IDBMS PROJECT

TOLL PLAZA MANAGEMENT SYSTEM

SUBMITTED TO

Dr. Ram Prakash Sharma

SUBMITTED BY

GROUP 11

GROUP MEMBERS	ROLL NO
1. Aashita Agarwal	19UCS160
2. Pranjal Gupta	19UCS162
3. Aayush Mehta	19UCS100
4. Umang Goyal	19UCS097

Introduction:

It demonstrates Basic working of a Toll management system that has multiple toll Stations across the country. It has 2 types of system users - administrator and customers (managed by toll worker).

Software Requirements:

1. This project is a desktop application which uses java swing in frontend.
2. MySQL server (version 8.0.20) in backend.
3. It is developed on Apache NetBeans IDE 12.1 and
4. Connection to MySQL using JDBC Driver version 8.0.22.

Basic Functionalities include

1. **Administrator can login** to the System.
2. He / She can view the **total revenue generated** at a toll station of his/her choice on **any particular day**.
3. He / She can also view the **details of a Vehicle** by entering its **"Vehicle number"**.
4. Customers can view their **total expenses** for the complete trip and also view what they will be paying at a particular toll station.
5. When using first time customers will be **automatically be registered**.
6. For every new entry done by toll worker, customers have to select **toll station, journey type(single/return) and date of the trip**. Then **calculate or add next destination** toll if they wish to continue or generate bill if they have finished journey.

Key notes & assumptions:

1. administrator username="admin" and password="1234" (Can't be changed).
2. If a person selects return as journey type then if he returns at the same station on the same day with journey type single he will be cost Rs 0 but if he does not return on same day he will be cost correct amount as specified.

But if the journey type is return again he will be paying same amount as specified in database.

Relations and Attributes:

We have created a database having 4 interconnected tables described as follows:

1) Booth (boothID, location)

The Booth table consists of details of booth's id and their location .

The boothID is set as PRIMARY KEY .The location is set as NOT NULL.

```
mysql> desc booth;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| boothid  | int         | NO   | PRI | NULL    |       |
| location | varchar(30) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

2) Fare(fareid, vehicletype, journeytype, fare_amt)

The Fare table consists of details of fareid, vehicletype, journeytype and fare_amt

The fareid is set as INT PRIMARY KEY.

for vehicletype it is NOT NULL varchar(50)

for journeytype it is NOT NULL varchar(50)

for fare_amt it is INT NOT NULL.

```
mysql> desc fare;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| fareid      | int         | NO   | PRI | NULL    |       |
| vehicletype | varchar(30) | NO   | MUL | NULL    |       |
| journeytype | varchar(30) | NO   | MUL | NULL    |       |
| fare_amt    | int         | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

3) Transactions (tranID, vehicle_no, boothID, journeyType, fareID, visitdate)

The Transactions table consists of details of tranid, vehicle_no, boothid, journeytype, fareid, visitdate.

tranid is set as PRIMARY KEY auto_increment

The vehicle_no is set as FOREIGN KEY which references to the details(vehicle_no), boothid is set as FOREIGN KEY which references to the booth(boothid), journeytype is set as FOREIGN KEY which references to the fare(journeytype), fareid is set as FOREIGN KEY which references to the fare(fareid). The Data types of attributes are as shown in the figure below.

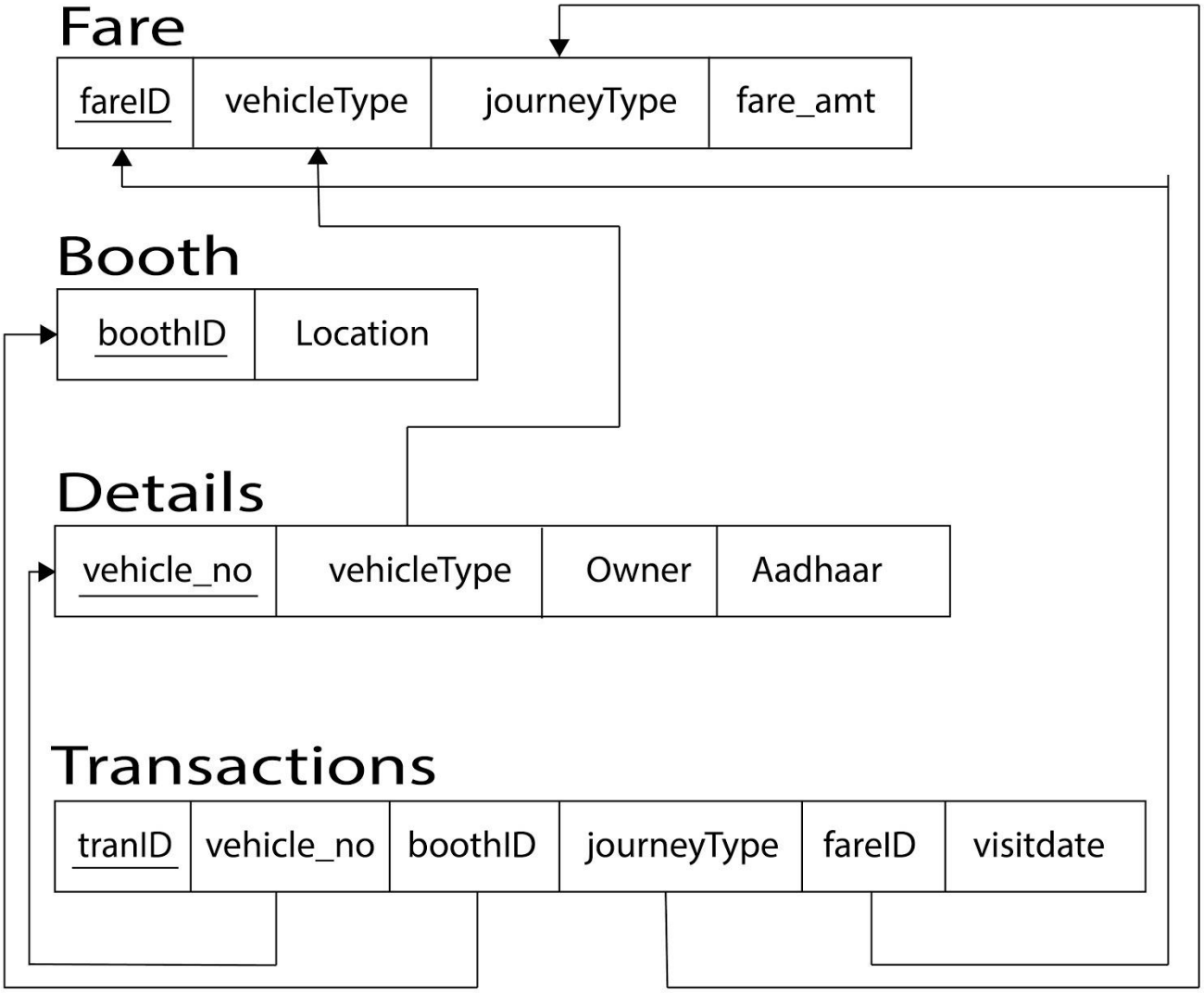
```
mysql> desc transactions;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| tranid      | int         | NO   | PRI | NULL    | auto_increment |
| vehicle_no  | varchar(30) | NO   | MUL | NULL    |                |
| boothid     | int         | NO   | MUL | NULL    |                |
| journeytype | varchar(30) | NO   | MUL | NULL    |                |
| fareid      | int         | NO   | MUL | NULL    |                |
| visitdate   | date        | NO   |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

4) Details(vehicle_no, vehicletype, owner, aadhaar)

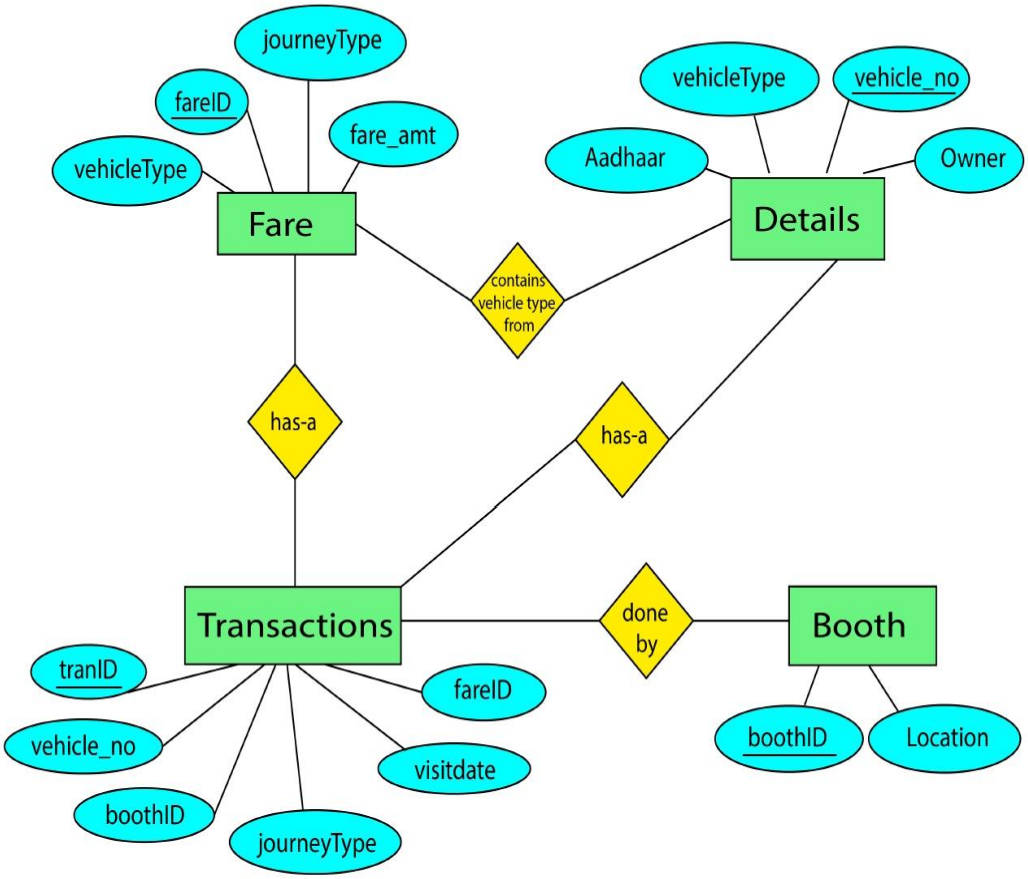
The Details table consists of details of vehicle_no, vehicletype, owner, aadhaar attributes. The vehicle_no is set as PRIMARY KEY. The vehicletype is set as FOREIGN KEY which references to the Fare(vehicletype). The Datatype of vehicle_no is varchar(15) , vehicletype is varchar(50), owner is varchar(30) and for aadhaar is varchar(15).

```
mysql> desc details;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| vehicle_no  | varchar(15) | NO   | PRI | NULL    |       |
| vehicletype | varchar(30) | NO   | MUL | NULL    |       |
| owner       | varchar(30) | YES  |     | NULL    |       |
| aadhaar     | varchar(15) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Relation diagram:



ER diagram:


journeyType
fareID
fare_amt
vehicleType
Fare
vehicleType
vehicle_no
Aadhaar
Owner
Details
contains vehicle type from
has-a
has-a
Transactions
done by
Booth
tranID
vehicle_no
boothID
journeyType
fareID
visitdate
boothID
Location

Relation's Tuples:

Fare:

```
mysql> select * from fare;
+--------+--------------+-------------+----------+
| fareid | vehicletype  | journeytype | fare_amt |
+--------+--------------+-------------+----------+
|    100 | 3 Wheeler    | Single      |       10 |
|    101 | 3 Wheeler    | Return      |       19 |
|    102 | Car/Jeep     | Single      |       30 |
|    103 | Car/Jeep     | Return      |       58 |
|    104 | LCV          | Single      |       49 |
|    105 | LCV          | Return      |       95 |
|    106 | Bus          | Single      |       78 |
|    107 | Bus          | Return      |      150 |
|    108 | Truck        | Single      |      117 |
|    109 | Truck        | Return      |      220 |
|    110 | MAV          | Single      |      234 |
|    111 | MAV          | Return      |      440 |
+--------+--------------+-------------+----------+
12 rows in set (0.00 sec)
```

Transactions:

```
mysql> select * from transactions;
+--------+-------------+---------+-------------+--------+------------+
| tranid | vehicle_no  | boothid | journeytype | fareid | visitdate  |
+--------+-------------+---------+-------------+--------+------------+
|  10000 | RJ14 GR 1206 |     100 | Single      |    100 | 2020-11-01 |
|  10001 | RJ14 GR 1206 |     105 | Single      |    100 | 2020-11-03 |
|  10002 | RJ14 RQ 7819 |     102 | Single      |    104 | 2020-11-05 |
|  10003 | DL90 TI 6629 |     100 | Return      |    111 | 2020-10-07 |
|  10004 | DL59 BZ 9221 |     103 | Return      |    107 | 2020-11-08 |
|  10005 | DL86 AE 5801 |     104 | Return      |    103 | 2020-11-10 |
|  10006 | DL22 AU 2662 |     100 | Single      |    100 | 2020-11-12 |
|  10007 | RJ41 JR 6910 |     102 | Return      |    111 | 2020-09-14 |
|  10008 | RJ05 SC 5673 |     101 | Single      |    106 | 2020-09-16 |
|  10009 | RJ56 LA 2075 |     103 | Single      |    104 | 2020-10-18 |
|  10010 | MH16 YB 6683 |     104 | Single      |    102 | 2020-11-20 |
|  10011 | MH04 ZF 4311 |     101 | Single      |    104 | 2020-11-22 |
|  10012 | MH04 ZF 4311 |     104 | Return      |    105 | 2020-11-24 |
|  10013 | MH26 BI 3702 |     102 | Single      |    108 | 2020-10-26 |
|  10014 | RJ01 KK 7367 |     105 | Single      |    102 | 2020-10-28 |
|  10015 | MH16 YB 0001 |     104 | Single      |    102 | 2020-11-01 |
|  10016 | MH04 ZF 4311 |     101 | Single      |    104 | 2020-11-03 |
|  10017 | MH04 ZF 4311 |     104 | Return      |    105 | 2020-11-05 |
|  10018 | MH26 BI 3702 |     102 | Single      |    108 | 2020-10-05 |
|  10019 | RJ01 KK 7367 |     105 | Single      |    102 | 2020-10-05 |
+--------+-------------+---------+-------------+--------+------------+
20 rows in set (0.00 sec)
```

Details:

```
mysql> select * from details;
+--------------+-------------+-----------+-----------------+
| vehicle_no   | vehicletype | owner     | aadhaar         |
+--------------+-------------+-----------+-----------------+
| DL22 AU 2662 | 3 Wheeler   | Sanskar   | 0006 XXXX 3564 |
| DL59 BZ 9221 | Bus         | Umang     | 0004 XXXX 1346 |
| DL86 AE 5801 | Car/Jeep    | Kartik    | 0005 XXXX 9874 |
| DL90 TI 6629 | MAV         | Aashita   | 0003 XXXX 4523 |
| MH04 ZF 4311 | LCV         | Dastagiri | 0011 XXXX 4542 |
| MH16 YB 0001 | Car/Jeep    | Ansh      | 0014 XXXX 2348 |
| MH16 YB 6683 | Car/Jeep    | Shivam    | 0010 XXXX 4587 |
| MH26 BI 3702 | Truck       | Anant     | 0012 XXXX 0973 |
| RJ01 KK 7367 | Car/Jeep    | Chaitanya | 0013 XXXX 9827 |
| RJ05 SC 5673 | Bus         | Rohit     | 0008 XXXX 9873 |
| RJ14 GR 1206 | 3 Wheeler   | Aayush    | 0001 XXXX 1234 |
| RJ14 RQ 7819 | LCV         | Pranjal   | 0002 XXXX 8762 |
| RJ41 JR 6910 | MAV         | Bhumur    | 0007 XXXX 4374 |
| RJ56 LA 2075 | LCV         | Satyam    | 0009 XXXX 4543 |
+--------------+-------------+-----------+-----------------+
14 rows in set (0.00 sec)
```

Booth:

```
mysql> select * from booth;
+---------+----------+
| boothid | location |
+---------+----------+
|     100 | Jaipur   |
|     101 | Delhi    |
|     102 | Nagpur   |
|     103 | Jodhpur  |
|     104 | Alwar    |
|     105 | Udaipur  |
+---------+----------+
6 rows in set (0.00 sec)
```

Working:

1. We will first create a database '*tollmanagement*'.
2. Then add the 4 tables as specified: '*fare*', '*booth*', '*details*', '*transactions*'.
3. Then we enter some sample data in the tables.
4. Now, as we run our project we select '*Login.java*' as our Main Class.
5. The execution of the project is described in the file ' '.
6. If we are an administrator then we enter the credentials and click on 'Login'.
    a. If the credentials are correct we are redirected to the *'Admin.java'* page otherwise an error message will appear.
7. On *'Admin.java'* page admin has 2 options:
    a. to view the total revenue of the day on a toll station
    b. view details of a vehicle using vehicle number.
8. As we enter the date and select Toll Station we first get the *boothid* according to the station from the *booth* table and then join *fare* and *transactions* according to *fareid* to get the total sum in the output field.
9. If we want to view details of a vehicle we enter the vehicle number which searches for details in *details* tables.
10. Admin can either go on looking at more data or exit using the exit button.
11. If we are a toll station worker we can directly click on calculate expenses on '*Login.java*' to redirect us to the *'Caculate.java'* page.
12. On *'Caculate.java'* page the toll station worker enters the details of the arriving vehicle then the transaction details (journey type, Toll Station, visit date) and then click on calculate.
13. It checks for the details in the *details* table if not found it will add the details to the *details* table and then it will search for *fareid* according to vehicle type and journey type selected from *fare* table and *boothid* according to booth selected from *booth* table.
14. If a person selects return as journey type then if he/she returns at the same station on the same day with journey type single he will be cost Rs 0 but if he doesn't return on the same day he/she pays the correct amount as specified.

15. If the journey type is single it searches the *transactions* table for the entered vehicle number return journey type and same date. If found the charges for the current transactions become 0 otherwise it is according to the fareid.

16. The method then inserts a new tuple in the *transactions* table accordingly and displays the charges for the current transactions in a label.

17. To continue the journey/trip, customers can ask the toll station worker to select add next station button. This resets journey type, Toll Station, visit date fields.

18. At the end of the trip the customer has 2 options

    a. either view the total fare receipt

    b. or directly exit.

19. According to this choice the toll station worker selects the generate bill or exit option.

20. If generate bill is selected a user defined constructor takes details from the previous page in an arraylist to display all the details of vehicles and the total expenses of the trip is displayed which the customer can view.