# Matching Users and Drivers using a Graph based Backend

**Arima Vu Ram Thayumanavar**
**Soumya Mohanty**

**Goals**:

- The primary aim of this project was to build an application that can be used to match users and drivers in a cab-sharing/car-pooling setting. And to support this application using a Graph based database.
- In order to accomplish this we had to build API functions which would perform all the CRUD operations on the database making the application database independent. Another requirement was to generate and load relevant data into the Graph database, so we could test the API and the application.
- The core of the application is the module which matches a user to a driver. We have implemented this using certain heuristics and also by using a Machine Learning model.
- We have used data stored in the graph as features and labels to train the model. And once trained set up functions a pipeline for it to get test data from the graph and give out the predictions.
- We have built a command line UI to Register new users, enable Login and a few other features. Along with a small demo to test the ride sharing application.

**Motivation**:

Learning about and implementing a Graph backend is what motivated us the most. This application turned out to be an interesting and appropriate application for using Graph databases. The use of Graph database made building some complex features very easy. We could easily pull the kind of data we needed from the database and not worry about efficiently joining the tables.

On top of that a graph database enables us to make dense and connected graphs which represent people interacting in real time. This kind of data can easily be harvested to use as training features of a ML model.

Integrating the model into the project was another motivating factor. Using different kinds of data we can build recommendation systems which will efficiently match users to drivers so the ride can be optimized to the users liking and convenience.

**Significance**:

A framework like this can be used for achieving various kinds of targets in a ride sharing setting. By using different ML models and fetching data needed to train them, we can have various kinds of functionality.

Our application implements one of such functionalities which is providing better matching of users and drivers. By better matching we mean, improving user experience by filtering the rides around them by suggesting the users rides similar to those they have given high ratings to before.

Illustrating that the use of Graph database not only makes development easier, it also reduces lookup time as compared to Relational Databases (because they need to join tables in order to extract the data we need)

**Model**:

We treat the problem of predicting which ride would the user like the most as a regression problem. The model predicts a rating as its output. Based on past data and other factors what rating would the user give this driver.

The model takes in averager rating of the driver, time for pickup, possible reroute distance in case the ride needs to pick-up more people, how old is the car and a history score as input and predicts the rating or a score that the user would give the driver.

The history score is computed based on the past relationship between the driver and the user. If the user and driver have never met then it is zero. Else the history score is calculated based on what rating the user had given the driver and a few other factors.

After training the model we implemented Backward elimination to eliminate features that had low impact on the results and kept features that had higher impact on the result. We estimated this by using the P values and the adjusted R squared value. By doing this we came to the conclusion that average driver rating and pickup time were the most optimal features that influenced the model predictions the most.

This project also illustrates the methodologies and best practices used to make use of graph database to power a machine learning model in real time.

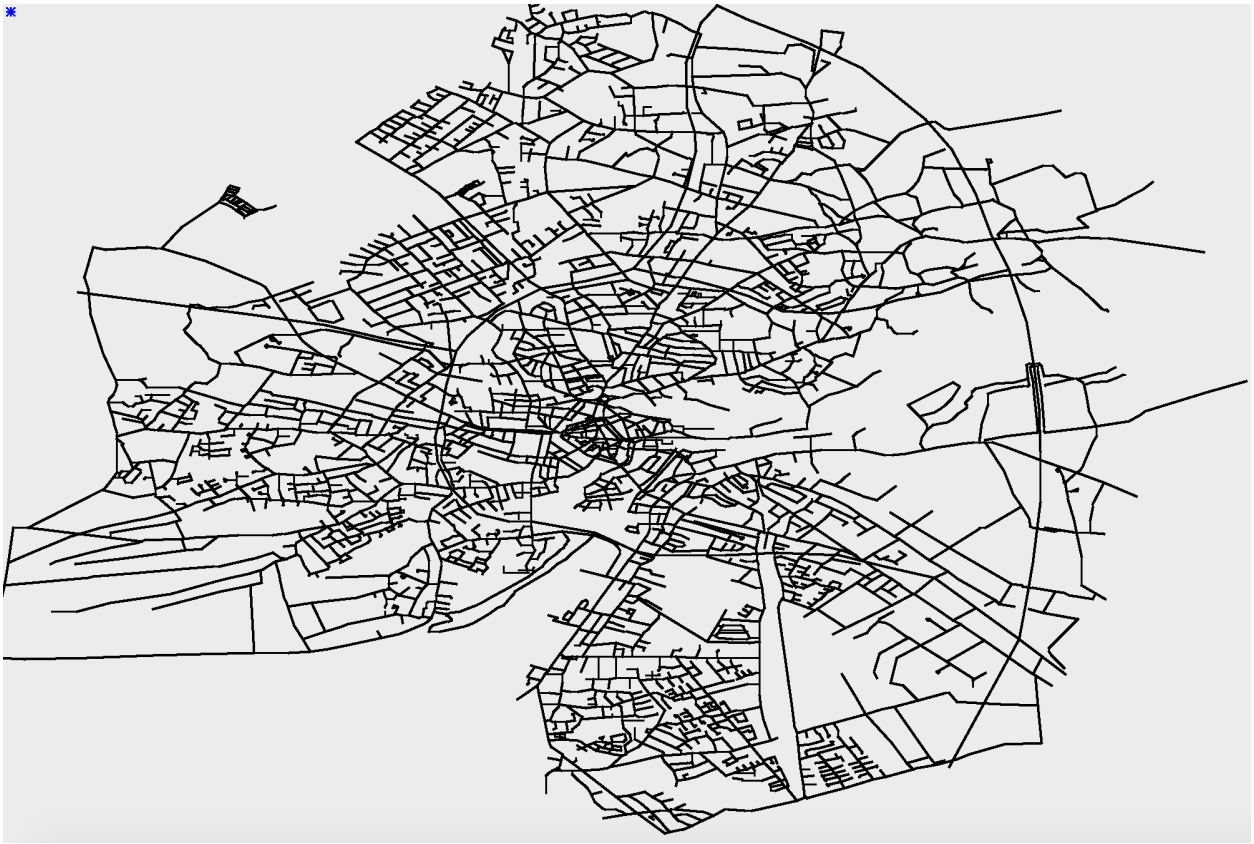
**Self-Assessment**:

We set out with the goals described above and have successfully implemented everything. Some improvements could be made and a few more features using the same framework can be added in the future. As of now we have done the following:

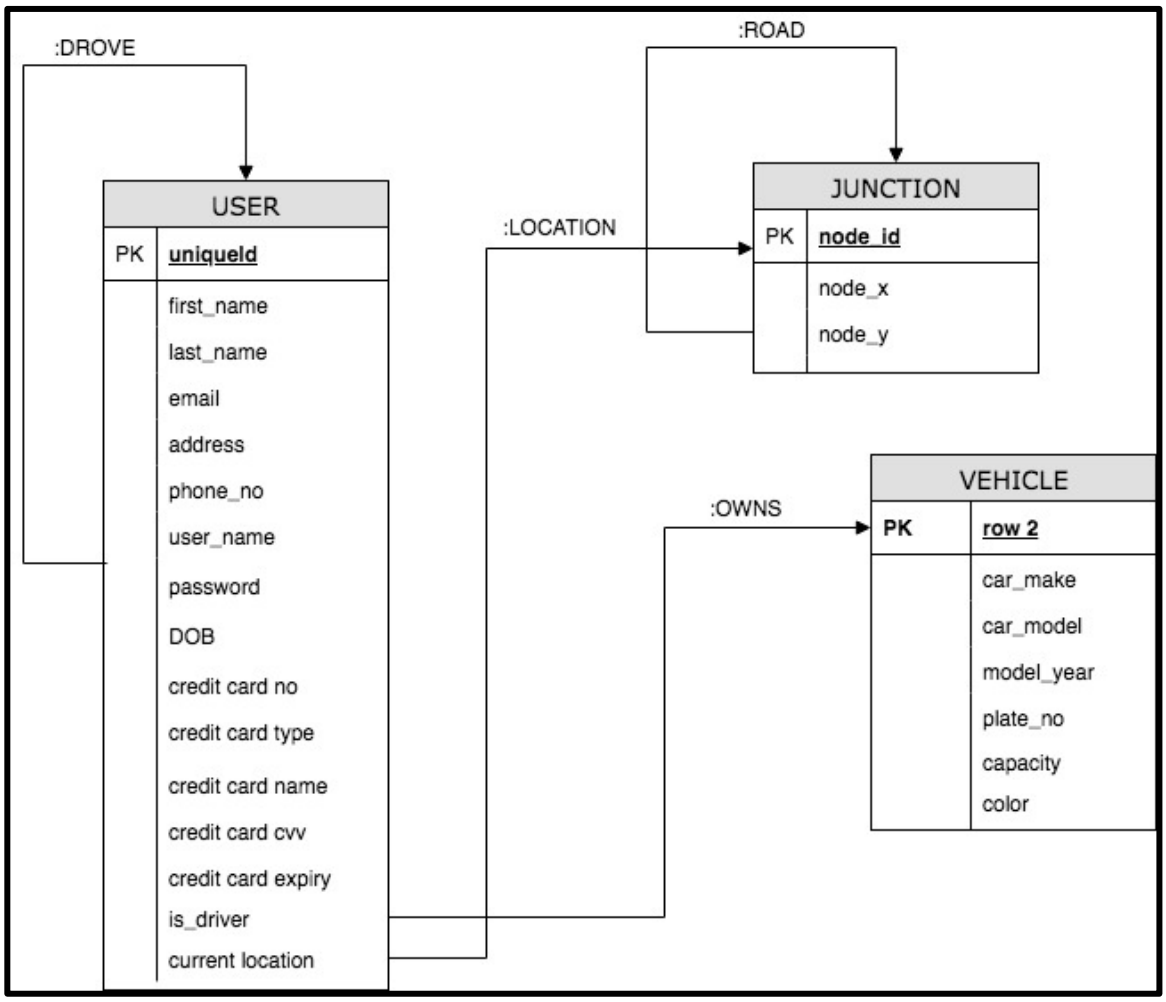
1. Learn and Implement a Graph backend
2. Make API to implement various CRUD operations
3. Build application to match users and drivers
4. Add a Machine Learning model to the matching application
5. Train the model to make predictions in real time
6. Create the pipeline for taking input feeding it to the trained model and producing the output.

**Data Acquisition Methodology**:

For our project, we didn't have a proper data source as its more user data, hence we ended up fabricating relevant user data. For the location map, we used the map of Oldenberg and each junction of road as location points. The map when plotted on mapplotlib looks as shown below,



**ER diagram:**




In a graph database there are different types of nodes and relationships between nodes that can be defined, instead of table and foreign key that links tables in a relational database management system. As shown above, for our use case we have designed common user datatype to represent both the user using the application and drivers picking up users. Hence, we define relationship between different users that saves the user transaction history and ratings. A detail list of features tracked by the relationship is shown below.

**:DROVE**

Is the edge that connects two USER type nodes, and records the driving history of the user with that particular driver. Holds the following information,

pickup_location
pickup_time
drop_location

drop_time
rating
ride_fare

**:ROAD**

Is the edge that connects two JUNCTION type nodes, and records the length of the road between the two junctions. Holds the following information,

len

**:LOCATION**

Is a dynamic edge that holds the current location of the user. Currently in our design implementation it is considered as an edge connecting the JUNCTION node, representing the users location, which however keeps updating as and when the location of the driver moves. However, we are looking at the trade off having it as a graph database RELATIONSHIP vs having it just as an information field of a node that keeps dynamically updating. Holds the following information,

node_id

**:OWNS**

This edge links the users in the USER table who represent the drivers, to the corresponding the car they own that's been pre-approved and stored in the database. Having such a model allows flexibility for different registered users to use only specifically owned cars and also perform reverse searches based on the user's preference for type of car (Pooling vs normal vs luxurious cars). Does not hold additional information apart from the link.

**DEMO**:

We have a client.py and driver.py that replicates a user and driver front-end behavior.

So let's say a client requests a ride and we have the corresponding driver around and we will show the UI of the corresponding users in the following demo screenshot:

Rider User UI

Driver User UI


Rider User UI

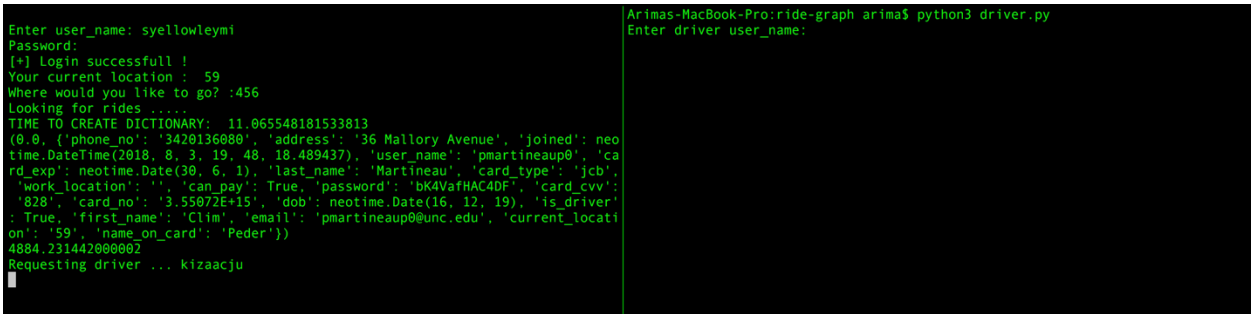
Driver User UI


Rider User UI

Driver User UI


Rider User UI

Driver User UI



Rider user UI:

```
Enter user_name: syellowleymi
Password:
[+] Login successfull !
Your current location :  59
Where would you like to go? :456
Looking for rides .....
TIME TO CREATE DICTIONARY:  11.065548181533813
(0.0, {'phone_no': '3420136080', 'address': '36 Mallory Avenue', 'joined': neo
time.DateTime(2018, 8, 3, 19, 48, 18.489437), 'user_name': 'pmartineaup0', 'ca
rd_exp': neotime.Date(30, 6, 1), 'last_name': 'Martineau', 'card_type': 'jcb',
 'work_location': '', 'can_pay': True, 'password': 'bK4VafHAC4DF', 'card_cvv':
 '828', 'card_no': '3.55072E+15', 'dob': neotime.Date(16, 12, 19), 'is_driver'
: True, 'first_name': 'Clim', 'email': 'pmartineaup0@unc.edu', 'current_locati
on': '59', 'name_on_card': 'Peder'})
4884.231442000002
Requesting driver ... kizaacju
```

Driver user UI:

```
Arimas-MacBook-Pro:ride-graph arima$ python3 driver.py
Enter driver user_name: kizaacju
Password:
[+] Login successfull !
Your current location :  5982
User 1 located at pickup location : 59
Enter "Y" to accept this ride ? Y
Routing to pickup the user  syellowleymi
```

Rider User UI

```
Reached checkpoint 5979
Approaching point 99 ....
Approaching point 99 ....
Approaching point 99 ....
Approaching point 99 ....
Approaching point 99 ....
Reached checkpoint 99
Approaching point 91 ....
Approaching point 91 ....
Approaching point 91 ....
Approaching point 91 ....
Approaching point 91 ....
Reached checkpoint 91
Approaching point 85 ....
Approaching point 85 ....
Approaching point 85 ....
Approaching point 85 ....
Approaching point 85 ....
Reached checkpoint 85
Approaching point 77 ....
Approaching point 77 ....
Approaching point 77 ....
Approaching point 77 ....
Approaching point 77 ....
Reached checkpoint 77
Approaching point 67 ....
Approaching point 67 ....
Approaching point 67 ....
Approaching point 67 ....
Approaching point 67 ....
Reached checkpoint 67
Approaching point 64 ....
Approaching point 64 ....
Approaching point 64 ....
Approaching point 64 ....
Approaching point 64 ....
Reached checkpoint 64
Approaching point 61 ....
Approaching point 61 ....
Approaching point 61 ....
Approaching point 61 ....
Approaching point 61 ....
Reached checkpoint 61
Approaching point 59 ....
Approaching point 59 ....
Approaching point 59 ....
```

Driver User UI

```
Approaching point 5982 ....
Approaching point 5981 ....
Approaching point 5981 ....
Approaching point 5981 ....
Approaching point 5981 ....
Approaching point 5979 ....
Approaching point 5979 ....
Approaching point 5979 ....
Approaching point 5979 ....
Approaching point 5979 ....
Approaching point 99 ....
Approaching point 99 ....
Approaching point 99 ....
Approaching point 99 ....
Approaching point 99 ....
Approaching point 91 ....
Approaching point 91 ....
Approaching point 91 ....
Approaching point 91 ....
Approaching point 91 ....
Approaching point 85 ....
Approaching point 85 ....
Approaching point 85 ....
Approaching point 85 ....
Approaching point 85 ....
Approaching point 77 ....
Approaching point 77 ....
Approaching point 77 ....
Approaching point 77 ....
Approaching point 77 ....
Approaching point 67 ....
Approaching point 67 ....
Approaching point 67 ....
Approaching point 67 ....
Approaching point 67 ....
Approaching point 64 ....
Approaching point 64 ....
Approaching point 64 ....
Approaching point 64 ....
Approaching point 64 ....
Approaching point 61 ....
Approaching point 61 ....
Approaching point 61 ....
Approaching point 61 ....
Approaching point 61 ....
Approaching point 59 ....
```

Rider User UI

```
Approaching point 2569 ....
Approaching point 2569 ....
Approaching point 2569 ....
Approaching point 2569 ....
Reached point 2569
Approaching point 2579 ....
Approaching point 2579 ....
Approaching point 2579 ....
Approaching point 2579 ....
Approaching point 2579 ....
Reached point 2579
Approaching point 2580 ....
Approaching point 2580 ....
Reached point 2580
Approaching point 2586 ....
Approaching point 2586 ....
Approaching point 2586 ....
Approaching point 2586 ....
Approaching point 2586 ....
Reached point 2586
Approaching point 2589 ....
Approaching point 2589 ....
Approaching point 2589 ....
Approaching point 2589 ....
Approaching point 2589 ....
Reached point 2589
Approaching point 2602 ....
Approaching point 2602 ....
Approaching point 2602 ....
Approaching point 2602 ....
Approaching point 2602 ....
Reached point 2602
Approaching point 2609 ....
Approaching point 2609 ....
Approaching point 2609 ....
Approaching point 2609 ....
Approaching point 2609 ....
Reached point 2609
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 2613 ....
Reached point 2613
Approaching point 5457 ....
```

Driver User UI

```
Approaching point 2557 ....
Approaching point 2557 ....
Approaching point 2563 ....
Approaching point 2563 ....
Approaching point 2563 ....
Approaching point 2563 ....
Approaching point 2563 ....
Approaching point 2569 ....
Approaching point 2569 ....
Approaching point 2569 ....
Approaching point 2569 ....
Approaching point 2569 ....
Approaching point 2579 ....
Approaching point 2579 ....
Approaching point 2579 ....
Approaching point 2579 ....
Approaching point 2579 ....
Approaching point 2580 ....
Approaching point 2580 ....
Approaching point 2586 ....
Approaching point 2586 ....
Approaching point 2586 ....
Approaching point 2586 ....
Approaching point 2586 ....
Approaching point 2589 ....
Approaching point 2589 ....
Approaching point 2589 ....
Approaching point 2589 ....
Approaching point 2589 ....
Approaching point 2602 ....
Approaching point 2602 ....
Approaching point 2602 ....
Approaching point 2602 ....
Approaching point 2602 ....
Approaching point 2609 ....
Approaching point 2609 ....
Approaching point 2609 ....
Approaching point 2609 ....
Approaching point 2609 ....
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 2613 ....
Approaching point 5457 ....
```

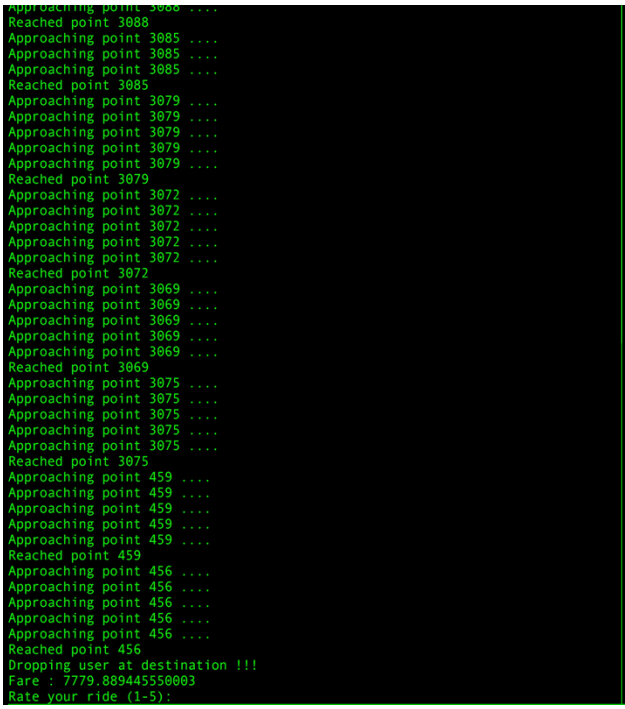
Rider User UI

```
Approaching point 1621 ....
Reached point 1621
Approaching point 1626 ....
Approaching point 1626 ....
Approaching point 1626 ....
Approaching point 1626 ....
Reached point 1626
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1635 ....
Reached point 1635
Approaching point 1661 ....
Approaching point 1661 ....
Approaching point 1661 ....
Approaching point 1661 ....
Approaching point 1661 ....
Reached point 1661
Approaching point 1672 ....
Approaching point 1672 ....
Approaching point 1672 ....
Approaching point 1672 ....
Reached point 1672
Approaching point 1651 ....
Approaching point 1651 ....
Approaching point 1651 ....
```

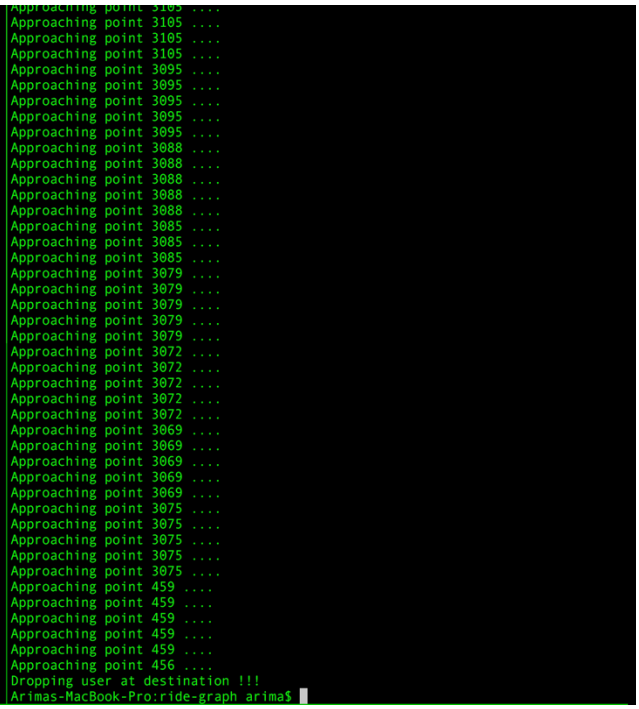
Driver User UI

```
Approaching point 1630 ....
Approaching point 1621 ....
Approaching point 1621 ....
Approaching point 1621 ....
Approaching point 1621 ....
Approaching point 1621 ....
Approaching point 1626 ....
Approaching point 1626 ....
Approaching point 1626 ....
Approaching point 1626 ....
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1635 ....
Approaching point 1661 ....
Approaching point 1661 ....
Approaching point 1661 ....
Approaching point 1661 ....
Approaching point 1661 ....
Approaching point 1672 ....
Approaching point 1672 ....
Approaching point 1672 ....
Approaching point 1672 ....
Approaching point 1672 ....
Approaching point 1651 ....
Approaching point 1651 ....
Approaching point 1651 ....
```

Rider User UI




Driver User UI



Rider User UI: