RideShare

Optimizing cab-services through cab-sharing

A Project Report Submitted in
Partial Fulfillment of Requirements for the Degree of

Bachelor of Technology

by

Praveen Kumar (2019CSB1108)
Manan Singhal (2019CSB1099)

Department of Computer Science & Engineering

Indian Institute of Technology Ropar

Rupnagar 140001, India

May 2023

Abstract

The transportation industry has been slowly influenced by ride-sharing services because they provide cost-effective transportation and are more environmentally friendly. Ride-sharing allows passengers with overlapping routes to travel together and reduces the fare for each passenger.

Both authors agree that ride-sharing is important because they faced issues while they were doing their internships at Gurugram and Bangalore. Many employees used to travel individually to their respective offices but they had overlapping routes. They could have shared the rides.

The objective of this project is to develop a cab-sharing platform that will help passengers traveling with overlapping routes to share rides. The GPS system will track the rides, routes will be updated optimally, and pricing will be done accordingly. There are two interfaces, the user interface, and the driver interface. Features for both interfaces can be found in Chapter 3.

This project can provide cost-effective rides and reduce traffic congestion, and air pollution. Also, it will reduce fuel consumption, increase driver earnings and encourage more customers to register for the service and increase average vehicle occupancy, etc. We believe that the project can be a novel and useful addition to current ride-booking apps.

Acknowledgements

We would like to thank all those who have contributed to this project.

First of all, we would like to thank our supervisor Dr. Apurva Mudgal who has been always providing his guidance since the start of this project.

We would also like to thank the evaluation panel for providing us with feedback about our project.

We would also like to express our gratitude to our families and friends who has been a constant support through this project.

Honor Code

We certify that we have properly cited any material taken from other sources and have obtained permission for any copyrighted material included in this report. We take full responsibility for any code submitted as part of this project and the contents of this report.

Praveen Kumar (2019CSB1108)

Manan Singhal (2019CSB1099)

Certificate

It is certified that the B. Tech. project “RideShare” has been done by Praveen Kumar(2019CSB1108), Manan Singhal (2019CSB1099) under my supervision. This report has been submitted towards partial fulfillment of B. Tech. project requirements.

Dr. Apurva Mudgal
Project Supervisor
Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar-140001

Contents

List of Figures

Chapter 1

Introduction

1.1 Single-sharing vs Ride-sharing cab services

This project is going to discuss the optimization of cab services. Let's discuss currently available cab services like OLA, Uber, etc. These third-party services allow passengers to book a ride through an app and connect them with a nearby driver who provides them with requested transportation at an agreed fare charge (shown in the app at the time of booking). This is called a single-sharing cab service. In these single sharing, it is possible that more than one passenger takes a cab. However, these passengers must have planned the journey from the same origin to the same destination together.

We consider the following scenarios. Suppose two passengers with two independent travel routes want to use the cab service. If the two travel routes have significant overlap, they can be provided with a single cab. The cab may first pick up Passenger 1, then Passenger 2, and then drop them off in order of their destinations. This is called Ride-sharing. Single-sharing service would have assigned individual cabs to both passengers. Fare charges will be reduced per head. Let's look at further how the project may be successful in basic details.

1.2 Advantages of Ride-sharing

1.2.1 Passenger fare reduction

The most important benefit of cab sharing is that it helps an individual to save money by sharing the ride fare with some other person. Instead of paying all of the fare an individual can divide the cost with the person(s) sharing the ride. Further, the fare calculation can be done properly on the basis of how much time a single person was there in the cab, and how much time there were two people in the cab. Suppose at a time there are n (n>1) persons in the cab, let r be the cost for traveling 1 unit distance and they travel x unit distance before the number of passengers changed in the cab, then $1.4 \times (\frac{x \times r}{n})$ cost would be added to each passenger fare. For n=1, fare cost will be added directly as $x \times r$. Significant fare reductions can be seen as per the formula.

1.2.2 Driver fare increment

Apart from the fare reduction for the passengers, cab sharing will also help drivers to get more fares for the same distance. The factor of 1.4 in the formula of fare addition is for the benefit of the driver. Let's take a very simple example. Suppose two passengers have significant overlapping paths and they shared the ride. Let r be the fare cost per unit distance, x be the distance when only Passenger 1 was in the cab, y be the distance when only Passenger 2 was in the cab, and z be the distance when they both were in the cab. The driver will get $(x \times r) + (y \times r)$ + $2 \times (1.4 \times (\frac{z}{2}) \times r)$. Hence, for the same distance, the driver got an extra fare of $.8 \times (\frac{z}{2}) \times r$.

1.2.3 Environmental Friendly

Cab sharing will reduce the number of cabs on the roads, causing less traffic congestion and less air and noise pollution. There would be less fuel consumption. This would also lead to higher average vehicle occupancy.

1.2.4 Increased social interaction

Sharing a cab helps people to interact with each other who have not yet met, allowing making new social connections. It would help to build a strong social network.

1.2.5 Organization benefits

Using cab-sharing, an organization can serve more customers with the same number of available cabs. Also, this would increase their earnings and strengthen their customer base.

1.3 Disadvantages of Ride-Sharing

1.3.1 Increased travel time

In the cases of ride-sharing, it might be possible that the ideal time of the journey increases for a passenger. It might be possible that the driver had to take a diversion to pick up another passenger which will definitely delay the ideal journey time. Hence, the passenger provides a tolerance value which signifies the maximum delay that he/she can afford.

Chapter 2

Previous Work

2.1 Mumbai Navigator

In the previous semester we studied a paper about Mumbai-Navigator. Below are some points and there descriptions about the Mumbai Navigator

2.1.1 About Mumbai Navigator

It was used in Mumbai to plan optimal travel using trains and transport buses. A user needs to enter the origin point and destination, then it generates a plan which consists of several routes showing where to take and switch the train/buses. The claim is that it shows the plan which takes the minimum total travel time along with that it shows all the possible routes to reach the destination. Also, it shows the expected waiting time, arrival time, and travel time for each bus/train. Also, the plans are flexible according to the arrival of different buses. The timing for trains is deterministic while for buses it is not. So it requires some precomputed data like expected arrival and travel time for buses from one stop to another. The travel time is usually predictable like it takes 2.5 hours to travel from X to Y whereas the waiting time at point X is not predictable from the data but with the statistics, we can find the expected value for the same.

2.1.2 Model of Mumbai Navigator

The model is in the form of a tree. In the tree each node u has a label as S(u) which indicates the stop at u. And an edge from u to v i.e (u,v) has a label B(u,v) which is a bus route that includes the stops u and v in that order. u and v need not be consecutive stops here. The root of the tree would be the origin and the destination will be the leaf nodes.

Algorithm: At each node, u do the following: Let v_1,v_2,....v_k be the children of the u. B(u,v_1), B(u,v_2).....B(u,v_k) are the buses where B(u, v_i) indicates the bus route from u to v_i. Now take the bus which arrives first. Now suppose we reach v_i. Now repeat the algorithm at vi. Overall we start from the root and go to a leaf node and at every stop, we take the bus that arrives first.

2.1.3 Optimal Plan

Let the height be the maximum number of buses used in a journey. Let T_b(s,h) be the expected total time taken in a bus b which starts from stop s and reaches its destination using h stops intermediate. The algorithm uses the plan for h-1 height to generate the plan for h height. So T_b(s,h) would be either **a)** same as optimal height h-1 **b)** the user gets down at some s_i and then follow the optimal path of h-1 height from there. R_b(s, s_i) is the time taken from s to s_i using the bus. T(s,h) be the expected time to complete the journey starting from stop s using the h height plan.
T_b(s,h) = min(T_b(s,h-1), R_b(s, s_i) + T(s_i, h-1))

Let B_s denote the buses which pass through the stop s and it is found that the optimal plan will only contain these buses. So we find the optimal plan with height h starting from stop s as below:

T(s,h)= min($\frac{1}{\sum_{b\in B} F(b)}$ +$\sum_{b\in B}$ $\frac{F(b)}{\sum_{b\in B} F(b)}$ $T_b(s,h)$)
$whereB$ $\subseteq$B$_s$

Now the need to find the subset of B_s i.e B which has total possible combinations are $2^{|Bs|}$. That makes it very time-consuming. But we can ignore some buses in order to save time. The ignoring criteria for the buses are explained below through lemmas.

Lemma 1: Let B_s= b_1, b_2, ..., b_3 denote the set of buses passing through stop s. Let T_{bi} = T_{bi} (s, h). Assume that T_{b1}, T_{b2} . . . . T_{bm} (if not, sort B_s and renumber). Then the optimal plan from s must involve waiting for exactly the buses b_1, b_2,...., b_k for some k and ignoring the other buses even if they arrive first.

Lemma 2: Suppose B_s is as above. In other words, T_i^* is the expected time

$$T_i^* = \frac{1}{\sum_{j=1}^{i} F(b_j)} + \sum_{j=1}^{i} \frac{F(b_j)}{\sum_{j=1}^{i} F(b_j)} T_{b_j}$$

if we wait for buses b_1, b_2. ...,b_i at stop s and thereafter follow the respective optimal restricted height h plans. Let l be smallest such that T_l^* $\leq$T$_{b_{l+1}}$. Then it suffices to wait for buses B = b_1, b_2...,b_l, and ignore the other buses even if they arrive first. In other words: T (s,h) = T_l^*

2.1.4 Preprocessing

Mumbai Navigator gets the data from Bombay Electric Supply and Transport (BEST). They provide a list of stops available on the routes along with this they provide the frequency of each bus route. They also provide the estimated time taken by the bus between consecutive bus stops. As discussed above, we need to get the values of every pair of consecutive bus stops on the given route. As the data provided by the BEST is inadequate for the travel time for the consecutive bus stops they used a two-phase approach. In the first phase, we estimate the distance between consecutive bus stops. In the first phase, we estimate the time travel between consecutive bus stops.

2.1.5 Express Buses

Till now, they did not think about express buses. Express buses are the ones that stop at a limited number of stops. Let's consider 3 stops A, B, and C on any routes. Now, Bus Z will directly move from stop A to stop C, i.e. the time taken to reach stop A to C will be less than the time taken by some other bus from A to B and then B to C. Their above methods did not take care of the following case. They violate this case and show that ordinary buses are better than this approach. So, they add these linear inequalities into the program wherever the routes of express buses overlap.

2.1.6 Local Trains

They integrated trains within their program by assuming the same as buses. To connect both the train and bus systems they add a walking connection into it. A walking connection is the time required to travel from the bus station to the railway station or vice-versa.

2.2 Cab Services and their features

We have also studied some of the currently available services and list out their advantages and disadvantages.

2.2.1 BlaBlaCar

BlaBlaCar is a carpooling service that connects drivers and passengers heading in the same direction. Let's look at how it works:
There are two user profiles, driver, and passenger. Both first need to sign up and provide proper ID. After the verification, whenever a driver is going to some location, he needs to provide the route, journey time, and the empty number of seats. A passenger who wants to travel can search for matching rides, then the app will show the matching rides with the driver's details and price per seat. If the passenger finds the ride suitable he can request to join, and the request will be sent to the driver for acceptance or rejection. If the ride is accepted,

a notification will be sent to the passenger. Now the driver would pick up the passenger from the agreed pick-up location. Fare would be calculated according to the distance traveled by the passenger. In this way, more passengers can join as per the available seats. Hence blablacar provides cost-effective rides for passengers by sharing the cost with the passengers who are heading in the same direction. This service is a little bit similar to our project as our project also has the idea of cab sharing which leads to fare cost reduction. There are differences also. 'blablacar' is designed for long-distance journeys, generally connecting two cities, while our project aims for short distances within a city. 'blablacar' drivers are private car owners who are already making a journey while our project is going to be connected with the full-time professional drivers. 'blablacar' have the fixed price system and passenger have to pay according to the BlaBlaCar's payment system while in our project the fare is going to be dynamic. 'blablacar' service is not available 24/7 as this is not operated by the full-time drivers while in our project passengers would be able to book cab anytime.

Chapter 3

Project Features and Implementation

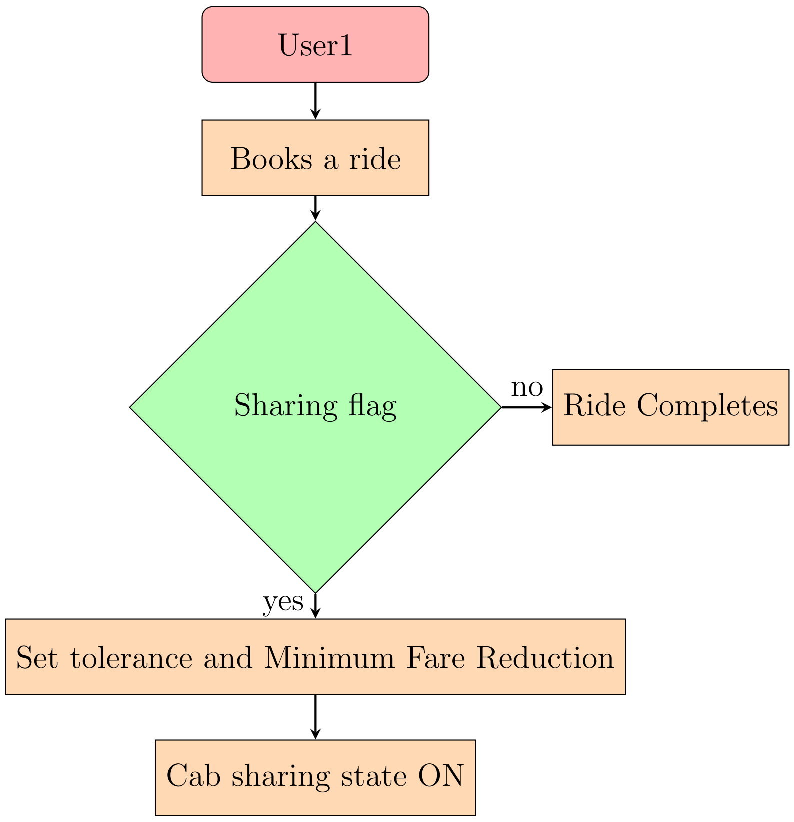

Figure 3.1: Flowchart of booking a fresh ride

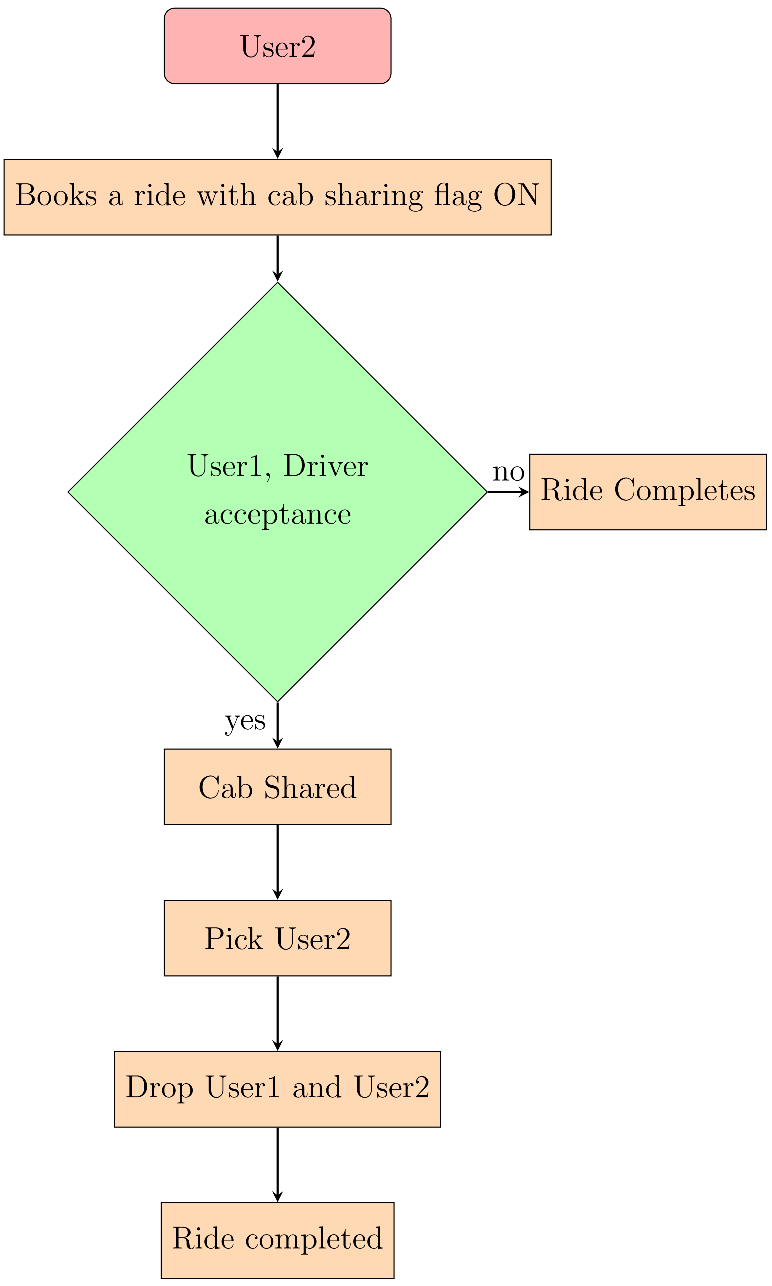

Figure 3.2: Flowchart of a sharing ride request

3.1 User Interface

The passenger gets a search bar in which the user can select the pick-up location and destination location. After setting these locations, the user can choose a car type, then the user has the option to enable cab sharing. If the user enables cab sharing user is asked to enter a tolerance value and minimum fare reduction. The tolerance value is the maximum delay user can afford in the estimated journey time .

There are two cases.

1) Passenger gets a new ride. If a user hadn't enabled cab sharing then the ride would be completed in a classical way. If a user had enabled cab sharing and another user requests a ride with cab sharing and have significant path overlapping, the request would be shown to the driver and user1. If both accept the request, then the driver would pick the user2. And then drop user1 and user2 in order of the distance.

2) Passenger gets an ongoing ride. When a user requests a ride with a cab sharing flag on and it matches with an ongoing ride. Driver picks the user and drops users in order of distance.

The request contains the updated fare cost for both users.

Please refer to page 9 and page 10 for flowcharts.

3.2 Driver Interface

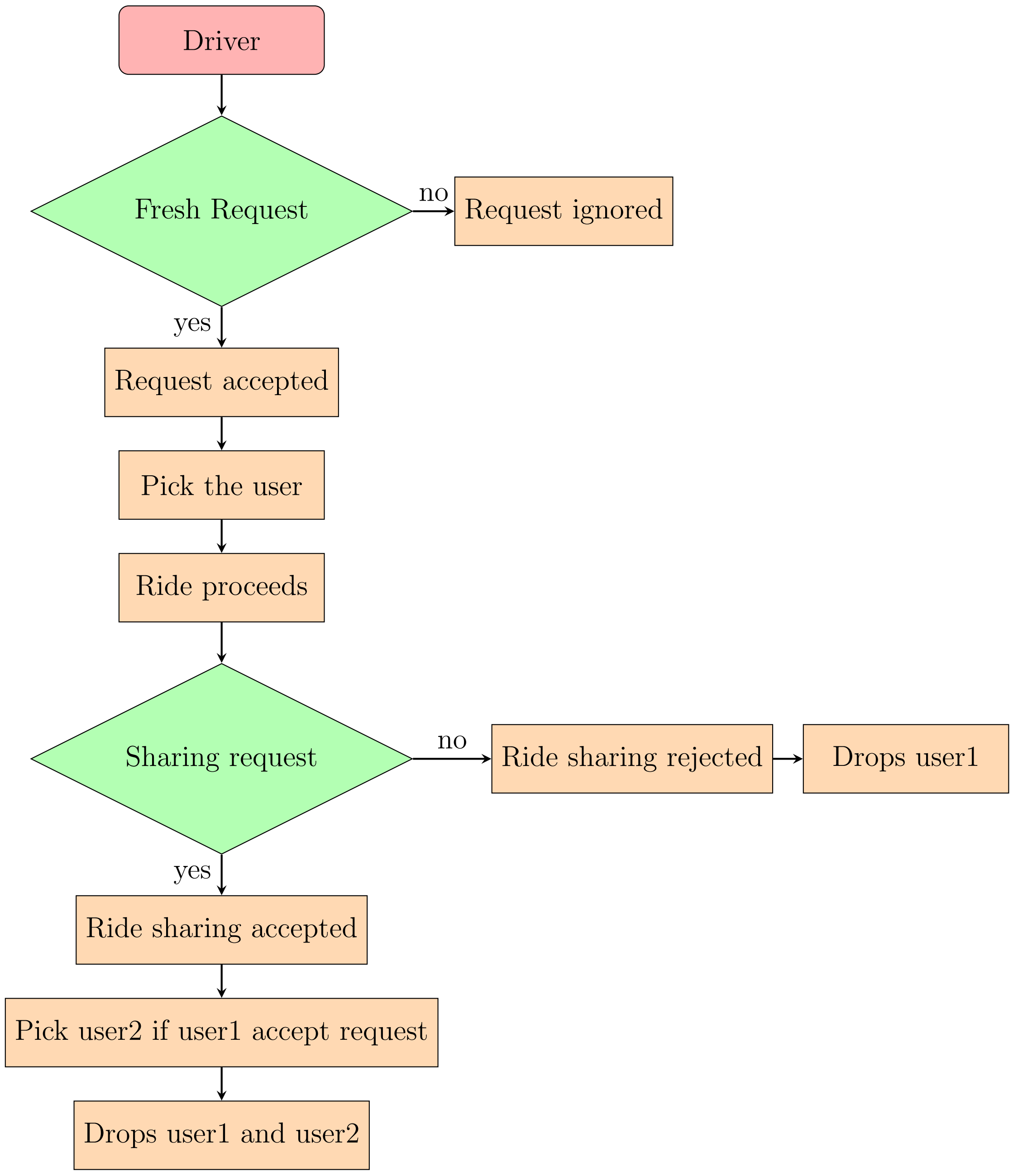

Figure 3.3: Flowchart of driver interface

There are two cases for the driver:
1) One is that he is sitting idle and gets a request with passenger details and fare. Now if the driver accepts the request he would immediately move to pick up the passenger.
2) Now if another passenger wants to join the ride then the request would be shown to the driver with the updated route and updated fare. Now it is up to the driver whether he wants to accept the request or not. The current passenger also needs to accept the request of the new passenger.
Please refer to page 12 for the flowchart.

3.3 Algorithm for ride allotment

The algorithm for single-sharing is explained below:
This is the usual algorithm for OLA, Uber, etc. Whenever some person requests a ride, a notification is shown to the nearby drivers. Interested drivers can accept the request and arrive to pick up the passenger and then the ride is started after entering the PIN and then tracked. Passengers have no freedom to choose the driver. Fare payment can be done during the journey or at the end of the journey.

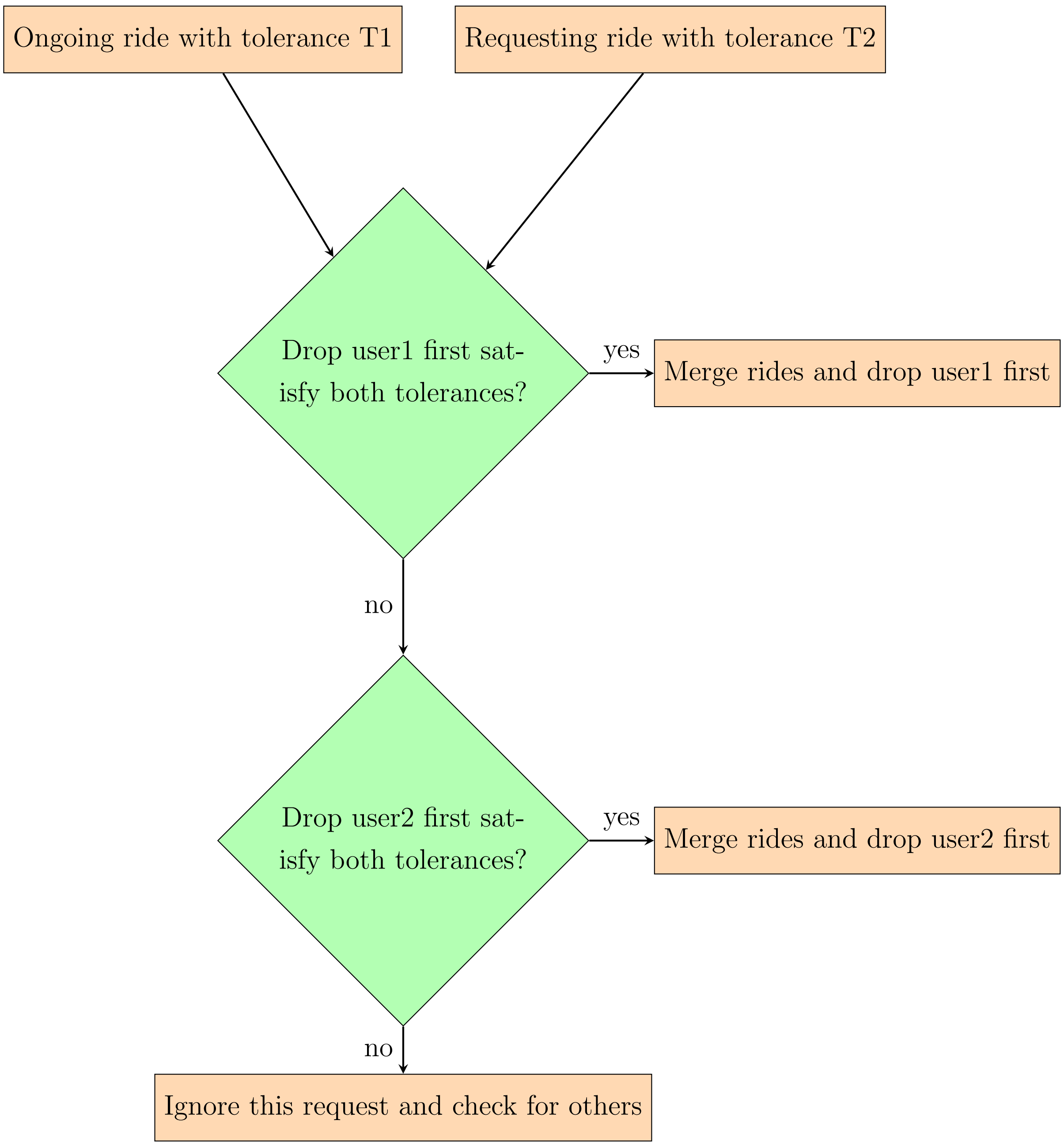
Ongoing ride with tolerance T1
Requesting ride with tolerance T2
Drop user1 first satisfy both tolerances?
yes
Merge rides and drop user1 first
no
Drop user2 first satisfy both tolerances?
yes
Merge rides and drop user2 first
no
Ignore this request and check for others

Figure 3.4: Flowchart for ride allotment

The ride-sharing algorithm can be as follows:
Suppose a passenger is traveling with a car-sharing flag enabled and some person in his vicinity request for a ride with a cab-sharing flag enabled. Both passengers have their own respective tolerances. The algorithm first would check if the new passenger is added to the ride, and would it satisfy both tolerances if the first passenger is dropped first to its destination. If yes then the algorithm will send a request to the driver and the passenger. If they accept the request then the new passenger can join the ride. Now take the second case when one or more tolerances are not satisfied when the first passenger is dropped first. Now the algorithm will check if it is possible to satisfy both tolerances if the second passenger is dropped first. If yes then a request would be shown to the driver and the passenger. If they accept the request then the new passenger can join the ride. And the fare charges would be updated for both passengers. In the third case when both tolerances cannot be satisfied, in that cases this request will be discarded by the algorithm and the algorithm would check for new requests. Please refer to the flowchart on page 14.

3.4 Algorithm for pricing

The algorithm for the pricing is quite simple. For single sharing, the price is already mentioned while booking the ride on the basis of the distance. For cab sharing, the different kinds of distances are calculated like when the first passenger is only in the cab when both passengers are in the cab, and when the second passenger is the only passenger in the cab. Then the fare is divided according to the proper proportion of the distance. Let us take an example, suppose there are two passengers who share a cab, x is the distance for which the first passenger was the only passenger in the cab, y is the distance when the second passenger was the only passenger in the cab, and z is the distance when both passengers were in the cab. Now suppose the fare is P per unit distance, then the first passenger would be charged $(P \times x) + (1.4 \times P \times \frac{z}{2})$. The second passenger would be charged $(P \times y) + (1.4 \times P \times \frac{z}{2})$. Fare saving for both passengers is $0.6 \times P \times \frac{z}{2}$

3.5 Useful Google Maps APIs

Below are some useful Google-Maps-Api Maps APIs with their sample input and output

3.5.1 Default API for getting directions

1. URL

2. **Input:** The origin, destination, and key are the input parameters we need to enter. origin and destination can be provided in longitude and latitude values.

3. **Function call format:**
   ```
   @GET('/directions/json')
   Future<DirectionsResponse> getDirectionData(
   @Query('origin') String origin,
   @Query('destination') String destination,
   @Query('key') String googleMapsApiKey);
   ```

4. **Output:** This API output gives us various details about the journey like total distance, and total time and also provides the steps for the whole journey in the form of a JSON message.

5. **Use in our platform:** Used this API when booking a ride while sending ride requests to drivers at most 3 km apart. We are updating the dropoff time in the rider app using the estimated time returned by this API.

3.5.2 API with Departure Time

1. URL

2. **Input:** The origin,destination,key, alternatives, and departure_time are the input parameters we need to enter. departure_time can be provided in the form of seconds or as 'now'.alternatives is a bool value, if set true then it may provide multiple paths for a route.

3. **Function call format:**
   @GET('/directions/json')
   Future<DirectionsResponse> getDirectionData(
   @Query('origin') String origin,
   @Query('destination') String destination,
   @Query('key') String googleMapsApiKey,
   @Query('departure_time') int departureTimeInSeconds);

4. **Output:** This API output gives us various expected details about the journey like total distance, and total time and also provides the steps for the whole journey in the form of a JSON message. Departure time is the extra output entry for this API.

5. **Use in our platform:** Used this API when a driver accepts the ride, showing the user the estimated time to reach the destination after 't' time from now, where, t denotes the time taken by the driver to reach the pickup location.

3.6 Design challenges faced

There were some issues that came up while deciding on the algorithm. When two passengers are in the cab, whom to drop first to his destination? When users have enabled ride-sharing we need to decide to which ongoing driver we need to send requests and make a maximum profit for the user. Similarly, in the driver app, we can't show all ride-sharing requests to the driver, we need to sort it in such a way that it will be beneficial for the driver and the user with up to 3 ride-sharing requests atmost at a time to the driver. We faced issues while designing the database schema for the ride-sharing rides generated by our application.

3.7 Project Link

You can find the application link here.
Click me

Chapter 4

Future Innovations

4.1 Extended version of this project

We now list some features that can be added to a later version of our application.

4.1.1 Filters for the user

Some useful filters that will enhance the user experience are as follows.

1. Finding a driver on the basis of a rating filter. This filter will be helpful in finding a high-rated driver. With a high-rated driver, a person can feel safer and more comfortable rather than with a low-rated driver. This will also act as an incentive for drivers to serve customers in the best possible way.

2. Future availability of cab in future time filter. This filter can be helpful to book a ride in advance (several hours before the departure time). This filter will reduce the last-time rush. Some reservations and security deposits can be charged to the person booking the ride. Security charges can be refunded after successful journey completion. It might be possible that a passenger books a ride and cancels it at the last time causing a loss to the driver, hence the security deposit would be charged.

3. Fare Payment Method filter. This filter can be useful when a user only has limited types of payment methods. It might be possible that a person only

has online payment methods. So he can use this filter to avoid the drivers who only take some other payment methods like cash only.

4. Drivers language filter. This filter can be used by a person who has been shifted to some new location for some work and only know a few languages and doesn't know the local language. In this case, he can apply a filter with language.

5. Availability of music system in the cab filter. This filter can be used to book cabs which can provide cabs which can provide music or radio as some users prefer to listen to music while traveling.

4.1.2 Filters for the driver

Some useful filters that can enhance the driver experience are as follows.

1. Finding a passenger on the basis of a rating filter. After every ride, a driver can give a rating to the passenger on the basis of the ride experience.

2. Receive requests within a particular distance filter. This will help the driver they will only travel to pick up the passengers who are within a fixed radius. For example, a driver may only want to pick up passengers who are within a 2 KM radius.

3. Receive requests within a particular time span filter. This will help the driver to set a specified time range in which he wants to pick up the passengers for the ride. For example, a driver may choose the time 7 AM to 2 PM, due to some work after 2 PM or whatever so reason.

4. Payment accept method filter.This filter will help to choose the payment methods that they want to accept. For example, some drivers want to take only cash.

4.2 Bidding feature for drivers

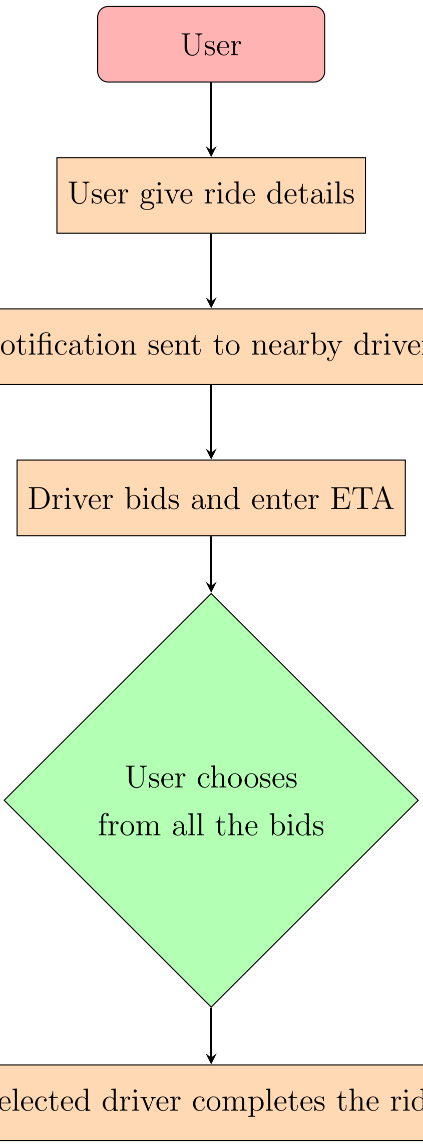
User
User give ride details
Notification sent to nearby drivers
Driver bids and enter ETA
User chooses from all the bids
Selected driver completes the ride

Figure 4.1: Flowchart for bidding feature in future innovations

A live bidding feature will help the driver to bid for a ride requested by a passenger. When a passenger requests a ride with a proper pick-up location and destination location, the app will show him an estimated fare. Then a notification will be sent to nearby drivers with the pick-up location and destination location. The drivers who have an interest in that ride can bid and enter the fare and ETA. Then the passenger can accept the ride on the basis of the fare, ETA, driver rating, etc. Then the respective driver will receive a notification to confirm the ride. The driver can pick up the passenger and drops him at his destination and the fare is charged as the bid amount. This feature will help the passengers to choose their ride on the basis of the factors mentioned above. Please refer to the flowchart on page 20.

4.3 Create some users and show the fare savings

After the successful implementation, we are going to create some real-time users and they will be using this app. We will record the price savings for an individual and that would be analyzed later on.

Chapter 5

Conclusions

This project aims to optimize cab services through a cab-sharing model. In the journey of this project, we studied Mumbai Navigator. Also, we studied a special kind of cab service BlaBlaCar, and find out the similarities and differences with our project. Also, we studied the Google Maps APIs and used some APIs in the implementation of this project. We developed a model for cab-sharing and implemented it in this project. Also, we have discussed the future innovations that the project can have. Currently, a base version of this project is being done and further updates are being in progress. We gained a lot of technical and analytical skills through this project.

References

BlaBlaCar. URL https://www.blablacar.com/. 7

Google-Maps-Api. URL https://developers.google.com/maps/documentation. 16

influenced. URL https://www.statista.com/statistics/1155981/ride-sharing-market-size-worldwide/. i

Mumbai-Navigator. URL https://www.cse.iitb.ac.in/~ranade/ijtmnavigator.ps. 4

OLA. URL https://www.olacabs.com/. 1, 13

Uber. URL https://en.wikipedia.org/wiki/Uber. 1, 13