Implementation of Real-Time Rickshaw Hiring Application

by

Md. Monzurul Haque
17101285
Ahmed Saquib
17101308
Monika Baishnab
17301007
Md Navid Hossain
17101511
Nafis Bin Reza
17101411

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

Md. Monzurul Haque
17101285

[illegible]

Ahmed Saquib
17101308

Navid

Md. Navid Hossain
17101511

[illegible]

Monika Baishnab
17301007

Nafis

Nafis Bin Reza
17101411

Approval

The thesis/project titled "Implementation of Real-Time Rickshaw Hiring" submitted by

1. Md Monzurul Haque (17101285)
2. Ahmed Saquib (17101308)
3. Monika Baishnab (17301007)
4. Md Navid Hossain (17101511)
5. Nafis Bin Reza (17101411)

Of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 02, 2021.

Examining Committee:

Supervisor:
(Member)

[illegible]

Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md. Golam Rabiul Alami, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Mahbubul Alam Majumdar, PhD
Professor and Dean, School of Data and Sciences
Department of Computer Science and Engineering
Brac University

Ethics Statement

Hereby, We consciously assure that for the thesis paper ”Implementation of Real-Time Rickshaw Hiring Application” the following is fulfilled:

1) This material is original work, which has not been previously published elsewhere.

2) The manuscript is not being considered for publication anywhere at this time.

3) The writers' research and analysis are reflected in the publication wholly and truthfully.

4) The paper appropriately acknowledges the efforts of co-authors and co-researchers.

5) The findings are discussed in the context of previous and ongoing research.

6) All sources used are correctly disclosed (correct citation). Text that has been copied must be marked as such with quote marks and a suitable reference.

7) All of the authors were directly and actively involved in the extensive effort that led to the implementation of the paper, and they will be held accountable for its content.

The norms of the Ethical Statement can have serious implications if they are broken.

We agree to the aforementioned declarations and certify that this submission adheres to Solid State Ionics' rules as described in the Authors' Guide and the Ethical Statement.

Abstract

Steps to eliminate the issue of rickshaw hiring are now a severe need. Furthermore, the best solutions to the problem are developing an algorithm and implementing a rickshaw app. Besides that, many people believe that educating rickshaw drivers on using a smartphone app to get customers is unrealistic. So, we need to develop a simple program for rickshaw drivers and let them reach more customers. Even so, the algorithm must be user-friendly and beneficial to both rickshaw drivers and users. Therefore, an algorithm for rickshaws has been implemented in this article and applications for both riders and users, which solve our everyday rickshaw hiring issue.

Keywords: Rickshaw; Hiring; Digital; Application; Algorithm

Dedication

Dedicated to all the rickshaw pullers who struggle hard for their livelihoods day and night and the passengers who use rickshaw as their transport.

Also dedicated to our parents, and honorable teachers.

Acknowledgement

We would like to start by thanking Allah for His blessing on us, without which nothing would be possible. Then we would express our gratitude to our respected supervisor, Amitabha Chakrabarty sir, for his invaluable assistance during the research. Finally, we want to emphasize how much guidance, support, and expertise we received from all of the faculties, as well as our parents, aided us in a variety of ways.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Rickshaws are one of the most important mediums of transport in Bangladesh. There are nearly more than 750,000 rickshaws, and they employ over one million people. In the past, however, rickshaws have been almost overlooked by strategy producers and researchers. For example, in the past few years, ride-sharing applications took care of cars and bikes. Uber, Pathao, Shohoz, etc., are some examples of successful ride-sharing applications in Bangladesh. However, no one talked about rickshaws. However, rickshaws are used by almost all people of all classes in our country. Whenever we go for a short tour, we use rickshaws rather than cars or bikes. Around 400,000 carts run every day as one of the fundamental vehicle modes for the occupants of Dhaka only. Though rickshaws are available everywhere in Bangladesh, it is challenging to find rickshaws in some residential areas such as Baridhara, Basundhara, Gulshan, Dhanmondi, and DOHS. Again, when we are visiting a new place, we may have little idea about our destination, which can cause a higher chance for rickshaw pullers to charge excessive amounts from people. Sometimes we can see that rickshaw pullers demand extra fare during extremely high temperatures or heavy rain, but people do not want to pay an extra fare for the same distance. These are some problems with using rickshaws for daily rides. Our objective is to develop an algorithm and an application based on rickshaws. It will help us to avoid some unpleasant situations such as heavy rain. During heavy rain, it is difficult to find a rickshaw. Even if we find some, they may charge an extra fare for a ride. We are planning to solve the problem with extra fare with our algorithm.

1.2 Motivation

Since the launch of Uber in 2009, it has changed the transportation industry. This ride-sharing concept was very new to the people. At first, users could hire only cars. This is confined to cars, and we can hide motorbikes, CNG(Auto Rickshaw), etc. Nevertheless, there is no service found for hiring a rickshaw. More reasons:

- Sometimes rickshaws cannot be found in specific locations, especially in residential areas like Gulshan, Baridhara, Banani, Niketan, etc. So they introduced a new rickshaw system, which decreases the number of rickshaws from 10,000 to a mere 1,230. These rickshaws are known as "Community rickshaws," and they are registered and licensed. Rickshaw pullers pull these rickshaws wearing orange uniforms and ID cards with their photo on them [3]. Because of the scarce availability of rickshaws in those areas, people have difficulty finding the rickshaws. In 2019, to alleviate traffic congestion, Dhaka's two city corporations voted to restrict rickshaws on three important highways. Progoti Sarani, which runs from Kuril to Sayedabad via Rampura and Khilgaon, Mirpur road, which runs from Gabtoli to Azimpur via Asad Gate, and Elephant road, which runs from Science Laboratory intersection to Shahbagh, all have substantial rickshaw traffic. This application aims to explore this issue and solve the problem by finding rickshaws quickly by using our app.

- The economic situation of rickshaw pullers is deficient. Many people join the labor market every year, and one of them is rickshaw because it does not need any kind of deposit, investment, lobbying, or educational requirements, or expertise [7]. They make a modest amount of money and have to feed their families every day with it. Sometimes they have to rent a garage, repair and buy some parts for their rickshaw, which is very costly. Even the majority of rickshaw drivers in Dhaka do not own a rickshaw. They need to hire a rickshaw on rent every day [7]. Sometimes it is hard for them to collect the rent they have to pay. They often do not find any passengers for not being in the correct place at the correct time. Our app will help the rickshaw pullers get more passengers. Rickshaw pullers can earn more money, and they can find the passenger easily from a reasonable distance by using our app.

- Going to different locations varies with different fares. In our daily life, we still debate with rickshaw-pullers about the fares. Sometimes rickshaw pullers are given less money. Sometimes they charge insane fares, defying regular fares as there is no fixed rate, so both parties propose different fares, which sometimes turn into violence. Our app will solve this issue by calculating the fare, which should be acceptable to both parties. Also, our app will consider climate change while calculating fares.

- The safety of life is more important than anything. We live in a society consisting of all kinds of people. We are not unfamiliar with crimes that are conducted by rickshaw pullers. Crimes like harassment, kidnapping, etc., occur frequently. On the other hand, the news of rickshaw pullers getting abused and beaten up by passengers is also there. Using our app, we can bring safety and justice to both sides. Through our app, we can track down the information we need for identification. This will keep away crimes as the criminal will not be able to escape. When using our app, passengers will see the details of the rickshaw puller and if they have got any negative reviews.

1.3 Research Objectives

Our thesis aims to make rickshaw hiring easier, and we will try to do that by digitizing our rickshaw service. To do so, we are focusing on creating a ride-sharing app to make this possible. A good number of tested and optimized algorithms will be implemented in our mobile application. We know we do not have any meter to measure our fare for rickshaw service right now, for which sometimes the rickshaw pullers do not get their expected fare or try to demand some extra from the customers. One of our main priority of us will be to solve this problem by implementing some algorithms to calculate the fare based on the environment, time, traffic jams, road conditions, etc. Again, rickshaw rents for a day are not the same for all the places; fares will vary from place to place based on that also. So that the rickshaw puller and the passenger do not need to think about fare counting or anything. On the other hand, a rickshaw puller cannot pull a rickshaw all over the day and all over the city because of physical ability and root restrictions. So, our target will be to solve this also and locate the permitted roots for the vehicles. Our other target is to make it easy to find the rickshaw for a place so that if someone wants to hire a rickshaw, she/he does not face difficulties. Top of that rating system will help the customer, and rickshaw puller knows about the person they will ride with. We are also focusing on personal safety. Individual safety is paramount. Rickshaw pullers and passengers will both be able to get each other's information if any crime happens. In short, we are focusing on doing:

- Our primary focus is to create, optimize, and implement a modified, appropriate, and relevant algorithm for this whole rickshaw ride-sharing app.
- Optimized and rickshaw-oriented fare management and calculation procedures will be applied to benefit both rickshaw pullers and passengers. We will do it by collecting some data from a random field-level survey from various areas in Dhaka and try to find, optimize, and make a pattern of accurate fare calculation with the help of the machine learning algorithms of Python.
- Real-time data generated from Google Map API and Weather forecast will be implemented in this project to identify the best possible route and get a reasonable fare for a particular journey. Furthermore, as we plan to implement this project with Flutter programming language, this language has automatic support for implementing this kind of 3rd-party service API installation process, which needs permissions from the user's mobile handset.
- We may need to modify the map, which is generated from Google API, to customize it according to the modified pathway of using narrow roads if needed because of the restrictions of rickshaws running on the main roads of Dhaka city.
- Creating an easy-to-use and user-friendly UI design for our app based on UI/UX design basics will be primarily implemented via Flutter's default UI-making libraries. The skeleton will be drawn with the help of Adobe XD or Figma.
- We will use NoSQL as a query language to implement our desired database. The database will be produced with Apache Web Server via XAMPP software to initially demonstrate the additional capability to shift to another more efficient DBMS.
- The testing phase will be done correctly by doing some Unit Testing with the default flutter testing mechanism. All the whole thesis and project procedures will be well documented, following the standard documentation standards.

- The testing phase will also include field testing. Field testing will include hiring a rickshaw and testing if the app is fully functioning as it is supposed to be. Also, verify that our algorithm is working as it is coded to be.

1.4 Challenges faced

- We have faced some challenges implementing algorithms. It is because the fare is one of the significant issues of argument between rickshaw pullers and passengers. In many cases, a standard argument turns into violence. So, we have to make sure that we can calculate the fare, which is fair for both sides.
- We have faced difficulties choosing a suitable API for our service. At first, we decided to work with Barikoi API. Barikoi is a location data provider in Bangladesh that provides reliable location data in the local context. However, we have faced some problems using their service. For example, when measuring duration from one location to another, they could not provide specific data for a specific vehicle. For example, if we needed duration between two places through rickshaw, Barikoi could not provide such data. Also, Barikoi service providers are new, so they have some bugs that create problems for us. Then we decided to use Google's API for our app. Google has enormous data to offer and has been giving service to many companies, which encouraged us to use their service for our app.
- Still, we have one problem that we are working on. In our country, all roads are not the same. As a rickshaw is not an automatic vehicle, a rickshaw puller must pull physically, so road conditions should also be considered. As in a good road, it is easy for a rickshaw puller to pull his rickshaw. However, nevertheless, on the wrong road, it is hard for him to pull. So the fare should differ in different road conditions. However, today, it is not possible to get that data of road condition, though we are considering the weather situation as we can get that data efficiently.

Chapter 2

Literature review and background

2.1 Concept of Real-time Rickshaw hearing application

Ridesharing is an improved service that employs safe and pleasant mobile technology in real-time ride pooling with two groups of individuals as the driver and passenger. [2].Real-time ride-matching services using "smartphones" and automatic ride-matching tools Improved easy carpooling methods that rely on "meeting points" are also being tested. In Canada and the United States, ridesharing accounts for roughly 8 to 11 percent of the total transportation modal share. In North America, there are about 638 ride-matching services [1]. In developing markets like Bangladesh, ride-sharing services have recently gained tremendous popularity. In the country, there are right-sharing applications for cars and bikes only. However, there is still no such application or any system for the country's most common vehicle, the rickshaw. People of the country face a lot of problems while hiring a rickshaw or paying the fare of the rickshaw puller. Sometimes, the rickshaw driver may charge the customer an additional fare, or the customer pays less than the regular fare. Customers and rickshaw drivers often get into fare disagreements, which is a very common phenomenon. Developing a well-structured system to solve this problem has become a demand of time. As a result, we have developed a real-time rickshaw hiring application and an algorithm to calculate the fare for a rickshaw ride.

2.2 Income from rickshaw pulling

Rickshaw pullers in Dhaka city earn between BDT364.8 and BDT695.8 per day on average. The Dhaka city rickshaw pullers' net average daily income is BDT371.7 with BDT100/- at least and BDT800/- at maximum. The owners of Rickshaw notified that the net income per rickshaw per day. After all, the related costs range from BDT 30 to 80, with an average of around BDT 55. Furthermore, Rickshaw pullers are frequently constrained. Almost half of them had been confined in Dhaka an average of 5.8 times. The main causes have been" signal infringement" and" driving on VIP roadways" Rickshaw drivers were held for an average of 51 minutes before being released with or without bribe fines. Through the LEAs, they also have tire punctures and occupy passenger seats. [6]. As a result, it is extremely difficult for rickshaw pullers to earn a living on a daily 5basis. Again, they are sometimes duped by passengers. Passengers' misbehavior was reported by all rickshaw pullers. These include physical assault, belittlement, scolding, unjust compensation, and a fare dispute. 63.7 percent of rickshaw drivers have been assaulted, it is clear. [6]. That is why it is essential to have a system in place that ensures they are paid properly after each ride. And our algorithm has been designed in such a way that the rickshaw puller is properly compensated, taking into consideration factors such as weather, traffic, availability, distance, and so on.

2.3 Developing an algorithm for traffic jam

Google Maps uses two types of data to generate its traffic views and faster-route suggestions are based on historical data on the average time it takes to traverse a particular portion of the road at given times on certain days and real-time data from sensors and cellphones that record how quickly cars are going right now. Google turned to crowd sourcing at the beginning of 2009 to increase the precision of its transport forecasts. If you activate the GPS position of your Android phone on your Google Maps App, your phone returns anonymous data to Google to tell them how quickly your automobiles move. Google Maps continually aggregates the data from all the road automobiles and transmits it back to traffic layers through the colorful lines [8]. However, Google's traffic prediction service is incompatible with rickshaws. Because these vehicles must go on very remote roads where Google's service cannot function correctly, we had to develop our own algorithm to handle the problem. Our system collects the rickshaw's location status at frequent intervals, and our algorithm compares the data to determine whether or not the rickshaw is stuck in traffic. And this procedure is repeated till the trip is completed. Unlike Google, our system is totally independent and can complete the process even if there is no other rickshaw using the application. This is how we have developed a system that is suitable for rickshaws.

2.4 Fare calculation based on different locations

A rickshaw is a low-cost three-wheel vehicle. It uses a battery-powered motor or a person's physical strength to carry other people and loads. However, it does not have a specific fare system [5]. Dhaka had 37 rickshaws in 1941 and 181 rickshaws in 1947. According to the 1951 census, Dhaka was a district town with 62,469 people before 1947. In 1998, however, the city's population grew by about 8 million people, 112,572 registered rickshaws. In that year, all other towns in Bangladesh had 274,265 rickshaws, and all villages had 91,040. In 2019, the city had roughly 280,000 rickshaws, which is more than double the number of rickshaws registered in 2000, it is a widespread assumption. Figures-based estimates [9]. In recent years, the number has grown. However, there is currently no procedure for calculating the fare for each ride. Our problem was to orient our algorithm for any area in the country as the fare may vary from place to place. We have taken a variable to accomplish that and modified the value of the variable based on different locations. And to determine the value of that variable, we examine current fares for that specific place, which assists our algorithm to generate an adequate price for that particular area which is in fact helpful for passengers and rickshaw pullers.

2.5 Security system of Ride-sharing application

One issue that rings a bell with the expansion of such services is the safety of passengers, particularly in countries with low legislative constraints. No severe steps could be taken because the driver, vehicle, and ride were not completely monitored by the operator. Fuel has also been poured into the flames through poor feedback. As long as they fill the gap of improved services worldwide, it is vital to take possible safety precautions in order to ensure passengers' and riders' safety till reaching the destination from the beginning of the journey [4]. In Rickshaw Mama, we have developed the application in such a way that also helps the passengers and the riders to stay safe. To do that Rickshaw Mama keeps track of every vehicle that is shown in range. Like, when a user requests a ride the system creates a log for that particular request and keeps track of every vehicle which is in the 200 or 500 meters range at that particular moment. So that, if there is any occurrence reported then the system can give information about the vehicles which were present there at that particular time.

Chapter 3

Methodology

3.1 Methodology

The purpose of implementing a rickshaw hiring application is to serve people to find a rickshaw more efficiently with standard pricing; to do that, we need to consider real-time data from GPS of a particular time. Firstly, we need to determine the distance between pickup and destination, multiplied by the mileage, to determine the initial fare. We also need to monitor the supply and demand of a particular time, same as the order time from GPS, to form a relationship between supply and demand. From the distance and mileage, we calculate distance fare (F_D). Then, we calculate the surge multiplier (M_S) from supply and demand, multiplied by F_d when demand exceeds supply.

Our application will also consider the weather to give some benefits to rickshaw pullers. We are checking if the weather is rainy or not. If it is raining, we believe a multiplying factor to the total fare, M_W=1.2. If it is not raining, then we are checking if the temperature is between 18 and 32 or not. If it is between 18 and 32, we multiply M_W=1 with our total fare, and if not, then we multiply M_W =1.10 with the total fare. Traffic jams are a common phenomenon in Bangladesh and many other countries. So we handled that factor also with a movement tracking ideology based on the GPS tracking system. To implement that, we are checking if the rickshaw passed 100 meters in one minute or not. If a rickshaw stays 100 meters within one minute, some extra fare will be added to the total fare. Figure 3.1 is showing the process in detail.

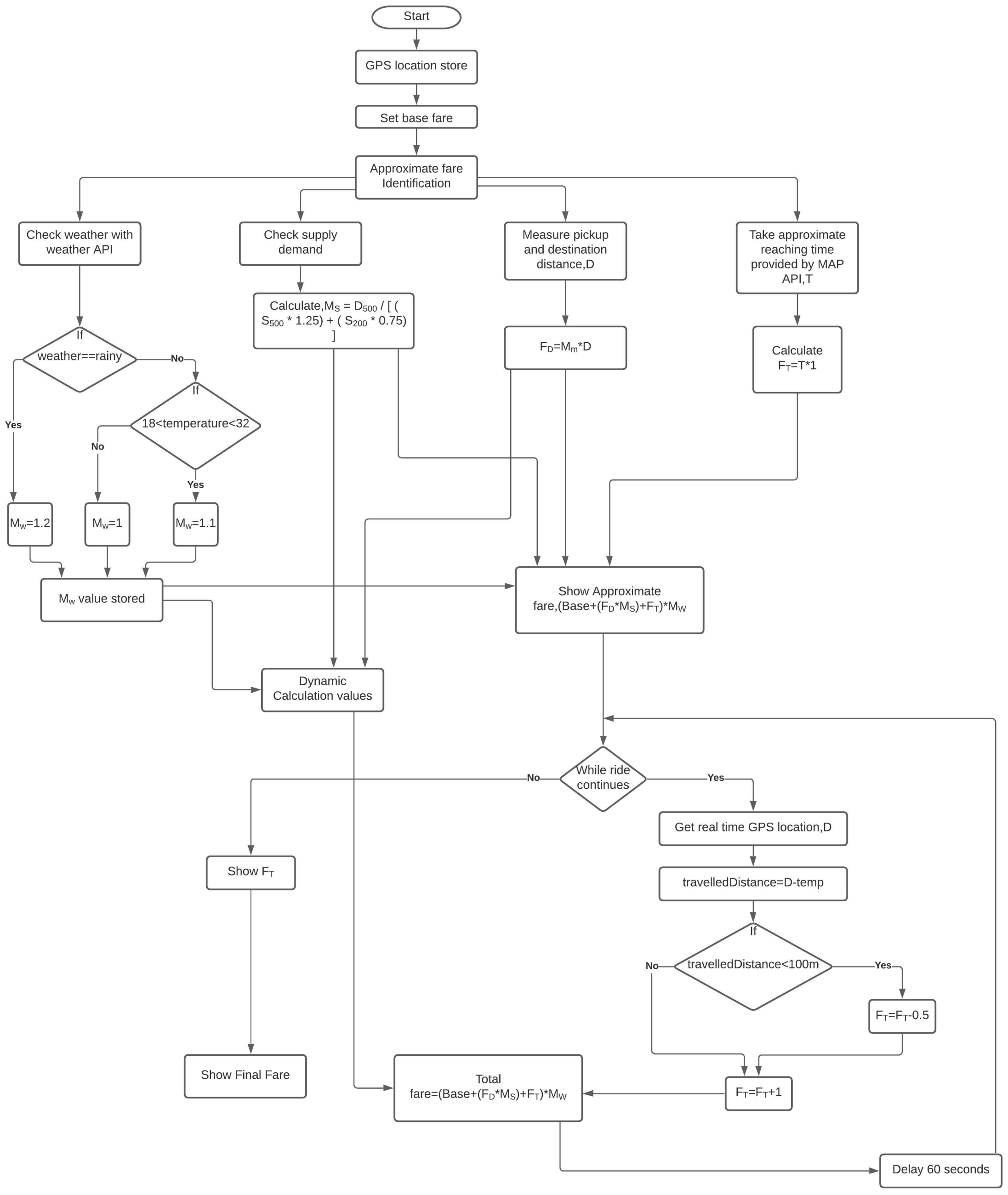

Start
GPS location store
Set base fare
Approximate fare Identification
Check weather with weather API
Check supply demand
Measure pickup and destination distance,D
Take approximate reaching time provided by MAP API,T
Calculate,$M_S = D_{500}$ / [(S_{500} * 1.25) + (S_{200} * 0.75)]
$F_D=M_m$*D
Calculate F_T=T*1
If weather==rainy
No
Yes
If 18<temperature<32
No
Yes
M_w=1.2
M_w=1
M_w=1.1
M_w value stored
Show Approximate fare,(Base+(F_D*M_S)+F_T)*M_W
Dynamic Calculation values
While ride continues
No
Yes
Get real time GPS location,D
travelledDistance=D-temp
If travelledDistance<100m
No
Yes
$F_T=F_T$-0.5
$F_T=F_T$+1
Show F_T
Show Final Fare
Total fare=(Base+(F_D*M_S)+F_T)*M_W
Delay 60 seconds

Figure 3.1: The flow chart of the Rickshaw Hiring Algorithm

3.2 Technologies Used

To make the concept accurate, we used some of the tools and technologies. Technology plays a vital role in implementation. There are various options to choose between, and each option has its advantages and disadvantages. So proper justification and relevance must be considered carefully to select the technologies while implementing. We will discuss here the technologies we used in "Rickshaw Mama Real-Time Rickshaw Hiring Service":

3.2.1 React Native

Facebook, Inc. developed React Native, an open-source mobile application framework. It allows developers to leverage React's framework alongside platform-specific capabilities to develop apps for Android, Android TV, iOS, macOS, tvOS, Web, Windows, and UWP. It is an exciting framework that allows web developers to use their existing JavaScript skills to create sophisticated mobile applications. In addition, it enables speedier mobile development and more efficient code sharing between iOS, Android, and the Web without compromising the end-user experience or application quality.

Why use React native?

React Native provides faster, more comprehensive, and efficient development, and it is easy to understand among those who have previous React JavaScript development experience. Being a cross-platform development environment, we can build apps for any device in the future, which ensures reliability. React Facebook backs up Native, so it is not going away too soon. There are massive projects, documentation, and references on react native as it has been a popular framework for quite some time, which helped us build this kind of app.

3.2.2 Amazon Web Services(AWS)

Amazon Web Services is an Amazon brand that operates metered pay-as-you-go cloud computing platforms and APIs to people, businesses, and governments. The Amazon Web Services portfolio includes over 100 services, including computation, databases, infrastructure management, application development, and security.
We used DynamoDB, a NoSQL database system, to store and maintain all our databases. We also took advantage of the Cognito service

from AWS, which manages user access points to mobile apps.

Why Use AWS?

AWS provides high-security measures to safeguard all its data and encrypted passwords. It is also free to use for research-oriented works and cheap for business-ready products considering its service. Moreover, AWS provides flexibility, scalability, elasticity, and openness to its service offer. Moreover, with the help of AWS CLI, it is easy to implement all back-end functionalities for an actual short period.

3.2.3 Google MAP API

Google Maps is Google's web mapping platform and consumer application. It provides satellite images, aerial photography, street maps, 360° interactive panoramic street views, real-time traffic conditions, and route planning for travel by foot, vehicle, air, and public transportation. The radar image in Google Maps is a "top-down" or bird's-eye perspective; some of the elevated footage of cities is aerial photos taken from airplanes flying at 800 to 1,500 feet (240 to 460 m), while the majority of other imagery is from satellites. Much of the accessible satellite photos are less than three years old and are routinely updated. Google Maps initially employed a variation of the Mercator projection, making it impossible to represent places near the poles correctly. The desktop version of Google Maps was upgraded in August 2018 to include a 3D globe. In the options, it may still go back to the 2D map.

Why Use Google Maps?

Google Map is 70% more popular than any other MAP API, and 77% of smartphone users use this service to navigate. Google MAP provides 25% more precise direction, 20% preferred features than any other MAP APIs. Google MAP is free to use for a limited usage which perfectly suits our needs. Google MAP is reliable, vastly documented, easy to use, and accurate.

3.3 Core of the Algorithm

Algorithms are called the heart of the code. We considered almost all possible situations and built the algorithm to predict the rickshaw fare accurately. We used reverse engineering and trial and error to reach our final algorithm. It is important to note that this algorithm does not work only on rickshaws but on every other vehicle as this is a fare prediction system. This algorithm has many parts which need to be understood separately.

3.3.1 Distance

When we set a pickup and destination location on MAP, then a route and distance are shown by the Google Map API. The distance is multiplied by a fixed mileage which is a constant value that can vary from place to place. The mileage is considered 5 takas per kilometer in our app because we made this app for a particular place named Dhaka City. We took 2, 3, 4, 5, 6, 7, 8, 9, and 10 taka mileage test cases. 5 taka reflected a 97% close fare approximation in comparison to the actual fare. We considered all the other variables constant and vary only this mileage constant among those test cases and input it to the final equation from which we came to take 5 taka mileage according to Dhaka city. It may vary the value of mileage in other areas.

$$\mathrm{F_D} = \mathrm{D^*M_M} \tag{3.1}$$

Here, the only fare based on distance, F_D, depends on the multiplication of distance (D) and the fare per kilometer M_M.

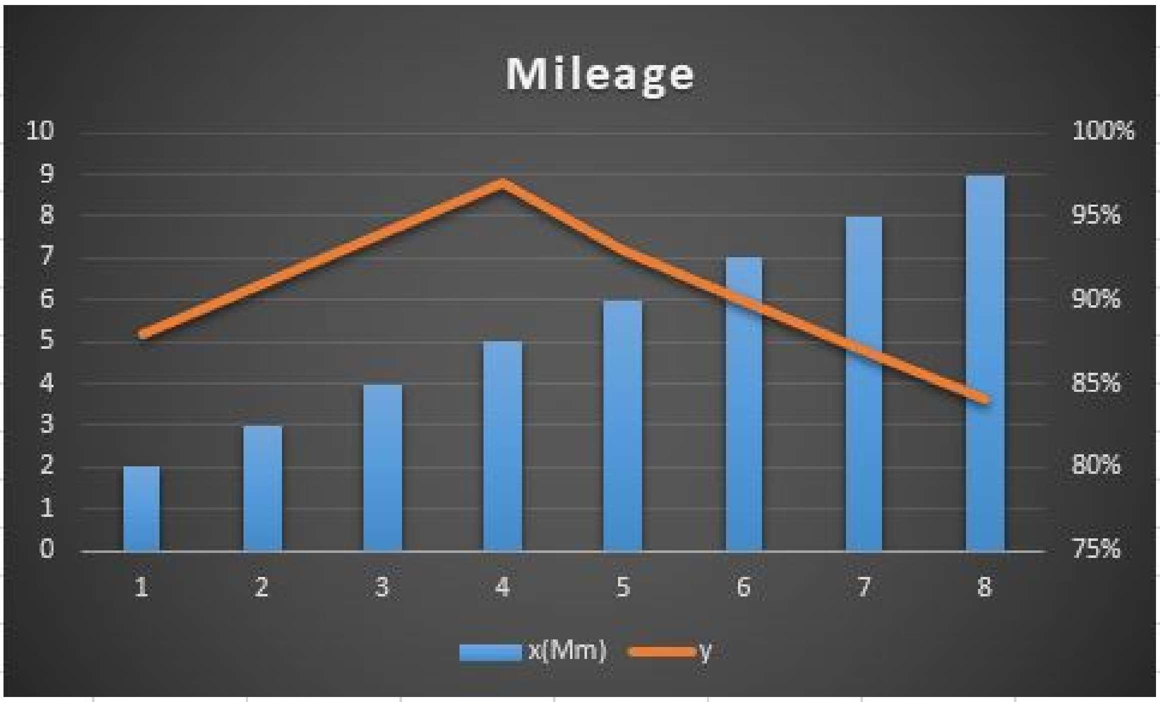


Figure 3.2: Mileage graph

3.3.2 Surge Multiplier

It is the multiplier depending on the demand (requests) and supply (availability). These works are based on a certain radius of a request at the specific request time that the passenger makes. If a ride request is made from a passenger, his GPS location is tracked and marked as a circle's center point. Then, within 200 meters radius, all available rickshaws are selected, and their total value is considered as S_{200} or the supply of rickshaws within a 200-meter radius. The weight of the rickshaw's availability(supply) within 200 meters, S_{200} is ma as 0.75 marked as the rickshaw is close to the passenger. Considering the rickshaw puller has to cover less distance to reach the pickup point.

On the other hand, due to more distance to cover for a rickshaw puller, rickshaws within 500 meters have a weight of 1.25, which is found similarly by considering request location as a center point and finding rickshaws within 500 meters. Please note that this rickshaw availability is 500 meters(Supply), S_{500} is considered from 201 meters to 500 meters as we considered 0-200 meters in the S_{200} part.
These 0.75 and 1.25 values are not taken arbitrarily. These values were diagnosed by applying specific techniques. At first, a real dataset of 68 real-life scenarios was taken. Then the multiplier value of 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, and 2 were placed in each of the S_{500} and S_{200} and inputted into the final total fare to observe which one reflects a more real-life fare system. 0.75 for S_{200} produced 6.7% better results than all other values, and similarly, 1.25 for S_{500} produced 7.4% better results than others. These constant values can be changed and vary from place to place.

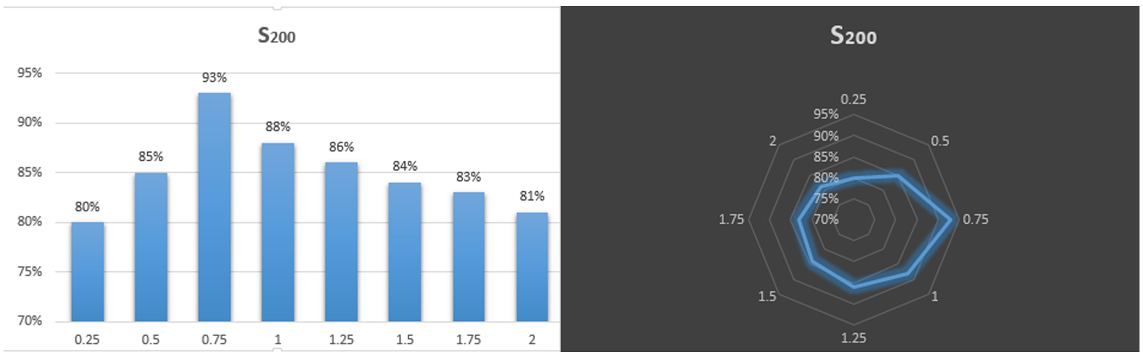


Figure 3.3: S200

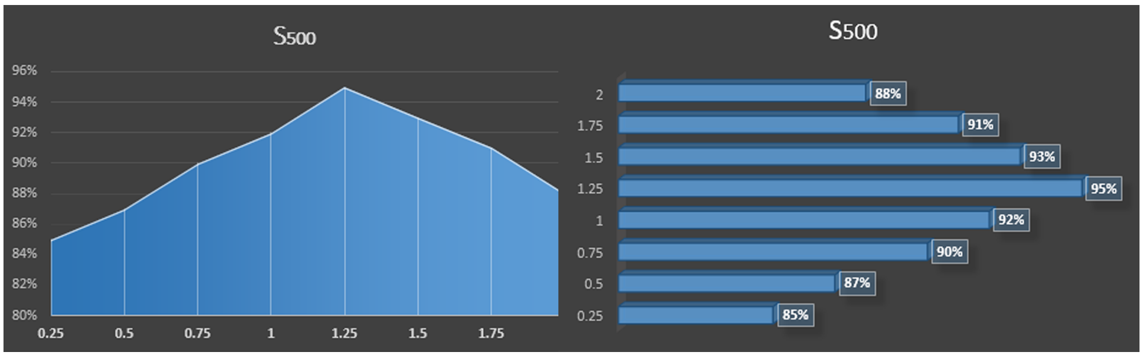


Figure 3.4: S500

Finally, demand is drawn similarly like considering a request to be center and pulling all the other active requests on a specific time within 500 meters radius, which is named demand of rickshaws within 500 meters radius or D_{500}. Finally, demand

and supply relation, demand is divided by the supply to form a relation and surge multiplier. So the surge multiplier of our system or M_S is shown below:

$$\mathrm{M_S} = \mathrm{D_{500}} / \left[(\mathrm{S_{500}} * 1.25) + (\mathrm{S_{200}} * 0.75)\right] \tag{3.2}$$

D_{500} indicates the requests in a 500-meter radius, and S_{500} is the available number of rickshaws in a 201-500 meter radius, likely S_{200} passes the number of rickshaws in a 200-meter radius. These values are pulled using Google Map API in each iteration.

3.3.3 Time Based Fare

This part calculates the fare based on the time of the ride dynamically. Traffic jams are also considered in this part. At time 0 when the ride starts, the value of timed fare, F_T, is set to be 0. As the time flows and it becomes 1,2,3,4,.., The fare changes. The current location is tracked by calling Map API reach time in 60 seconds or 1 minute. This time frame could be lowered, but it will cause an API call. Due to limitations of API call in the trial version, we set it to be 60 seconds but implementing it in an industry level application, and we should consider the time delay to be 1 second or less.

The current location is marked as D, and after 60 seconds, the D becomes the previous location marked as a temp. So in each iteration, we store the previous location and get the current location. Generally, rickshaws can cover 130-180 meters per minute. So this terminology was used to find out if a traffic jam was present or not. A variable was introduced named traveled distance which is found by calculating the distance covered every 60 seconds. Moreover, 100 meters were considered to find out if the rickshaw is currently in a traffic jam or not. Inside the implementation, if the rickshaw does not move 100 meters within 1 minute, that is marked as a traffic jam.

```
While ( ride continues ){
    travelledDistance = D – temp;
    if (time $\leq$ 120 minutes){
        if ( travelledDistance < 100 meter){
                $F_T = F_T$ - 0.5;
        }
        $F_T = F_T + 1$;
    }
    else if ( 140 minutes $\leq$ time $\leq$ 180 minutes){
        $F_T = F_T + 0.8$;
    }
    else if ( 200 minutes $\leq$ time $\leq$ 240 minutes){
        $F_T = F_T + 0.6$;
    }
    temp=D
    Delay(60sec)
}
```

Here traveled distance is the variable that will track the difference between the currently updated distance and the temporary distance. For each minute, 1 taka will be added with the fare, and if the traveled distance is less than 100 meters, then we consider that as a traffic jam and increase the fare by 0.5 takas for each minute while being stuck in a traffic jam. Lastly, when the ride is over, and the rickshaw puller presses the "End Ride" button on the driver side app, the loop ends, and we get the final value of F_T. These iterations will occur in every step and dynamically change the F_T, which results in changing the total fare.

We considered a particular case while calculating the F_T. Generally, a rickshaw ride lasts less than 120 minutes. Nevertheless, if the ride lasts more than 120 minutes and less, then the per minute fare increase must be reduced to help the user. This type of inconvenience can only occur a few times. However, for the 120-140 minutes, no extra fare will be charged. However, from 140minutes to 180 minutes, the fare will again start to increase at the rate of 0.8 instead of 1. Then again, 20 minutes no fare charge and again 0.6 increase rate. Therefore, we considered the ride to be 240 minutes as it is not practical for a rickshaw ride to be more than 240 minutes or 6 hours.

When showing the approximate value, we take the pickup to the destination, reaching a time approximation provided by Google Map API. It is not an actual one, just an approximate fare while the ride starts, so we multiply the time with one, which produces per minute one taka rate. This thing is only shown in the approximate fare calculation period, but the strategy mentioned above is used during the actual ride.

3.3.4 Weather-based multiplier

The fare will vary based on the weather condition. Considering the weather in Dhaka, Bangladesh, it was found that rickshaw pullers tend to demand more fare for a ride during two times. Those two weather conditions are during rain and too much hot or cold weather. As rickshaw pulling is physically challenging, the extra hard work should be paid off. For the first condition, during rain, we found out that, considering seventy-seven real-life scenarios, rickshaw pullers tend to demand 1.2 times more fare on average. Secondly, when it comes to hot or cold weather, the demand is 1.1 times higher than the actual fare. So keeping that nature of demanding more than the actual fare, a simple multiplier was introduced to reflect this phenomenon. Weather condition is generated by weather API to learn about the current weather on a specific time. The weather multiplier, M_W, works like this:

```
if (weather = rainy){
    $M_W = 1.20$;
}
else if (temperature $<$ 18  temperature $>$ 32){
    $M_W = 1.10$;
} else {
    $M_W$=1;
}
```

Here, If it was raining when the request was made, then the temperature multiplier, M_W, is set to be 1.2. Secondly, We considered 18 degrees celsius temperature as the cold weather floor temperature. If the current temperature is lower than 18 degrees celsius, then the temperature multiplier, M_W, is set to be 1.1. Similarly, we considered 32 degrees celsius as the ceiling temperature threshold. If the current temperature is higher than 32 degrees celsius, then the temperature multiplier, M_W, is set to be 1.1. Finally, we keep the multiplier, $M_W = 1$, if the weather condition does not fall under any mentioned conditions.

3.4 The Final Equation

The raw data were processed using above mentioned equations before inserting them for realistic simulation. From there, the algorithm calculates and gets all its necessary variable values to calculate the final or total fare. We get the values of fare for the distance F_D, the multiplier for the demand and supply condition M_S, fare for the time F_T, and multiplier for the weather M_W variables, and these values are passed to calculate the final fare. The equation of the final fare is:

$$[\text{Base Fare } + (F_D * M_S) + F_T]^* \, M_W \tag{3.3}$$

Here, the Base fare can vary from place to place. For example, in Dhaka, Bangladesh, the base fare is set to be 10 takas. When the request is made initially, an approximate calculated fare is shown to both users and a rider app. This approximate fare is calculated where all the variables are initiated using google API approximation values. However, when the ride starts, all the real-time values are implanted into this equation, and this equation runs and calculates real-time fare value in each iteration for 60 seconds to show the current real-time fare. This fare is essential as if the user or driver wants to end the ride, this value will be shown as the final fare.

Chapter 4

Application Preview

4.1 Front-End

Two Android apps include both the user side and driver side. So the name of the apps are:

1. Rickshaw Mama App: Passenger app
2. Rickshaw Mama Driver App: Rickshaw puller app

Firstly, there is login/signup functionality for both user and driver apps. Then the user can set pick up and destination and request for a ride. The request is pinged at all nearby driver apps, and any of the drivers can accept the request. Meanwhile, approximate fare and all other necessary information are shown to both user and driver. When the ride starts, all the real-time fare monitoring, API calls, and calculations are done on the driver app. Both apps show the route towards destination, and it is being updated on each API call on the driver side. There are four different states in the whole system. They are:

1. **Lazy State:** If the user app is opened but still not placing a request or the driver app is offline, then the apps are in this state.
2. **Request State:** A request for a ride has been made from the user, but it is currently pending as no driver accepted the request.
3. **Picking Up State:** A driver has already accepted the request and is coming to pick up the passenger or user.
4. **Riding State:** The driver picked up the passenger, and the ride originally began to reach the destination from the pickup location.
5. **Completion State:** The driver successfully ends the ride, and the final fare is shown to both sides to collect the desired fare. After this, both apps load the home page.

4.1.1 Sign in and Sign up Page:

The initial page for both apps where the user or driver has to fill up their login/signup form and submit it. For logging in, "Username" and "Password" are essential. Signing up depends on the new email address. The driver has to provide two more data named "Vehicle Type" and "License Number." Vehicle type is introduced to future widening improvements. However, "Name," "Email," "Password," "Mobile no." are standard information to signup in both driver and user side apps.

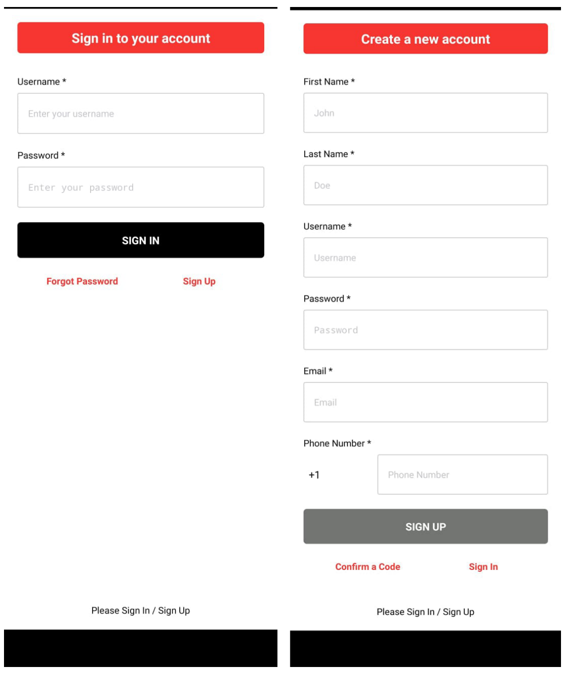


Figure 4.1: Sign in and sign up Page.

4.1.2 Home Page:

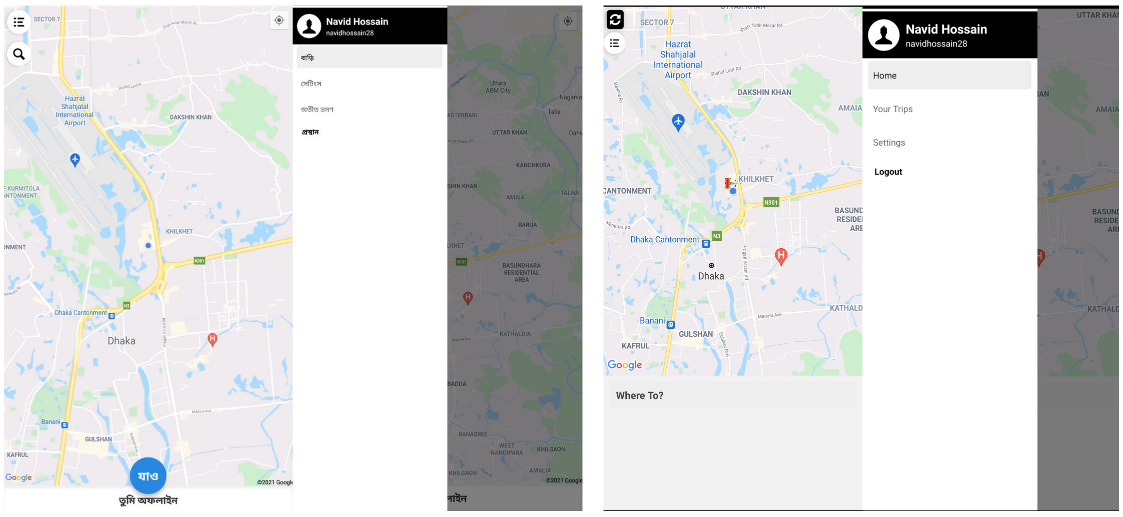
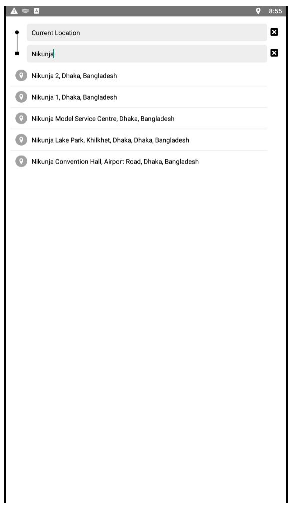


Figure 4.2: Home and side bar of driver (left) and user (right) application.

After logging in, both apps load the home page, which includes a real-time integrated map pointing to their current location. Please note that both of the apps need to access the GPS permissions on android devices. Both of the apps have an app drawer on the left side of the screen. In the user app, there is a request ride button on the bottom side of the screen. There is a search icon at the top left corner in the driver app that pulls all the pending requests from the server. If the driver app gets a request, it is directly shown on the main screen, including all the necessary information about the ride, payment, and passenger details. The driver app has a ”Go Offline” button on the bottom side of the screen, which toggles the driver from online to offline.

4.1.3 Pickup and Destination Selection Page(Only on the user side):



8:55

Current Location

Nikunja

Nikunja 2, Dhaka, Bangladesh

Nikunja 1, Dhaka, Bangladesh

Nikunja Model Service Centre, Dhaka, Bangladesh

Nikunja Lake Park, Khilkhet, Dhaka, Dhaka, Bangladesh

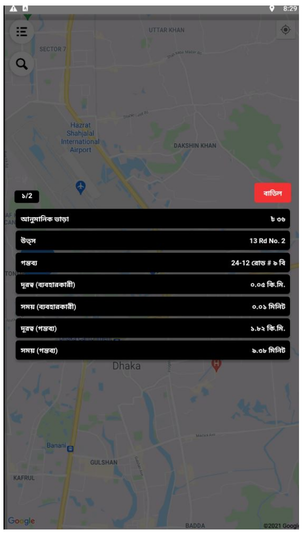
Nikunja Convention Hall, Airport Road, Dhaka, Bangladesh

Figure 4.3: Pickup and destination selection.

This page is only available on the user-side app. If the user wants to place a riding request, it loads a new page after pressing the ”where to?” button. This new page has only an input box to fill up, one is for pickup location, and another one is for the destination location. The user can set his current location as pickup directly. The auto search option was used to find places using Google Autofill API. Also, users can manually select their current location.

4.1.4 Available requests and accepting the request:

After selecting pickup and destination points by the users will show the requests to the online driver app selecting the pickup and destination points. Furthermore, the drivers can see the available recommendations from the driver application and pick one by choice. However, if any driver does not want to accept the request, he/she can decline the request also. Drivers can see their user origin, destination, the distance of user and destination, and the estimated fare. This will show the status and accept or decline user requests on the next page.

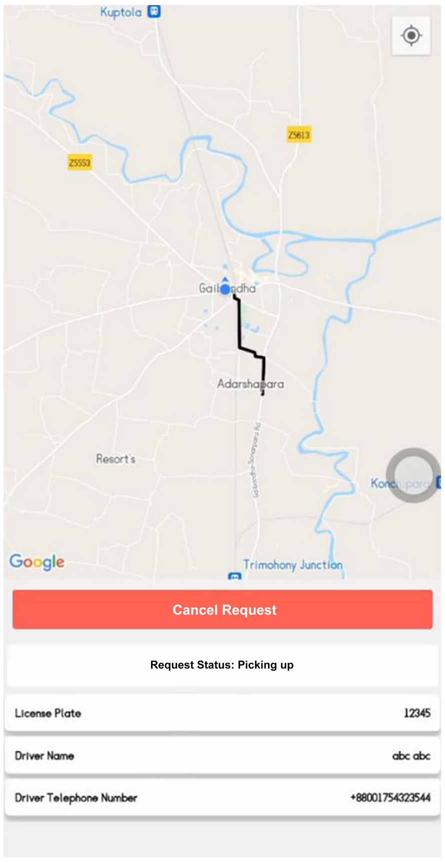


১/2

আনুমানিক ভাড়া	৳ ৩৬
উত্স	13 Rd No. 2
গন্তব্য	24-12 রোড # ৯ বি
দূরত্ব (ব্যবহারকারী)	০.০৫ কি.মি.
সময় (ব্যবহারকারী)	০.০১ মিনিট
দূরত্ব (গন্তব্য)	১.৮২ কি.মি.
সময় (গন্তব্য)	৯.০৮ মিনিট

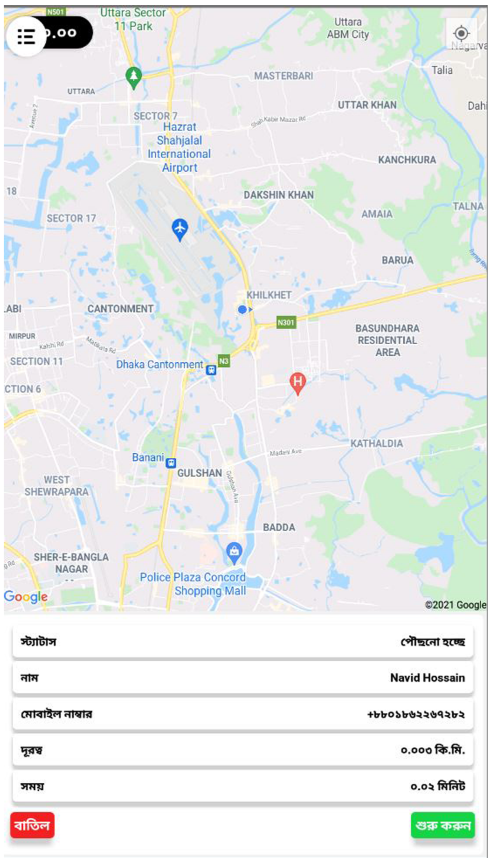
Figure 4.4: Available requests.

4.1.5 Picking Up State:



Request Status: Picking up

License Plate	12345
Driver Name	abc abc
Driver Telephone Number	+88001754323544



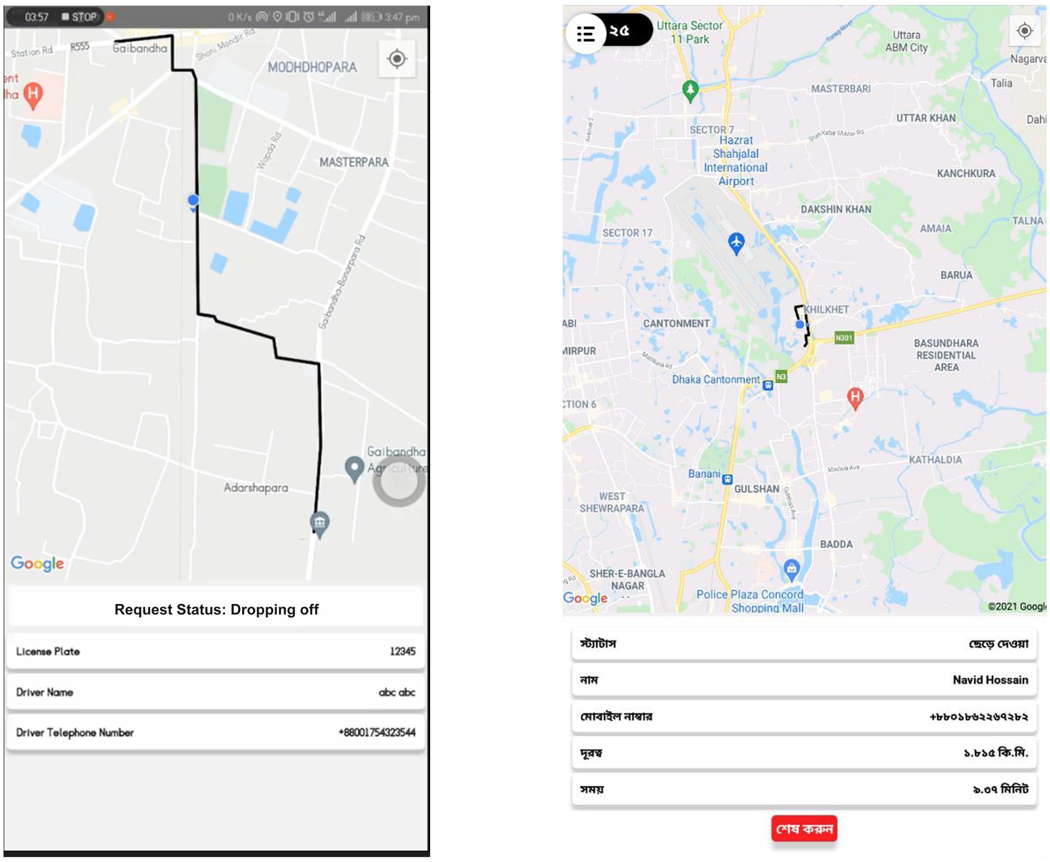
স্ট্যাটাস	পৌছনো হচ্ছে
নাম	Navid Hossain
মোবাইল নাম্বার	+৮৮০১৮৬২২৬৭২৮২
দূরত্ব	০.০০০ কি.মি.
সময়	০.০২ মিনিট

Figure 4.5: Picking Up State of user (left) and driver (right) application.

After selecting the location, it will show the estimated fare. It will also show the weather, duration, and distance. Then users need to select the River in the River icon and click Confirm Ride. After that, it will lead to a new page with Cancel Request Button and Order status. Users can cancel the request anytime by clicking the cancel button. Order status indicates whether the order is accepted or not. If the order is not accepted, then it will show order status idle. If the order is accepted, it will show order status, picking up additional information like driver name, driver telephone number, and license plate.

4.1.6 Riding State:

After starting the ride, the driver will be presented with a UI on the driver page that consists of ride status, name, telephone, distance, and duration. Here, the name and telephone indicate the passenger's name and his/her telephone number. Moreover, the distance points to the numeric value distance, which is left to reach the passenger's destination. Duration also shows the time that it might take to reach the destination. On the other hand, the passenger will see ride status, license plate, driver name, and telephone number on the rider page. The license plate shows the plate number to the passenger, which will help the passenger find out / verify the



Request Status: Dropping off

License Plate	12345
Driver Name	abc abc
Driver Telephone Number	+88001754323544

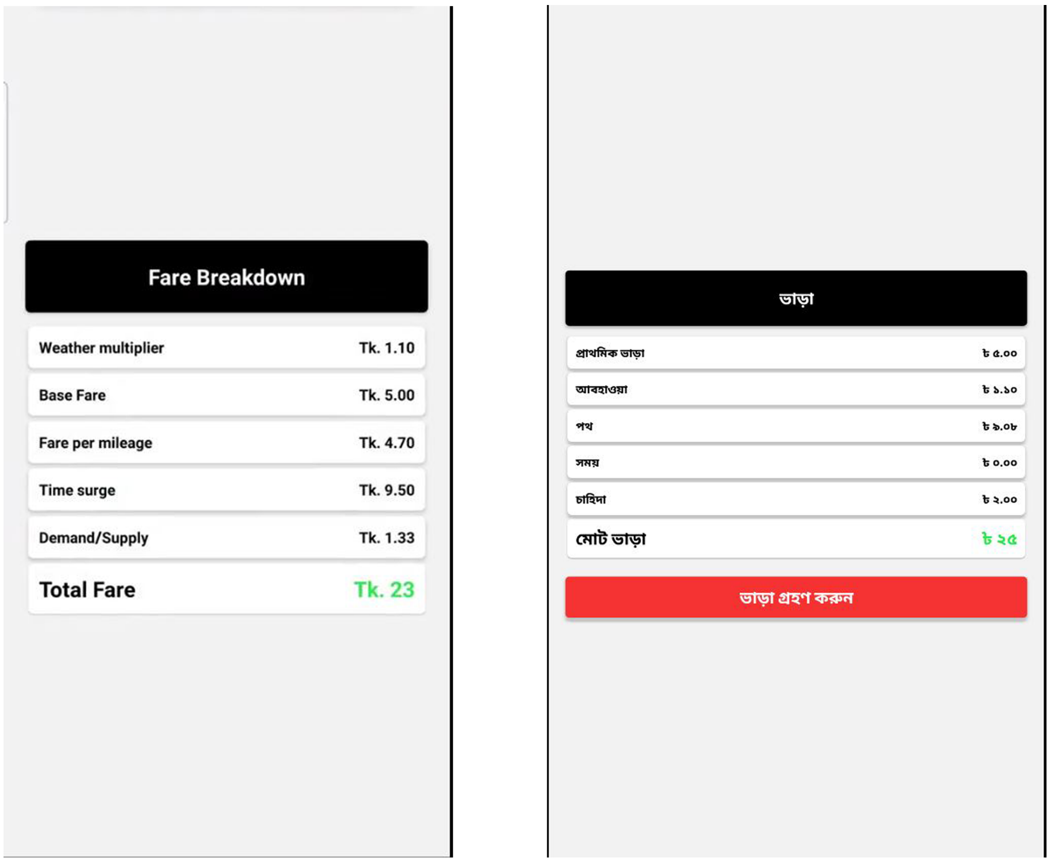
স্ট্যাটাস	ছেড়ে দেওয়া
নাম	Navid Hossain
মোবাইল নাম্বার	+৮৮০১৮৬২২৬৭২৮২
দূরত্ব	১.৮১৫ কি.মি.
সময়	৯.৩৭ মিনিট

Figure 4.6: Riding State of user (left) and driver (right) application.

rickshaw. Again, the passenger has the driver's number and name, which will help the passenger connect with the rider, and also, for future security, this information will be helpful. The passenger will be able to see his/her location on the map and quickly identify which road the driver follows to reach the destination.

4.1.7 Ride Complete State:

After reaching the destination, they will see a Fare Breakdown page on both driver and user app. The fare breakdown shows the total fare based on multiple factors like weather multiplier, base fare, fare per mileage, demand/supply, and time surge. After calculating numerous factors then it gives the total output fare. When the driver clicks the Receive button, it will notify the user that the payment is received and reload to the starting page for the user app and driver app. The driver can again take the order from the user, and the user can again request a ride.



Fare Breakdown

Weather multiplier	Tk. 1.10
Base Fare	Tk. 5.00
Fare per mileage	Tk. 4.70
Time surge	Tk. 9.50
Demand/Supply	Tk. 1.33
Total Fare	**Tk. 23**

ভাড়া

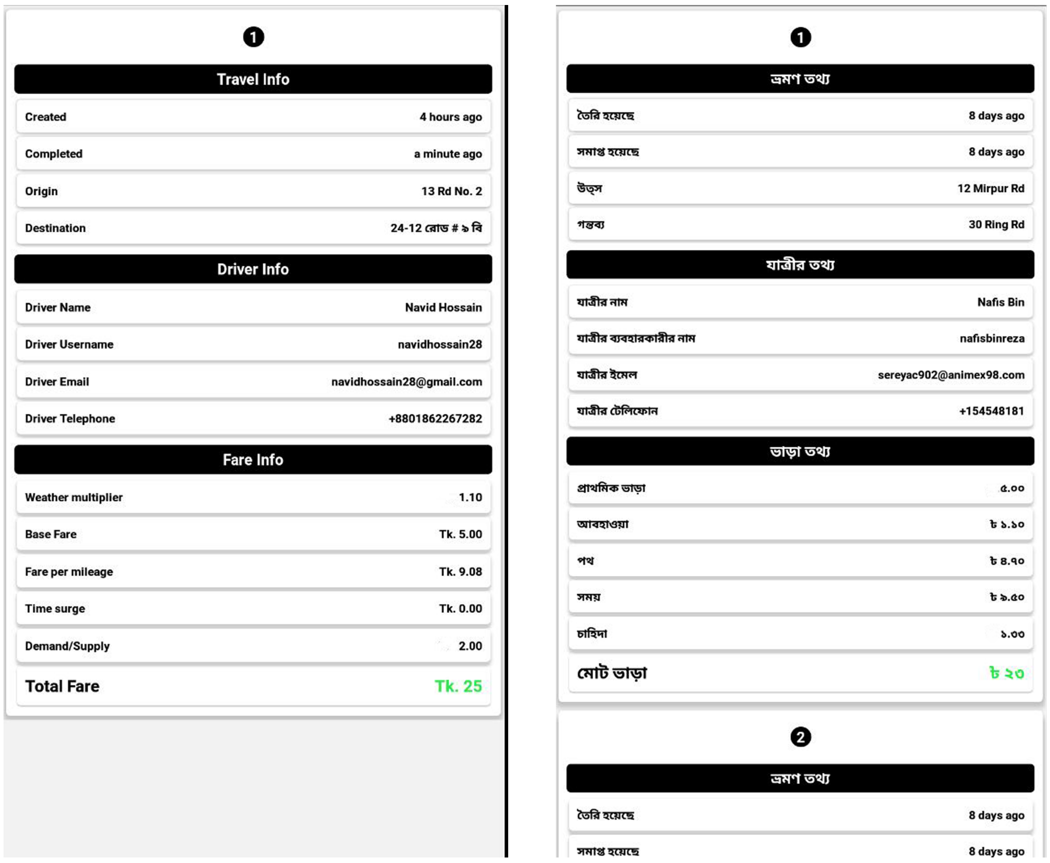
প্রাথমিক ভাড়া	৳ ৫.০০
আবহাওয়া	৳ ১.১০
পথ	৳ ৯.০৮
সময়	৳ ০.০০
চাহিদা	৳ ২.০০
মোট ভাড়া	**৳ ২৫**

ভাড়া গ্রহণ করুন

Figure 4.7: Ride Complete State of user (left) and driver (right) application.

4.1.8 Previous trips:

There is a history page implemented with both the driver-side and user-side app. The primary purpose of this page is to show all the previous trip information like pickup location, destination location, timestamp of request creation and completion, and fare breakdown. Moreover, in the user app, the user can also see the details like name, username, email, and the contact number of the particular rickshaw puller. Finally, in the driver app, the driver can check the details of the specific user of a selected trip, User side shows this page in English and driver app shows this page in Bengali.



1

Travel Info

Created	4 hours ago
Completed	a minute ago
Origin	13 Rd No. 2
Destination	24-12 রোড # ৯ বি

Driver Info

Driver Name	Navid Hossain
Driver Username	navidhossain28
Driver Email	navidhossain28@gmail.com
Driver Telephone	+8801862267282

Fare Info

Weather multiplier	1.10
Base Fare	Tk. 5.00
Fare per mileage	Tk. 9.08
Time surge	Tk. 0.00
Demand/Supply	2.00
Total Fare	**Tk. 25**

1

ভ্রমণ তথ্য

তৈরি হয়েছে	8 days ago
সমাপ্ত হয়েছে	8 days ago
উত্স	12 Mirpur Rd
গন্তব্য	30 Ring Rd

যাত্রীর তথ্য

যাত্রীর নাম	Nafis Bin
যাত্রীর ব্যবহারকারীর নাম	nafisbinreza
যাত্রীর ইমেল	sereyac902@animex98.com
যাত্রীর টেলিফোন	+154548181

ভাড়া তথ্য

প্রাথমিক ভাড়া	৫.০০
আবহাওয়া	৳ ১.১০
পথ	৳ ৪.৭০
সময়	৳ ৯.৫০
চাহিদা	১.৩৩
মোট ভাড়া	**৳ ২৩**

2

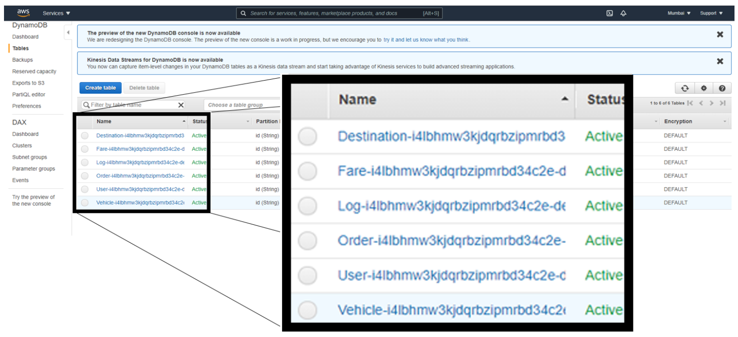
ভ্রমণ তথ্য

তৈরি হয়েছে	8 days ago
সমাপ্ত হয়েছে	8 days ago

Figure 4.8: Previous trips of user (left) and driver (right) application.

4.2 Back-End

For the back-end implementation, we used Amazon AWS, where we used DynamoDB as our NoSQL database. Moreover, we also used the Cognito service, a secure encryption method to secure and manage passwords for our app. Here is a brief description of these implementations:



The preview of the new DynamoDB console is now available
We are redesigning the DynamoDB console. The preview of the new console is a work in progress, but we encourage you to try it and let us know what you think.

Kinesis Data Streams for DynamoDB is now available
You now can capture item-level changes in your DynamoDB tables as a Kinesis data stream and start taking advantage of Kinesis services to build advanced streaming applications.

Name	Status	Partition	Encryption
Destination-i4lbhmw3kjdqrbzipmrbd3	Active	id (String)	DEFAULT
Fare-i4lbhmw3kjdqrbzipmrbd34c2e-d	Active	id (String)	DEFAULT
Log-i4lbhmw3kjdqrbzipmrbd34c2e-de	Active	id (String)	DEFAULT
Order-i4lbhmw3kjdqrbzipmrbd34c2e-	Active	id (String)	DEFAULT
User-i4lbhmw3kjdqrbzipmrbd34c2e-c	Active	id (String)	DEFAULT
Vehicle-i4lbhmw3kjdqrbzipmrbd34c2e	Active	id (String)	DEFAULT

Figure 4.9: DynamoDB tables.

4.2.1 DynamoDB

In the database part, we needed to create some tables to organize, manage, and optimize all the necessary data for our application. Brief description of our tables are mentioned below:

1. Destination: In this table, unique id, user id, the address of the destination, altitude of the map, and creation time are the columns—this table updates when the user makes a request.

2. Fare: In the fare table, we have the breakdown of complete fare statements mentioning for which fare, how much money was allocated in each section to generate the total fare. Here we have columns like fare for distance, the fare for surge multiplier, the fare for weather, the fare for time, fare id, etc.

3. Log: In the log table, we have all the log files. When a user requests, this request is passed to nearby drivers. At each timestamp, to whom the request was sent, if the request was accepted or not, this type of data is stored on the log table.

4. Order: In this table, all the necessary information of a request is kept, like which user made the request, which driver responded, pickup and destination location, timestamp, date, etc., are kept.

5. User: In the user table, we store all the information of a user or driver, like name, phone number, email address, etc., based on a unique user id.

6. Vehicle: in this table, we store the vehicle type and license plate number of a particular vehicle, linked to the driver information.

4.2.2 Cognito

Cognito is a scalable and secured cloud-based encryption system. It is effortless to use and maintain. Cognito creates a user pool that provides signup and sign-in services. It also features social sign-in with Facebook, Google, etc., so it would be easy to implement in the future. Cognitive comes with a built-in user directory management and profile system. For security purposes, the system employs multi-factor authentication; checks compromised credentials, takeover protection, email verification, etc. Cognito also customizes procedures and user migration by triggering AWS Lambda.

Chapter 5

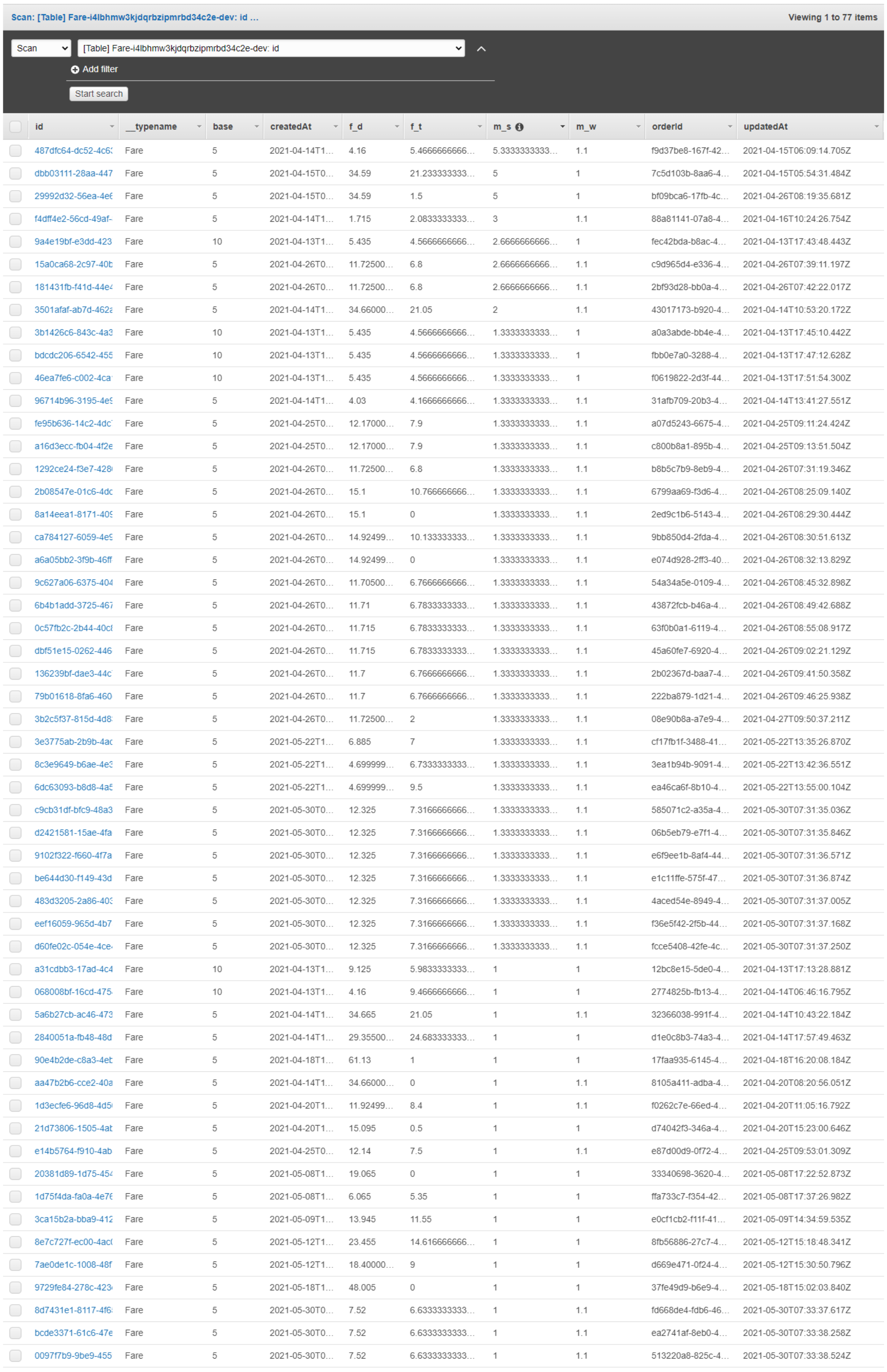
Result and Analysis

5.1 Result

After running 77 real test-drive of the apps, we populated our database with test-drive data. Meanwhile, we noted all the actual fares of every ride. After that, a comparison between the demanded fare and the app-generated fare was calculated. The output performed as expected by having 95% accuracy in reflecting the real-life fare demanded by the rickshaw pullers.

We can see that the fares our app outputs very differently based on F_D, M_S, F_T, and M_W. If any one of these variables changes, then the output also varies such that it can project almost similar to real value adjusting itself. For example, suppose the distance is 10 kilometers. In that case, ten requests for rickshaws in a particular area, two rickshaws within 500 meters range of each request, two rickshaws within 200 meters of the request making total supply $(2 + 2) = 4$, the riding time 10 minutes, and in hot weather produces 148.50 taka fare which is almost similar to our real-life scenario of being 150 takas of the genuine rickshaw fare for such a scenario.

The test-run fare data from a table named "fare" in the DynamoDB database is shown below. This table is populated while the requests are completed. This table has all the breakdown of fare according to the sections. We can see from the table that it has columns named "_typename," which correspond to identify it as a fare. In the "base" column, the base fare of a ride is shown. The next "createdAt" column presents the creation time of this fare breakdown. Sequentially, "f_d", "f_t", "m_s", and "m_w" correspond to our algorithm's core variables. Then a unique "order_id" is given, and "updated_at" represents the update timestamp.


Scan: [Table] Fare-i4lbhmw3kjdqrbzipmrbd34c2e-dev: id ...

Viewing 1 to 77 items

id	__typename	base	createdAt	f_d	f_t	m_s	m_w	orderId	updatedAt
487dfc64-dc52-4c6	Fare	5	2021-04-14T1...	4.16	5.4666666666...	5.3333333333...	1.1	f9d37be8-167f-42...	2021-04-15T06:09:14.705Z
dbb03111-28aa-447	Fare	5	2021-04-15T0...	34.59	21.233333333...	5	1	7c5d103b-8aa6-4...	2021-04-15T05:54:31.484Z
29992d32-56ea-4e6	Fare	5	2021-04-15T0...	34.59	1.5	5	1	bf09bca6-17fb-4c...	2021-04-26T08:19:35.681Z
f4dff4e2-56cd-49af-	Fare	5	2021-04-14T1...	1.715	2.0833333333...	3	1.1	88a81141-07a8-4...	2021-04-16T10:24:26.754Z
9a4e19bf-e3dd-423	Fare	10	2021-04-13T1...	5.435	4.5666666666...	2.6666666666...	1	fec42bda-b8ac-4...	2021-04-13T17:43:48.443Z
15a0ca68-2c97-40b	Fare	5	2021-04-26T0...	11.72500...	6.8	2.6666666666...	1.1	c9d965d4-e336-4...	2021-04-26T07:39:11.197Z
181431fb-f41d-44e4	Fare	5	2021-04-26T0...	11.72500...	6.8	2.6666666666...	1.1	2bf93d28-bb0a-4...	2021-04-26T07:42:22.017Z
3501afaf-ab7d-462a	Fare	5	2021-04-14T1...	34.66000...	21.05	2	1.1	43017173-b920-4...	2021-04-14T10:53:20.172Z
3b1426c6-843c-4a3	Fare	10	2021-04-13T1...	5.435	4.5666666666...	1.3333333333...	1	a0a3abde-bb4e-4...	2021-04-13T17:45:10.442Z
bdcdc206-6542-455	Fare	10	2021-04-13T1...	5.435	4.5666666666...	1.3333333333...	1	fbb0e7a0-3288-4...	2021-04-13T17:47:12.628Z
46ea7fe6-c002-4ca	Fare	10	2021-04-13T1...	5.435	4.5666666666...	1.3333333333...	1	f0619822-2d3f-44...	2021-04-13T17:51:54.300Z
96714b96-3195-4e9	Fare	5	2021-04-14T1...	4.03	4.1666666666...	1.3333333333...	1.1	31afb709-20b3-4...	2021-04-14T13:41:27.551Z
fe95b636-14c2-4dc	Fare	5	2021-04-25T0...	12.17000...	7.9	1.3333333333...	1.1	a07d5243-6675-4...	2021-04-25T09:11:24.424Z
a16d3ecc-fb04-4f2e	Fare	5	2021-04-25T0...	12.17000...	7.9	1.3333333333...	1.1	c800b8a1-895b-4...	2021-04-25T09:13:51.504Z
1292ce24-f3e7-428	Fare	5	2021-04-26T0...	11.72500...	6.8	1.3333333333...	1.1	b8b5c7b9-8eb9-4...	2021-04-26T07:31:19.346Z
2b08547e-01c6-4dc	Fare	5	2021-04-26T0...	15.1	10.766666666...	1.3333333333...	1.1	6799aa69-f3d6-4...	2021-04-26T08:25:09.140Z
8a14eea1-8171-409	Fare	5	2021-04-26T0...	15.1	0	1.3333333333...	1.1	2ed9c1b6-5143-4...	2021-04-26T08:29:30.444Z
ca784127-6059-4e9	Fare	5	2021-04-26T0...	14.92499...	10.133333333...	1.3333333333...	1.1	9bb850d4-2fda-4...	2021-04-26T08:30:51.613Z
a6a05bb2-3f9b-46ff	Fare	5	2021-04-26T0...	14.92499...	0	1.3333333333...	1.1	e074d928-2ff3-40...	2021-04-26T08:32:13.829Z
9c627a06-6375-404	Fare	5	2021-04-26T0...	11.70500...	6.7666666666...	1.3333333333...	1.1	54a34a5e-0109-4...	2021-04-26T08:45:32.898Z
6b4b1add-3725-467	Fare	5	2021-04-26T0...	11.71	6.7833333333...	1.3333333333...	1.1	43872fcb-b46a-4...	2021-04-26T08:49:42.688Z
0c57fb2c-2b44-40c	Fare	5	2021-04-26T0...	11.715	6.7833333333...	1.3333333333...	1.1	63f0b0a1-6119-4...	2021-04-26T08:55:08.917Z
dbf51e15-0262-446	Fare	5	2021-04-26T0...	11.715	6.7833333333...	1.3333333333...	1.1	45a60fe7-6920-4...	2021-04-26T09:02:21.129Z
136239bf-dae3-44c	Fare	5	2021-04-26T0...	11.7	6.7666666666...	1.3333333333...	1.1	2b02367d-baa7-4...	2021-04-26T09:41:50.358Z
79b01618-8fa6-460	Fare	5	2021-04-26T0...	11.7	6.7666666666...	1.3333333333...	1.1	222ba879-1d21-4...	2021-04-26T09:46:25.938Z
3b2c5f37-815d-4d8	Fare	5	2021-04-26T0...	11.72500...	2	1.3333333333...	1.1	08e90b8a-a7e9-4...	2021-04-27T09:50:37.211Z
3e3775ab-2b9b-4ac	Fare	5	2021-05-22T1...	6.885	7	1.3333333333...	1.1	cf17fb1f-3488-41...	2021-05-22T13:35:26.870Z
8c3e9649-b6ae-4e3	Fare	5	2021-05-22T1...	4.699999...	6.7333333333...	1.3333333333...	1.1	3ea1b94b-9091-4...	2021-05-22T13:42:36.551Z
6dc63093-b8d8-4a5	Fare	5	2021-05-22T1...	4.699999...	9.5	1.3333333333...	1.1	ea46ca6f-8b10-4...	2021-05-22T13:55:00.104Z
c9cb31df-bfc9-48a3	Fare	5	2021-05-30T0...	12.325	7.3166666666...	1.3333333333...	1.1	585071c2-a35a-4...	2021-05-30T07:31:35.036Z
d2421581-15ae-4fa	Fare	5	2021-05-30T0...	12.325	7.3166666666...	1.3333333333...	1.1	06b5eb79-e7f1-4...	2021-05-30T07:31:35.846Z
9102f322-f660-4f7a	Fare	5	2021-05-30T0...	12.325	7.3166666666...	1.3333333333...	1.1	e6f9ee1b-8af4-44...	2021-05-30T07:31:36.571Z
be644d30-f149-43d	Fare	5	2021-05-30T0...	12.325	7.3166666666...	1.3333333333...	1.1	e1c11ffe-575f-47...	2021-05-30T07:31:36.874Z
483d3205-2a86-403	Fare	5	2021-05-30T0...	12.325	7.3166666666...	1.3333333333...	1.1	4aced54e-8949-4...	2021-05-30T07:31:37.005Z
eef16059-965d-4b7	Fare	5	2021-05-30T0...	12.325	7.3166666666...	1.3333333333...	1.1	f36e5f42-2f5b-44...	2021-05-30T07:31:37.168Z
d60fe02c-054e-4ce	Fare	5	2021-05-30T0...	12.325	7.3166666666...	1.3333333333...	1.1	fcce5408-42fe-4c...	2021-05-30T07:31:37.250Z
a31cdbb3-17ad-4c4	Fare	10	2021-04-13T1...	9.125	5.9833333333...	1	1	12bc8e15-5de0-4...	2021-04-13T17:13:28.881Z
068008bf-16cd-475	Fare	10	2021-04-13T1...	4.16	9.4666666666...	1	1	2774825b-fb13-4...	2021-04-14T06:46:16.795Z
5a6b27cb-ac46-473	Fare	5	2021-04-14T1...	34.665	21.05	1	1.1	32366038-991f-4...	2021-04-14T10:43:22.184Z
2840051a-fb48-48d	Fare	5	2021-04-14T1...	29.35500...	24.683333333...	1	1	d1e0c8b3-74a3-4...	2021-04-14T17:57:49.463Z
90e4b2de-c8a3-4eb	Fare	5	2021-04-18T1...	61.13	1	1	1	17faa935-6145-4...	2021-04-18T16:20:08.184Z
aa47b2b6-cce2-40a	Fare	5	2021-04-14T1...	34.66000...	0	1	1.1	8105a411-adba-4...	2021-04-20T08:20:56.051Z
1d3ecfe6-96d8-4d5	Fare	5	2021-04-20T1...	11.92499...	8.4	1	1.1	f0262c7e-66ed-4...	2021-04-20T11:05:16.792Z
21d73806-1505-4ab	Fare	5	2021-04-20T1...	15.095	0.5	1	1	d74042f3-346a-4...	2021-04-20T15:23:00.646Z
e14b5764-f910-4ab	Fare	5	2021-04-25T0...	12.14	7.5	1	1.1	e87d00d9-0f72-4...	2021-04-25T09:53:01.309Z
20381d89-1d75-454	Fare	5	2021-05-08T1...	19.065	0	1	1	33340698-3620-4...	2021-05-08T17:22:52.873Z
1d75f4da-fa0a-4e76	Fare	5	2021-05-08T1...	6.065	5.35	1	1	ffa733c7-f354-42...	2021-05-08T17:37:26.982Z
3ca15b2a-bba9-412	Fare	5	2021-05-09T1...	13.945	11.55	1	1	e0cf1cb2-f11f-41...	2021-05-09T14:34:59.535Z
8e7c727f-ec00-4ac	Fare	5	2021-05-12T1...	23.455	14.616666666...	1	1	8fb56886-27c7-4...	2021-05-12T15:18:48.341Z
7ae0de1c-1008-48f	Fare	5	2021-05-12T1...	18.40000...	9	1	1	d669e471-0f24-4...	2021-05-12T15:30:50.796Z
9729fe84-278c-423	Fare	5	2021-05-18T1...	48.005	0	1	1	37fe49d9-b6e9-4...	2021-05-18T15:02:03.840Z
8d7431e1-8117-4f6	Fare	5	2021-05-30T0...	7.52	6.6333333333...	1	1.1	fd668de4-fdb6-46...	2021-05-30T07:33:37.617Z
bcde3371-61c6-47e	Fare	5	2021-05-30T0...	7.52	6.6333333333...	1	1.1	ea2741af-8eb0-4...	2021-05-30T07:33:38.258Z
0097f7b9-9be9-455	Fare	5	2021-05-30T0...	7.52	6.6333333333...	1	1.1	513220a8-825c-4...	2021-05-30T07:33:38.524Z

Figure 5.1: Sample real life data.

5.2 Analysis

The table shown below represents expected output of our algorithm and test output of our app. It also shows the difference between expected and test outputs with error percentage of each test. After taking 25 tests output in consideration we get 5.17% difference in our expected and test output.

Expected Output	Test Output	Absolute Error	% Difference
15	16	1	6.25%
20	23	3	13.04347826 %
25	23	2	8.695652174 %
50	49	1	2.040816327%
35	37	2	5.405405405%
80	82	2	2.43902439%
60	62	2	3.225806452%
100	105	5	4.761904762%
60	63	3	4.761904762%
120	123	3	2.43902439%
30	40	10	25%
55	58	3	5.172413793%
70	68	2	2.941176471%
25	25	0	0%
22	25	3	12%
15	20	5	25%
30	33	3	9.090909091%
45	47	2	4.255319149%
40	38	2	5.263157895%
100	112	12	10.71428571%
35	35	0	0%
38	41	3	7.317073171%
30	30	0	0%
30	33	3	9.090909091%
25	30	5	16.66666667%
Total(1155)	Total(1218)	Total(77)	5.172413793%

Table 5.1: Expected output vs Test output

Two graphs are shown below to show the visual representation of the Expected output and tests output. The first one is column graph and the second graph is the Line Graph which are generated using table 5.1. Additionally, from the line graph and column graph we can see that both the lines of expected value and algorithm outputs are almost identical.

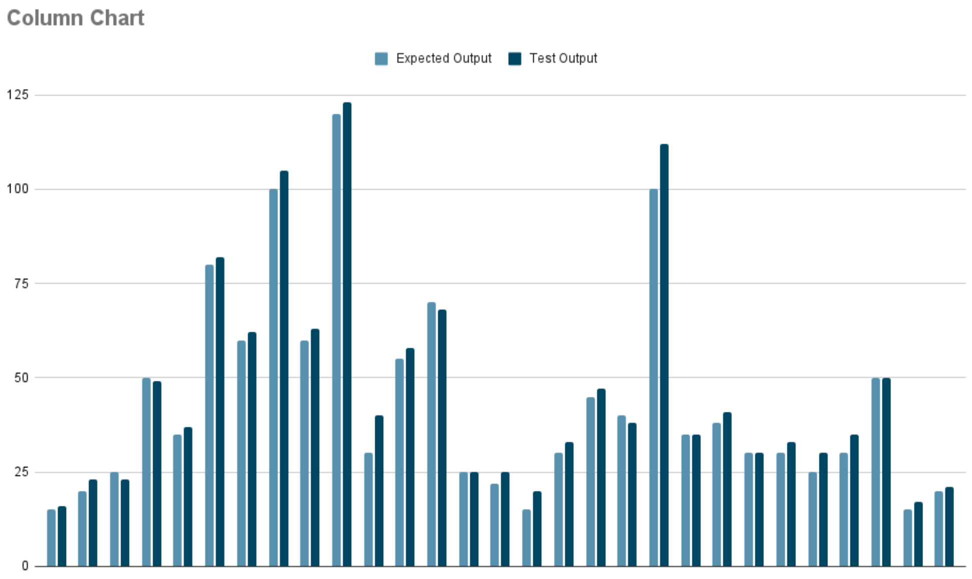


Figure 5.2: Column chart of Expected output vs Test output

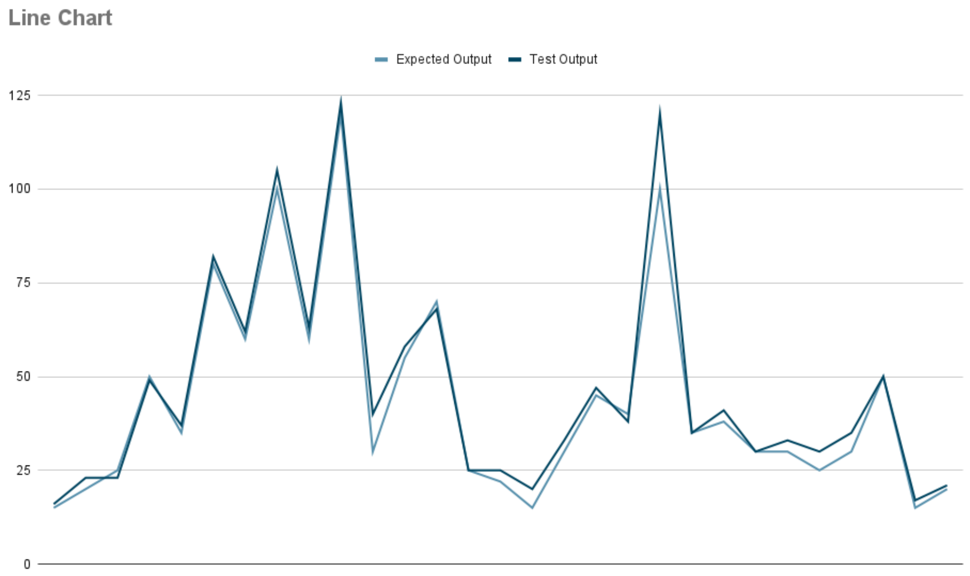


Figure 5.3: Line chart of Expected output vs Test output.

Chapter 6

Limitations and Future plans

6.1 Measuring road condition

In our country, road conditions are not the same. Road conditions keep changing. For rickshaw pullers, it is pretty tough to pull a rickshaw on the road with such bad conditions as the rickshaw is not automatic, and the rickshaw puller has to pull physically. For example, on such a condition road in rainy weather, many things can happen. However, on the other hand, a rickshaw puller can pull his rickshaw easily on a good road. Nevertheless, on the wrong road, it is hard for him to pull. So, the fare should be different based on road conditions. However, it is not possible to get that data of all road conditions at this moment, though we are considering the weather situation as we can get that data quickly. So we are facing a limitation there as the rickshaw pullers can demand extra fare based on road conditions that our algorithm does not measure

6.2 Roadblocks for rickshaw

There are many roads available where a rickshaw can go at a particular time, but at other times it is blocked for the rickshaw to take that road. There are also some roads available where at day two-way transportation is possible, but at night it becomes only a one-way road. The problem here is that our algorithm will choose the path to go to the destination and does not know about the roadblocks. When starting a ride, it will show an assumed fare for these uncertain roadblocks, but changing the road for roadblocks might affect the fare so much more than accepted because of traffic jams, extra distance, etc. At this moment, we do not have the data of this roadblock schedule. So, there is a limitation there which we are figuring out to solve.

6.3 Future Works

In the upcoming days, we are planning to make our app available in IOS. Now our app is only available on the google play store. So, only android users can use our app. After making our app available for IOS, iPhone users will be able to use our app.

6.3.1 Algorithm

Update the algorithms with more facilities. Then, we will be able to overcome our limitations of API calls when industrial implementation. We can then generate more accurate results because of lower delay and unlimited searching. In addition, however, this algorithm can be tweaked and reverse engineered to predict any vehicle's approximate fare.

6.3.2 Front-end

We are remaking UI. We will try to make our app more user-friendly—more user-friendly for both sides of the app users, including the driver and the passenger. Improved UI will improve user experience and be easier to understand.

6.3.3 Back-end

A)Database:

As we have just developed our app, its user base is minimal. That is why we are using a third-party database to store our data and all other information. Nevertheless, we have planned to shift to our private database to get a significant user base for our app. In addition, using a private database will help in data security, user privacy, controlling data backup, and recovery at any time needed.

B)Transaction:

In our app, now riders will have to pay cash physically. However, we are thinking of including a digital payment gateway system. In that way, there is no need to have physical currency to hire a rickshaw. Passengers or users will be able to pay their rickshaw fare with digital payment gateways like Bkash, Nagar, etc. Moreover, on the other hand, drivers will get the fare automatically included in their wallets, which will make the payment system easier and faster for both sides of the user.

Chapter 7

Conclusion

This thesis is aimed to introduce a new application for rickshaws. The literature on this topic reinforces the importance of the digital system of rickshaw hiring. As rickshaws are one of the regular transports, it will be more convenient for people to use an application to hire a rickshaw. We wanted to make a good algorithm to calculate the fare for the application to be appropriate for users. After implementing our algorithm, the results matched our expectations. This application will bring a new change to society. The findings above suggest that other transportation methods' impacts can be minimized by an effective rickshaw pulling digitalization architecture. A robust rickshaw pulling framework is also efficient and on-demand management; when requests are made, it can coordinate the rides. As a result, rather than receiving traditional help, it is critical to offer. In extreme weather conditions like a rainy day, passengers can discover rickshaws by using the app. Therefore, it can benefit the rickshaw driver who can solve their economic situation quickly. Similarly, the calculations for robotized coordinating capacity are tedious and boring, but they can be applied to comparative scenarios. As a result, it should prepare the model to use an intelligent transportation framework for the rickshaw service, similar to other ride-sharing services, to allow for short ride coordination and the shortest route. Based on these conclusions, practitioners should consider this project to be implemented for practical uses. Right now, our application is in the testing period and will be using in just a specific area. In the future, we will try to develop and make it usable for the people living outside Dhaka. We hope that our app will contribute significantly to our society.

Bibliography

[1] N. D. Chan and S. A. Shaheen, “Ridesharing in north america: Past, present, and future,” *Transport reviews*, vol. 32, no. 1, pp. 93–112, 2012.

[2] M. Feeney, “Is ridesharing safe?” *Cato Institute Policy Analysis*, no. 767, 2015.

[3] A. Prins, *Rickshaw restrictions: Privilege for some, disaster for the puller*, Nov. 2017. [Online]. Available: https://www.thedailystar.net/star-weekend/rickshaw-restrictions-1488862.

[4] B. Chaudhry, S. El-Amine, E. Shakshuki, *et al.*, “Passenger safety in ride-sharing services,” *Procedia computer science*, vol. 130, pp. 1044–1050, 2018.

[5] S. Akter, T. T. Jui, T. Athaya, A. Zaman, and S. Rafi, “A proposed system for fare measurement for rickshaws of bangladesh,” in *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*, IEEE, 2019, pp. 414–419.

[6] M. R. Karim and K. A. Salam, *Organising the Informal Economy Workers: A Study of Rickshaw Pullers in Dhaka City.* Bangladesh Institute of Labour Studies, 2019.

[7] M. A.-M. Molla, *Ban on rickshaw: How logical is it?* Jul. 2019. [Online]. Available: https://www.thedailystar.net/opinion/politics/news/ban-rickshaw-how-logical-it-1767535.

[8] S. Bittihn and A. Schadschneider, “Braess’ paradox in the age of traffic information,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 3, p. 033 401, 2021.

[9] *Rickshaw.* [Online]. Available: http://en.banglapedia.org/index.php?title=Rickshaw.