# CSCE 608

# Project 1

Jicheng Lu

525004048

# Design and Implementation of a National Ride-sharing System between Cities in United States

## 1. Introduction

Due to the conflict between global warming and growing demand of transportation, people tend to choose other ways to travel, such as bicycling and public transportation. Among these options, the ride-sharing is the most promising way that deserves further developing.

One of the benefits about ride-sharing is that it literally reduces the number of cars on the streets. When there is less congestion, there would be typically less driving stress. In other words, less time spent behind the wheel, and less risk of car accidents occur. In economic aspect, the longer we avoid accidents and car insurance claims, the lower our premium would go.

Even if we successfully avoid an accident, the stop-and-go traffic would cause a major headache. Anyone who drives with such a traffic would be exhausted and frustrated when he finally gets off the car. Thus, ride-sharing offers us a great opportunity to increase the occupancy of vehicles and efficiency of our transportation system. In reality, many states reward carpoolers with their own private comfort lane. The U.S. Department of Transportation (DOT) claims that in most cases, these high-occupancy vehicle (HOV) lanes are both more efficient at handling congestion and safer than their unrestricted counterparts.

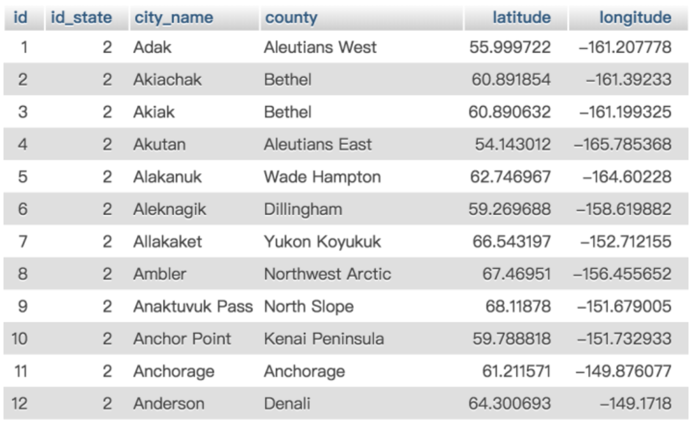
Considering the environmental issue, ride-sharing can contribute to reducing our carbon footprint. When there are fewer cars running on the street, it means that there would be fewer exhaust pipes putting out harmful emissions. Moreover, the DOT reports that HOV lanes have positively affected surrounding air quality.

In addition to the economic and environmental issues mentioned, ride-sharing also offers a healthy social alternative to a solo commute. This is absolutely true when we need to drive for a very long journey. Instead of traveling alone with radio channels, it is better to ride with other interesting colleagues and make new friends.

Therefore, based on the benefits we have discussed, it is important to establish a ride-sharing system in our life. In this project, we developed a national ride-sharing system between various cities in United States. The system is called “PickMeUp”, where anyone traveling between two cities can post his or her ride information. People can update and withdraw their posts anytime in case of emergency. As passengers, we can search and select any available ride between two locations. Moreover, passengers can cancel their booked rides whenever they change their mind. Note that this ride-sharing system does not involve with payment module. It is just an information gathering system, where any user can pick or post their rides based on those information in the database. Furthermore, some of the data in the current system are not real, especially the user information, such as user name.

## 2. Data Collection

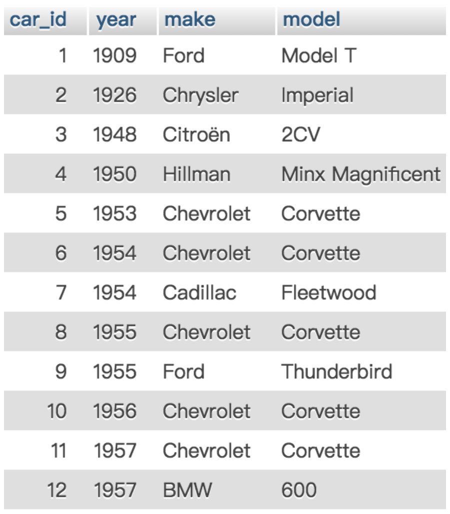
We developed the database for this ride-sharing system using MySQL. The name of the database is “pickmeup”, consisting of five tables: users, Ride, us cities, VehicleModelYear and book_user. We collected the data of US cities from the following website: https://github.com/kelvins/US-Cities-Database/blob/master/us_cities.sql. It contains more than 1,000 tuples with city name, county, latitude and longitude. Part of the data in this table is presented in Figure 2.1.



| id | id_state | city_name | county | latitude | longitude |
|---|---|---|---|---|---|
| 1 | 2 | Adak | Aleutians West | 55.999722 | -161.207778 |
| 2 | 2 | Akiachak | Bethel | 60.891854 | -161.39233 |
| 3 | 2 | Akiak | Bethel | 60.890632 | -161.199325 |
| 4 | 2 | Akutan | Aleutians East | 54.143012 | -165.785368 |
| 5 | 2 | Alakanuk | Wade Hampton | 62.746967 | -164.60228 |
| 6 | 2 | Aleknagik | Dillingham | 59.269688 | -158.619882 |
| 7 | 2 | Allakaket | Yukon Koyukuk | 66.543197 | -152.712155 |
| 8 | 2 | Ambler | Northwest Arctic | 67.46951 | -156.455652 |
| 9 | 2 | Anaktuvuk Pass | North Slope | 68.11878 | -151.679005 |
| 10 | 2 | Anchor Point | Kenai Peninsula | 59.788818 | -151.732933 |
| 11 | 2 | Anchorage | Anchorage | 61.211571 | -149.876077 |
| 12 | 2 | Anderson | Denali | 64.300693 | -149.1718 |

Figure 2.1 Part of the data in table “us cities”.

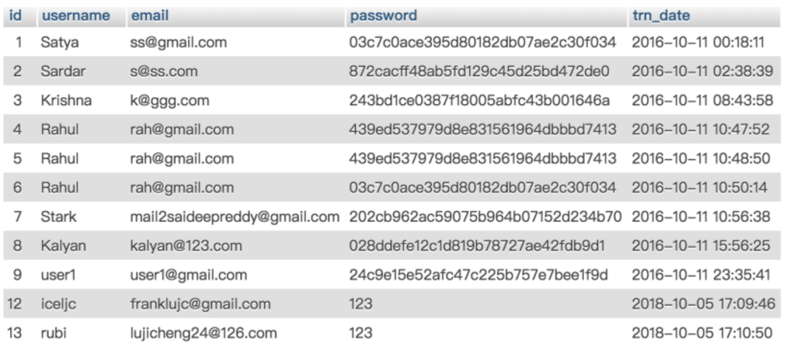
We also collected data of vehicle from the following website: https://github.com/n8barr/automotive-model-year-data/blob/master/data.sql. It contains more than 1,000 tuples with the produced year, make and model. Part of the data in this table is presented in Figure 2.2:



| car_id | year | make | model |
|---|---|---|---|
| 1 | 1909 | Ford | Model T |
| 2 | 1926 | Chrysler | Imperial |
| 3 | 1948 | Citroën | 2CV |
| 4 | 1950 | Hillman | Minx Magnificent |
| 5 | 1953 | Chevrolet | Corvette |
| 6 | 1954 | Chevrolet | Corvette |
| 7 | 1954 | Cadillac | Fleetwood |
| 8 | 1955 | Chevrolet | Corvette |
| 9 | 1955 | Ford | Thunderbird |
| 10 | 1956 | Chevrolet | Corvette |
| 11 | 1957 | Chevrolet | Corvette |
| 12 | 1957 | BMW | 600 |

Figure 2.2 Part of the data in table “VehicleModelYear”.

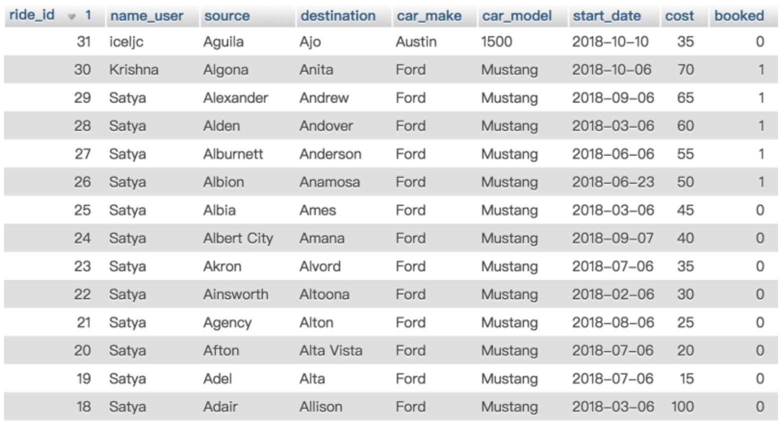
The rest of the data in this database is created on our own. The “user” table contains the user information, such as user name, email, password and created date. Part of the data in this table is presented in Figure 2.3.



| id | username | email | password | trn_date |
|---|---|---|---|---|
| 1 | Satya | ss@gmail.com | 03c7c0ace395d80182db07ae2c30f034 | 2016–10–11 00:18:11 |
| 2 | Sardar | s@ss.com | 872cacff48ab5fd129c45d25bd472de0 | 2016–10–11 02:38:39 |
| 3 | Krishna | k@ggg.com | 243bd1ce0387f18005abfc43b001646a | 2016–10–11 08:43:58 |
| 4 | Rahul | rah@gmail.com | 439ed537979d8e831561964dbbbd7413 | 2016–10–11 10:47:52 |
| 5 | Rahul | rah@gmail.com | 439ed537979d8e831561964dbbbd7413 | 2016–10–11 10:48:50 |
| 6 | Rahul | rah@gmail.com | 03c7c0ace395d80182db07ae2c30f034 | 2016–10–11 10:50:14 |
| 7 | Stark | mail2saideepreddy@gmail.com | 202cb962ac59075b964b07152d234b70 | 2016–10–11 10:56:38 |
| 8 | Kalyan | kalyan@123.com | 028ddefe12c1d819b78727ae42fdb9d1 | 2016–10–11 15:56:25 |
| 9 | user1 | user1@gmail.com | 24c9e15e52afc47c225b757e7bee1f9d | 2016–10–11 23:35:41 |
| 12 | iceljc | franklujc@gmail.com | 123 | 2018–10–05 17:09:46 |
| 13 | rubi | lujicheng24@126.com | 123 | 2018–10–05 17:10:50 |

Figure 2.3 Part of the data in table "user".

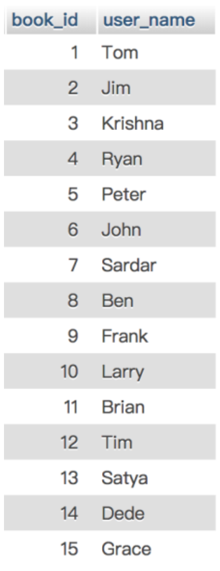
The "Ride" table contains all the ride information, such as user name who posted this ride, source, destination, car make, car model, start date, cost and status. Part of the data in this table is presented in Figure 2.4.



| ride_id 1 | name_user | source | destination | car_make | car_model | start_date | cost | booked |
|---|---|---|---|---|---|---|---|---|
| 31 | iceljc | Aguila | Ajo | Austin | 1500 | 2018–10–10 | 35 | 0 |
| 30 | Krishna | Algona | Anita | Ford | Mustang | 2018–10–06 | 70 | 1 |
| 29 | Satya | Alexander | Andrew | Ford | Mustang | 2018–09–06 | 65 | 1 |
| 28 | Satya | Alden | Andover | Ford | Mustang | 2018–03–06 | 60 | 1 |
| 27 | Satya | Alburnett | Anderson | Ford | Mustang | 2018–06–06 | 55 | 1 |
| 26 | Satya | Albion | Anamosa | Ford | Mustang | 2018–06–23 | 50 | 1 |
| 25 | Satya | Albia | Ames | Ford | Mustang | 2018–03–06 | 45 | 0 |
| 24 | Satya | Albert City | Amana | Ford | Mustang | 2018–09–07 | 40 | 0 |
| 23 | Satya | Akron | Alvord | Ford | Mustang | 2018–07–06 | 35 | 0 |
| 22 | Satya | Ainsworth | Altoona | Ford | Mustang | 2018–02–06 | 30 | 0 |
| 21 | Satya | Agency | Alton | Ford | Mustang | 2018–08–06 | 25 | 0 |
| 20 | Satya | Afton | Alta Vista | Ford | Mustang | 2018–07–06 | 20 | 0 |
| 19 | Satya | Adel | Alta | Ford | Mustang | 2018–07–06 | 15 | 0 |
| 18 | Satya | Adair | Allison | Ford | Mustang | 2018–03–06 | 100 | 0 |

Figure 2.4 Part of the data in table "Ride".

The "book_user" table contains the information about the user name who pick the ride and its corresponding ride ID. Part of the data in this table is presented in Figure 2.5.



| book_id | user_name |
|---|---|
| 1 | Tom |
| 2 | Jim |
| 3 | Krishna |
| 4 | Ryan |
| 5 | Peter |
| 6 | John |
| 7 | Sardar |
| 8 | Ben |
| 9 | Frank |
| 10 | Larry |
| 11 | Brian |
| 12 | Tim |
| 13 | Satya |
| 14 | Dede |
| 15 | Grace |

Figure 2.5 Part of the data in table "book_user".

## 3. Entity-Relationship Diagram and Table Normalization

In this section, we first describe the Entity-Relationship Diagram and describe how it is implemented in our ride-sharing system “PickMeUp”. Then we use table normalization to improve the structure of database.

The ER diagram is presented in Figure 3.1. There are four entity tables (i.e., ‘user’, ‘Ride’, ‘us cities’ and ‘VehicleModelYear’) and one relation table (i.e., ‘booking’).

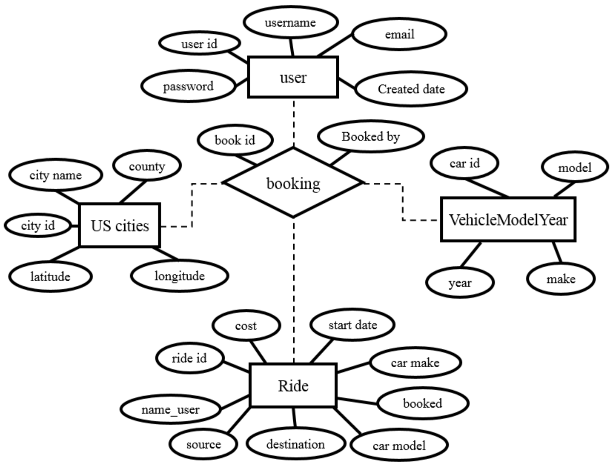
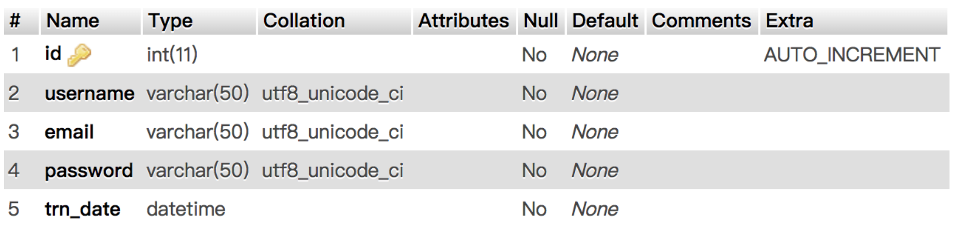


Figure 3.1 Entity-Relationship Diagram.

We notice that all the entity table has a relation ‘booking’. This is because once a passenger books a ride, all the information about this ride would be included, such as the user who posted this ride, its source and destination city, car make and model, cost, and the date of travel. Moreover, when a passenger wants to cancel his booking, it only deletes its booking information. All the other information, such as user, city, car and ride, would still maintain. Additional functions of this ride-sharing system would be illustrated in the next section.

Then we use table normalization to improve the structure of the database. First, we give the structure of each table to check if it is already in BCNF (Boyce-Codd Normal Form).

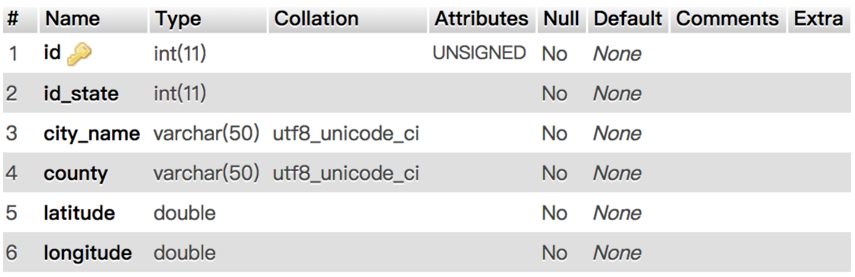
Figure 3.2 gives the structure of table “user”. Note that the nontrivial function dependency (FD) is ‘user id’ →’username’, ’email’, ‘password’, ‘trn_date’. Thus this relation is already in BCNF.



| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | **id** | int(11) | | | No | *None* | | AUTO_INCREMENT |
| 2 | **username** | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 3 | **email** | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 4 | **password** | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 5 | **trn_date** | datetime | | | No | *None* | | |

Figure 3.2 The structure of table “user”.

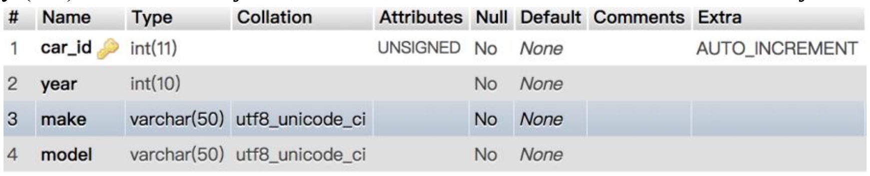
Figure 3.3 gives the structure of table “us cities”. Note that the nontrivial function dependency (FD) is ‘city id’ →’id_state’, ’city_name’, ’county’, ‘latitude’, ‘longitude’. Thus this relation is already in BCNF.



| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | UNSIGNED | No | *None* | | |
| 2 | id_state | int(11) | | | No | *None* | | |
| 3 | city_name | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 4 | county | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 5 | latitude | double | | | No | *None* | | |
| 6 | longitude | double | | | No | *None* | | |

Figure 3.3 The structure of table “us cities”.

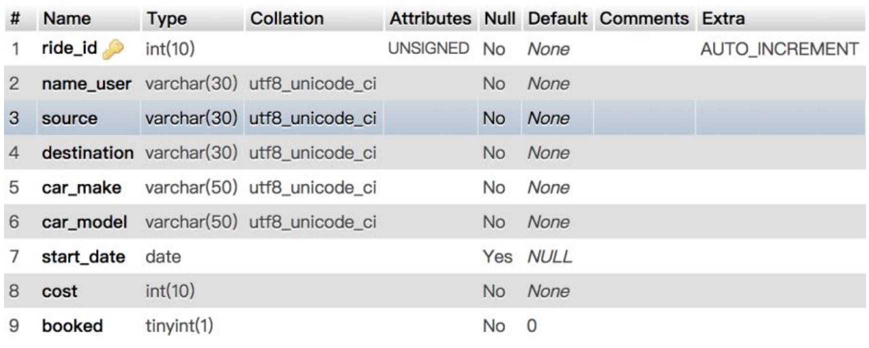
Figure 3.4 gives the structure of table “VehicleModelYear”. Note that the nontrivial function dependency (FD) is ‘car id’ →’year’, ‘make’, ‘model’. Thus this relation is already in BCNF.



| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | car_id | int(11) | | UNSIGNED | No | *None* | | AUTO_INCREMENT |
| 2 | year | int(10) | | | No | *None* | | |
| 3 | make | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 4 | model | varchar(50) | utf8_unicode_ci | | No | *None* | | |

Figure 3.4 The structure of table “VehicleModelYear”.

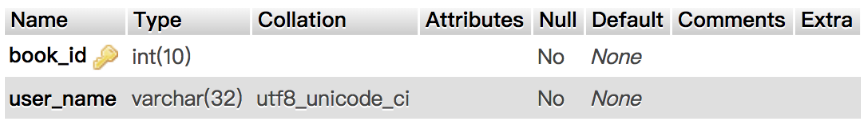
Figure 3.5 gives the structure of table “Ride”. Note that the nontrivial function dependency (FD) is ‘ride id’ →’name_user’, ‘source’, ‘destination’, ‘car_make’, ‘car_model’, ‘start_date’, ‘cost’, ‘booked’. Thus this relation is already in BCNF.



| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | ride_id | int(10) | | UNSIGNED | No | *None* | | AUTO_INCREMENT |
| 2 | name_user | varchar(30) | utf8_unicode_ci | | No | *None* | | |
| 3 | source | varchar(30) | utf8_unicode_ci | | No | *None* | | |
| 4 | destination | varchar(30) | utf8_unicode_ci | | No | *None* | | |
| 5 | car_make | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 6 | car_model | varchar(50) | utf8_unicode_ci | | No | *None* | | |
| 7 | start_date | date | | | Yes | *NULL* | | |
| 8 | cost | int(10) | | | No | *None* | | |
| 9 | booked | tinyint(1) | | | No | 0 | | |

Figure 3.5 The structure of table “Ride”.

Figure 3.6 gives the structure of table “book_user”. Note that the nontrivial function dependency (FD) is ‘book id’ →’user_name’. Thus this relation is already in BCNF.



| Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|
| book_id | int(10) | | | No | *None* | | |
| user_name | varchar(32) | utf8_unicode_ci | | No | *None* | | |

Figure 3.6 The structure of table “book_user”.

Therefore, all of the table structures remain the same because they are already in Boyce-Codd Normal Form.

## 4. User Interface

The “PickMeUp” ride-sharing system is composed of seven components: Login/Logout, Registration, Booked a Ride, Cancel My Booked Ride, Post a Ride, Cancel My Posted Ride and Update My posted Ride.

Before showing the user interface, we need to create a database called “pickmeup”. Then we import all the data we have collected or generated. To do this, we simply import the SQL files in the ‘Dataset’ folder. Once these two steps are done, we can go to our website (http://localhost/pickup/login.php). Figure 4.1 presents the login screen.

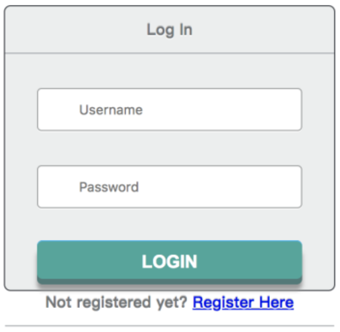


Figure 4.1 Login screen.

Since we do not have an account yet, we can go to the “Registration” page and create our user account. Note that the user name has to be unique, otherwise the registration cannot be complete. Figure 4.2 presents the registration screen.

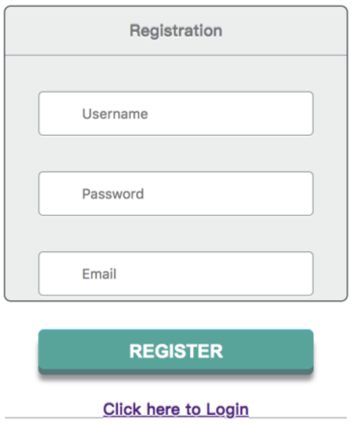


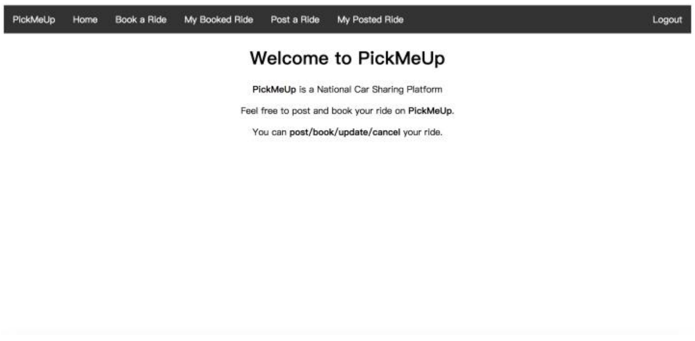
Figure 4.2 Registration screen.

For convenience, the two accounts have been created:

1) Username: iceljc, password: 123.
2) Username: rubi, password: 123.

We will use these two accounts for further illustration.

Once we log in, we will be directed to the home page, where there is a greeting slogan and the following instructions guide us to each module of this system. Figure 4.3 presents the home page.



Welcome to PickMeUp

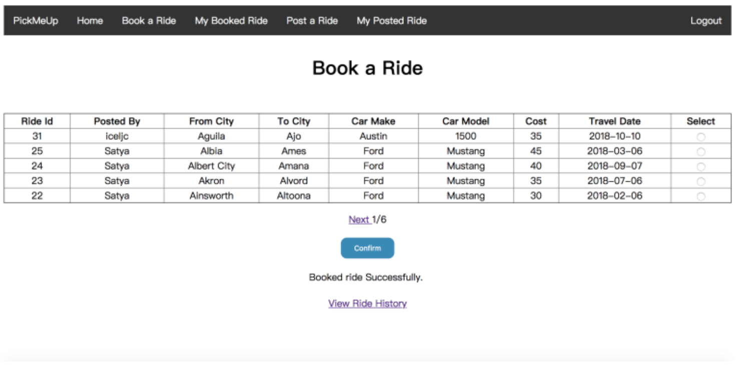
**PickMeUp** is a National Car Sharing Platform

Feel free to post and book your ride on **PickMeUp**.

You can **post/book/update/cancel** your ride.

Figure 4.3 Home page.

We first go to the "Book a Ride" page, as is shown in Figure 4.4. We can pick a ride by selecting the radio button on the right. Once we have successfully booked a ride, there will be a hint below to indicate that our booking operation is completed. Then we can click "View Ride History" and move to the next page "My Booked Ride". Note that once we book a ride, we cannot see the information about this ride in the "Book a Ride" page. This is because in this system, one ride can only pick up one passenger. Moreover, we are not allowed to select the ride which is posted by ourselves.



Book a Ride

| Ride Id | Posted By | From City | To City | Car Make | Car Model | Cost | Travel Date | Select |
|---|---|---|---|---|---|---|---|---|
| 31 | iceljc | Aguila | Ajo | Austin | 1500 | 35 | 2018-10-10 | ○ |
| 25 | Satya | Albia | Ames | Ford | Mustang | 45 | 2018-03-06 | ○ |
| 24 | Satya | Albert City | Amana | Ford | Mustang | 40 | 2018-09-07 | ○ |
| 23 | Satya | Akron | Alvord | Ford | Mustang | 35 | 2018-07-06 | ○ |
| 22 | Satya | Ainsworth | Altoona | Ford | Mustang | 30 | 2018-02-06 | ○ |

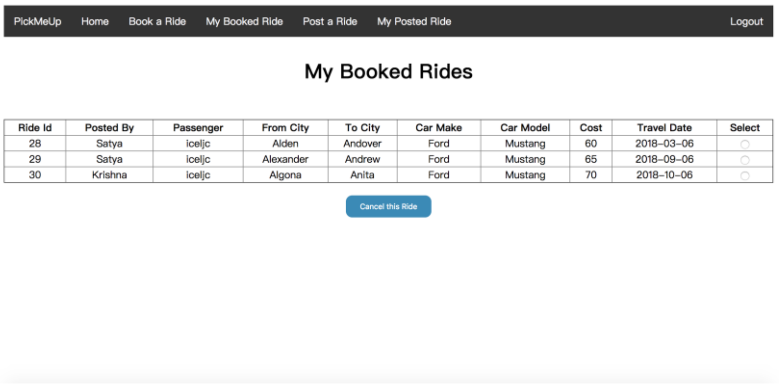
Next 1/6

Booked ride Successfully.

View Ride History

Figure 4.4 "Book a Ride" page.

After we booked a ride, we will be redirected to the "My Booked Ride" page, where we can check our booking information. If we select a radio button on the right and press the blue button below, we can cancel our booked ride and this ride can show up in the previous page again. Figure 4.5 presents the "My Booked Ride" page.

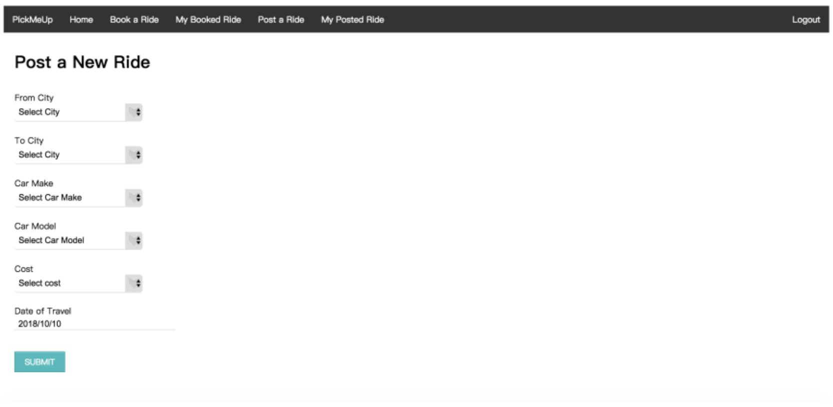


My Booked Rides

| Ride Id | Posted By | Passenger | From City | To City | Car Make | Car Model | Cost | Travel Date | Select |
|---|---|---|---|---|---|---|---|---|---|
| 28 | Satya | iceljc | Alden | Andover | Ford | Mustang | 60 | 2018-03-06 | ○ |
| 29 | Satya | iceljc | Alexander | Andrew | Ford | Mustang | 65 | 2018-09-06 | ○ |
| 30 | Krishna | iceljc | Algona | Anita | Ford | Mustang | 70 | 2018-10-06 | ○ |

Figure 4.5 "My Booked Ride" page.

We can also post a ride information in “PickMeUp”. Figure 4.6 presents the “Post a Ride” page, where we can input our ride information, such as source and destination, car make and model, travel cost, and date of travel. Once we finish the input, click the “submit” button below and the ride information will show up in the “Book a Ride” page.

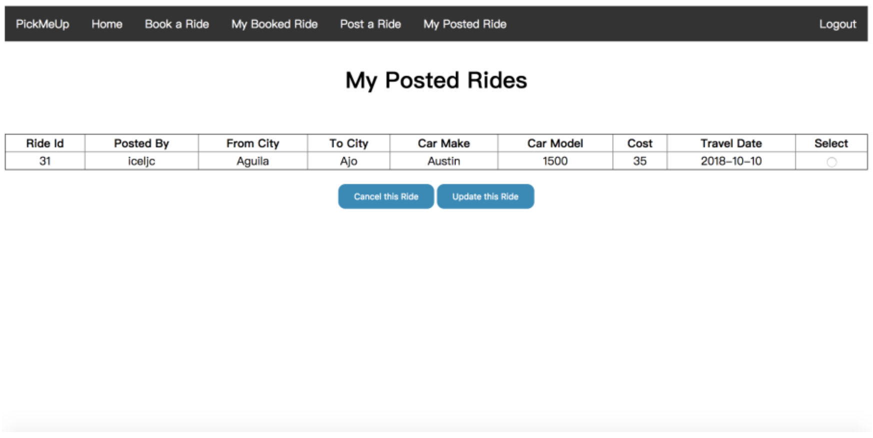


Post a New Ride

| | |
|---|---|
| From City | Select City |
| To City | Select City |
| Car Make | Select Car Make |
| Car Model | Select Car Model |
| Cost | Select cost |
| Date of Travel | 2018/10/10 |

Figure 4.6 “Post a Ride” page.

After we post a ride, we can go to the “My Posted Ride” page, where we can cancel or update our posted page by doing the similar operation. Note that if we cancel the posted ride, the booking of the passenger who selected this ride gets canceled as well. Figure 4.7 presents the “My Booked Ride” page.



My Posted Rides

| Ride Id | Posted By | From City | To City | Car Make | Car Model | Cost | Travel Date | Select |
|---|---|---|---|---|---|---|---|---|
| 31 | iceljc | Aguila | Ajo | Austin | 1500 | 35 | 2018-10-10 | ○ |

Figure 4.7 “My Posted Ride” page.

When we finish the operations in the system, we can click “Logout” and the system will redirect us to the Login page.

(The complete code and data are at the github: https://github.tamu.edu/iceljc/CSCE608)

## 5. Discussion

To be honest, I am new to the database system because I transfer from other major. Thus, I spent some time on the SQL first and got familiar with the typical operations, such as insert, delete, select, update and etc. Then I go through part of the text book, mainly about the basics of database system, such as schema, relation, ER diagram, table normalization etc. I have tried my best to go through the table normalization based on the application in this report. I am not completely sure that it is correct. If there is something wrong, please let me know so that I can enhance my knowledge.

Then it comes to the biggest difficulty in this project: how to use PHP to combine MySQL and create an interface. Thus, I first went to the library to check if there is any learning material about PHP and MySQL, fortunately the book "PHP & MYSQL: Novice to Ninja" gave me a great help and I started to get familiar with PHP. Moreover, I got some advice from my colleagues who used to do projects on database and their suggestion helped me fix the issues in my code.

Therefore, after going through this project, I finally get myself familiar with the basics of database system. Meanwhile, I have also gained knowledge about how to implement the MySQL query in PHP and presented these SQL operations in the user interface.