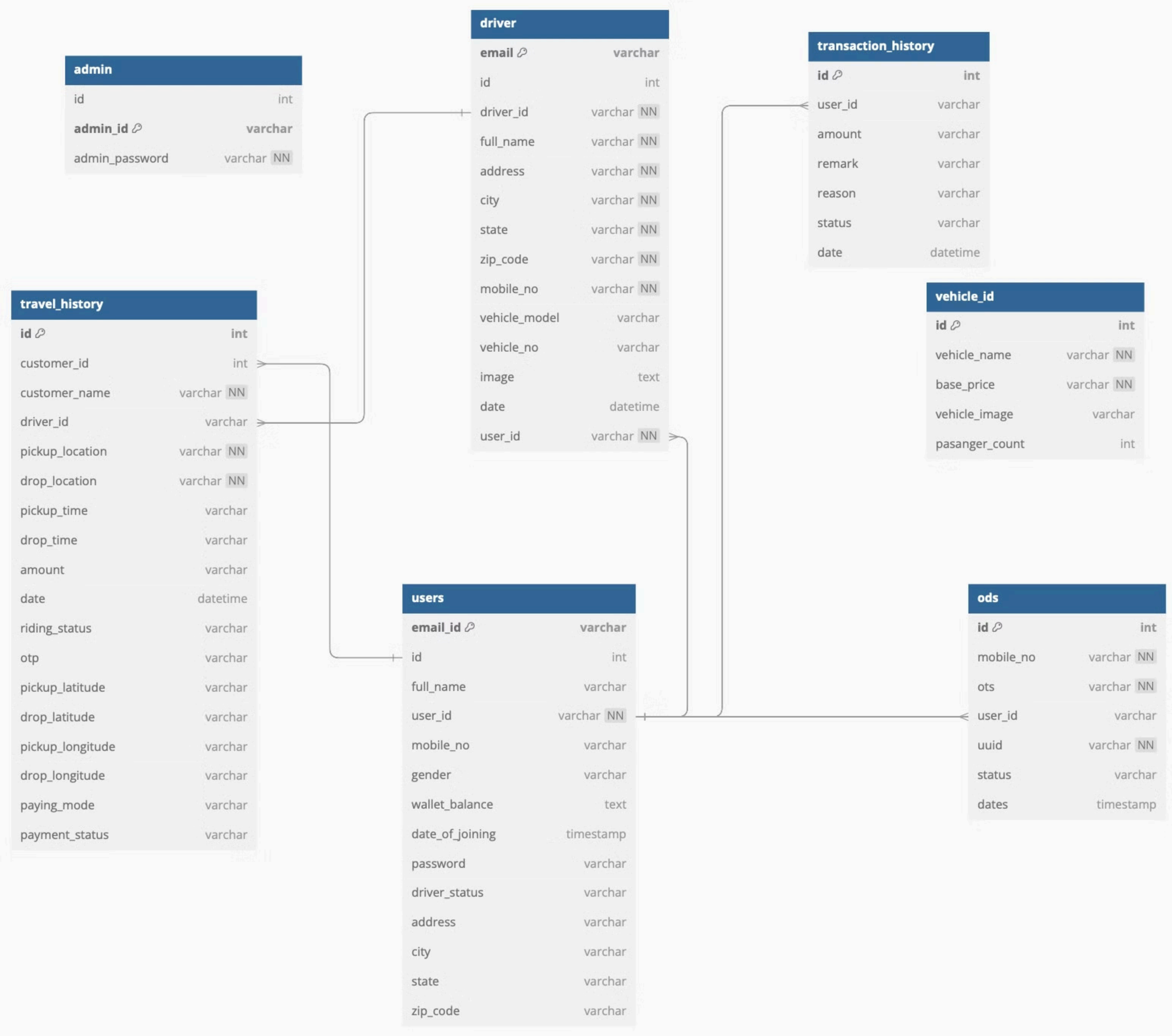


Final Project - DATA 236

Group-1 (Members):

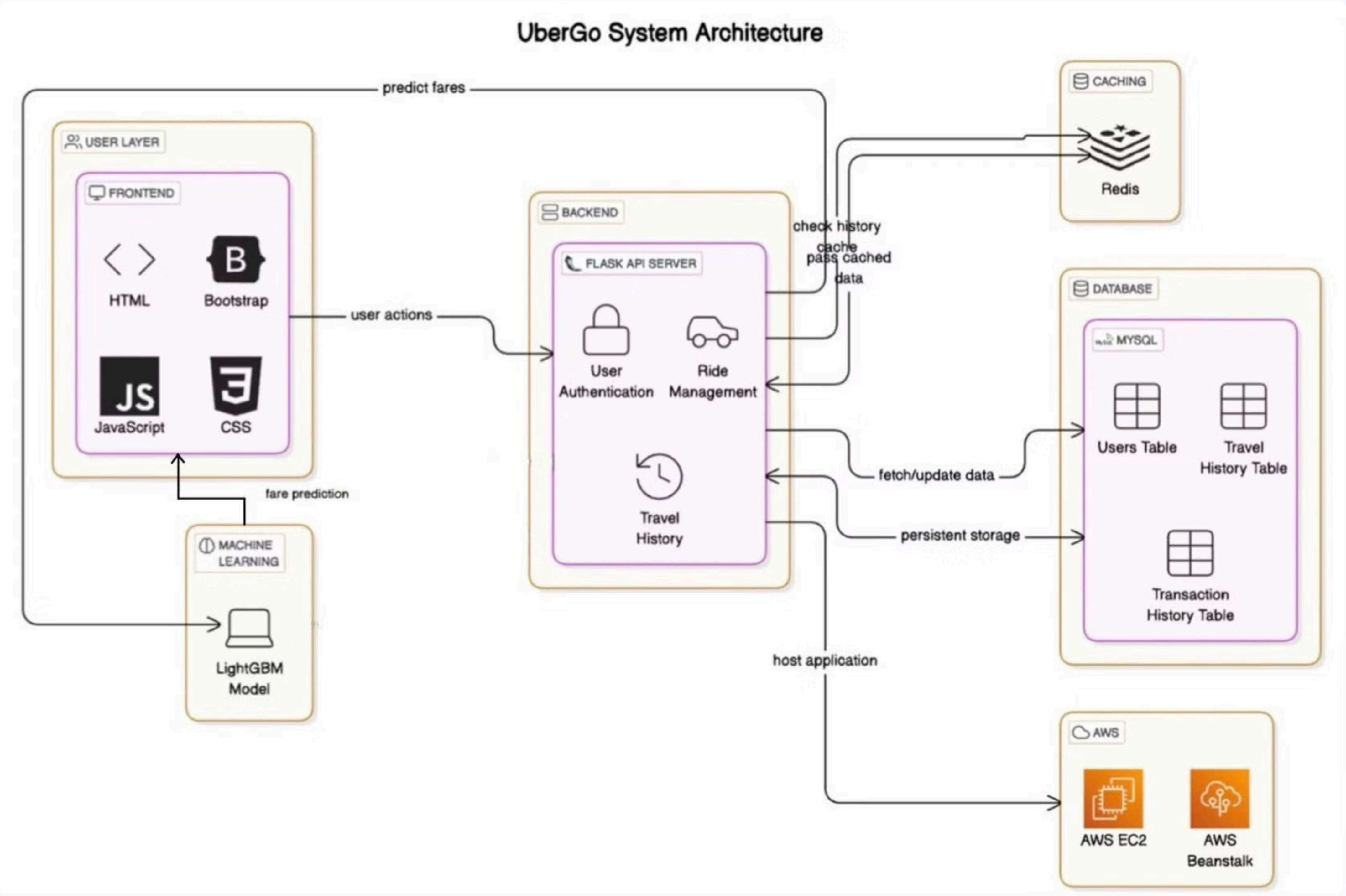
Hamsalakshmi Ramachandran	017423666
Kush Bindal	017441359
Soumya Challuru Sreenivas	017518618
Sugandha Chauhan	017506190

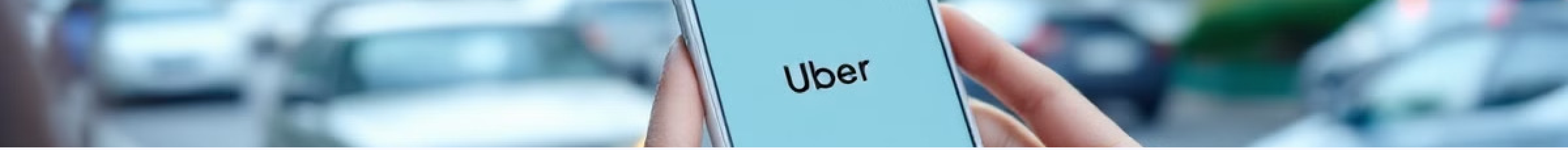
1. Database Schema



dbdiagram.io

2. System Architecture Design Diagram





3. Dynamic Pricing Algorithm

User Input: pickup/dropoff coordinates, datetime, passenger_count, is_event

Calculate Distance using Haversine Formula

Extract Temporal Features: hour, day_of_week, month

Predict Base Fare using LightGBM Regressor Model

Calculate Demand-Supply Multiplier using trip counts

Check for Event/Holiday and Apply Event Multiplier

Compute Adjusted Fare: adjusted_fare = base_fare * demand_multiplier * event_multiplier

Model Comparison

- We have utilized the LightGBM Regressor, a highly efficient and scalable machine learning model, to train our dataset for dynamic price prediction.

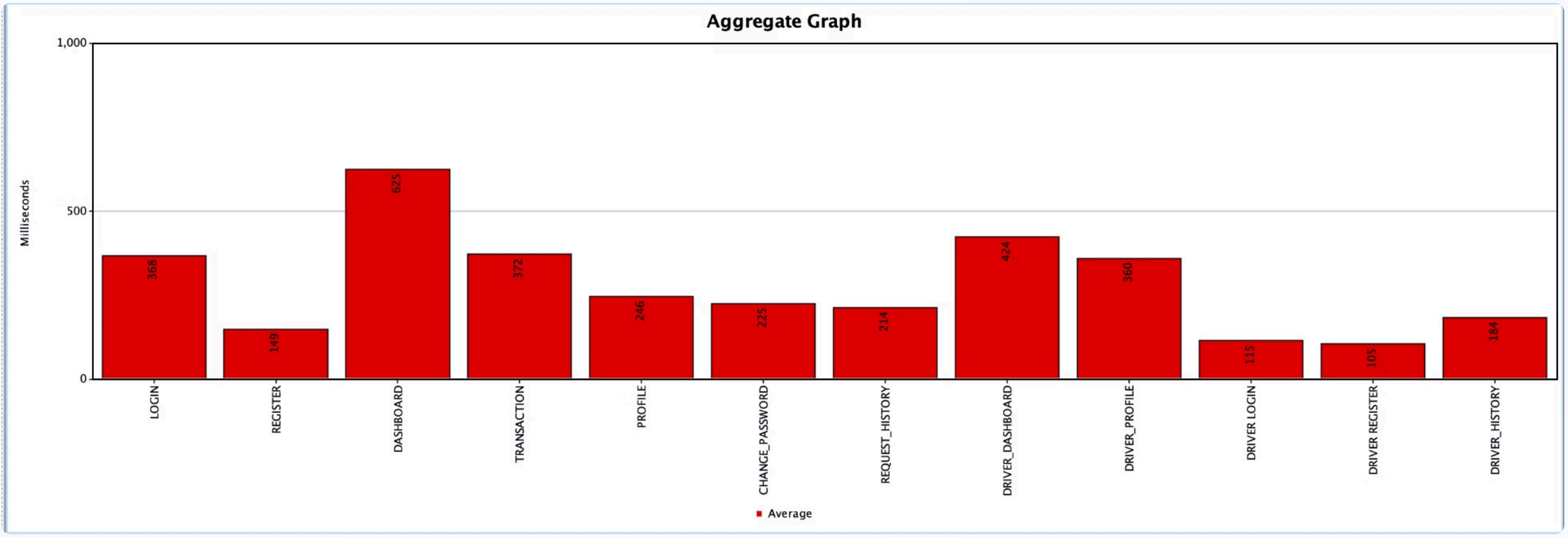
- LightGBM Regressor is a machine learning model from the LightGBM framework that is specifically designed to solve regression tasks, where the goal is to predict continuous numerical values (e.g., house prices, ride fares). It is based on the gradient boosting framework and builds an ensemble of decision trees to make accurate predictions

- This model is specifically designed to handle complex, nonlinear relationships between features such as distance, time, and demand patterns, ensuring accurate fare predictions while maintaining fast training and prediction speeds. Its ability to process large datasets and automatically handle missing data makes it an ideal choice for real-time pricing scenarios in ride-service applications.

Feature	LightGBM	XGBoost	Random Forest	Linear Regression
Type of Model	Gradient Boosting (Leaf-wise Trees)	Gradient Boosting (Level-wise Trees)	Bagging (Random Trees)	Linear (Equation-Based)
Complexity Handling	Excellent for Nonlinear Relationships	Excellent for Nonlinear Relationships	Moderate for Nonlinear Relationships	Poor for Nonlinear Relationships
Speed (Training)	**Fastest** (Histogram-Based)	Fast (but slower than LightGBM)	Moderate	Fast
Speed (Prediction)	**Fastest**	Fast	Moderate	Fast
Scalability	Excellent (Handles Big Data)	Excellent (Handles Big Data)	Moderate	Poor (Not Suitable for Big Data)
Feature Engineering Required	Minimal	Minimal	Minimal	Requires Manual Scaling & Transformation
Interpretability	Moderate (Feature Importance Available)	Moderate (Feature Importance Available)	Moderate	Excellent (Easy to Explain Coefficients)
Handling Missing Data	Yes (Automatically Handled)	Yes (Automatically Handled)	No (Requires Preprocessing)	No (Requires Preprocessing)
Overfitting Handling	Excellent (With Regularization)	Excellent (With Regularization)	Moderate (May Overfit with Deep Trees)	Poor
Performance on Nonlinear Data	Excellent	Excellent	Good	Poor
Data Volume Support	High	High	Moderate	Low

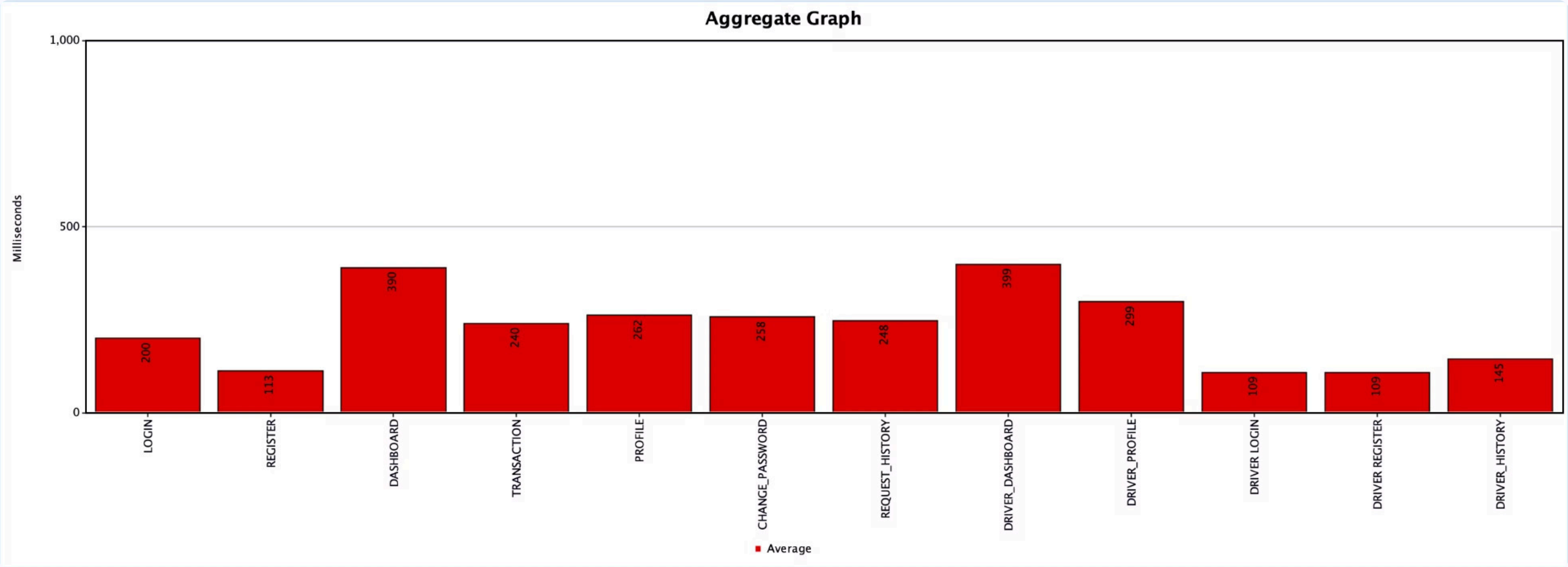
4. Average response Time Graphs : Apache JMeter

Base + ML



Base + Redis Cache+ ML

After implementing caching on Login, Travel History following are the results:



Endpoints	Base + ML	Base + Redis Cache + ML	% Improvement
User Login	~348ms	~208ms	~40% faster
Driver Login	~113ms	~100ms	~11.5% faster
User History	~274ms	~234ms	~14.6% faster
Driver History	~184ms	~145ms	~21% faster

Questions ?

Thank you