Parking Lot Management System

Project Description:

The purpose of this project is to track and manage occupancy of a parking garage and allow customers to find and reserve available parking places.

The parking garage currently operates without any computerized system. The management has concerns about drivers not able to use parking space in optimal way. Congestion inside the garage is often caused by drivers searching for vacant spots.

Now we're remodelling the parking lot management system in a way so that we can keep track of number of vehicles inside the parking lot, number of vacant slots inside the parking lot ,etc.

We'll put up a portal in which customers can check for vacant parking slots, reserve a slot for a particular period of time, purchase monthly passes ,etc.

Introduction

Types of customers

- A regular customer who has purchased a biweekly, monthly, or yearly pass.
- A prepaid customer who has booked a slot previously using mobile.
- A customer who neither has a pass nor booked a slot remotely. Parking slot for this type of customer is assigned based on the availability of vacant parking slots.

Parking lot reserves an entire floor for regular customer so that they make sure regular customers always have a slot to park their vehicle at any given time.

Regular customers can also reserve a particular slot for themselves to park their vehicles in the same designated slots everyday but this costs little bit extra money.

Customers who make a remote reservation have to park their vehicles in their designated slots, they’ll be penalised accordingly if they don’t vacant their slots after the stipulated time. If the customer isn’t able to make it in time and requested for a refund before one hour of his booking time gets 70% of money as refund else loses his money and the particular slot is kept vacant.

In the parking lot few slots have a fixed minimum time window.

Eg. Though the customer doesn’t want to park his vehicle more than two hours, he has to pay for 3 hours since he has booked a slot which has fixed time window of 3 hours.

Walk-in customers reservation entirely depends on availability of slots. When there’s a high demand for slots surge pricing comes into the picture.

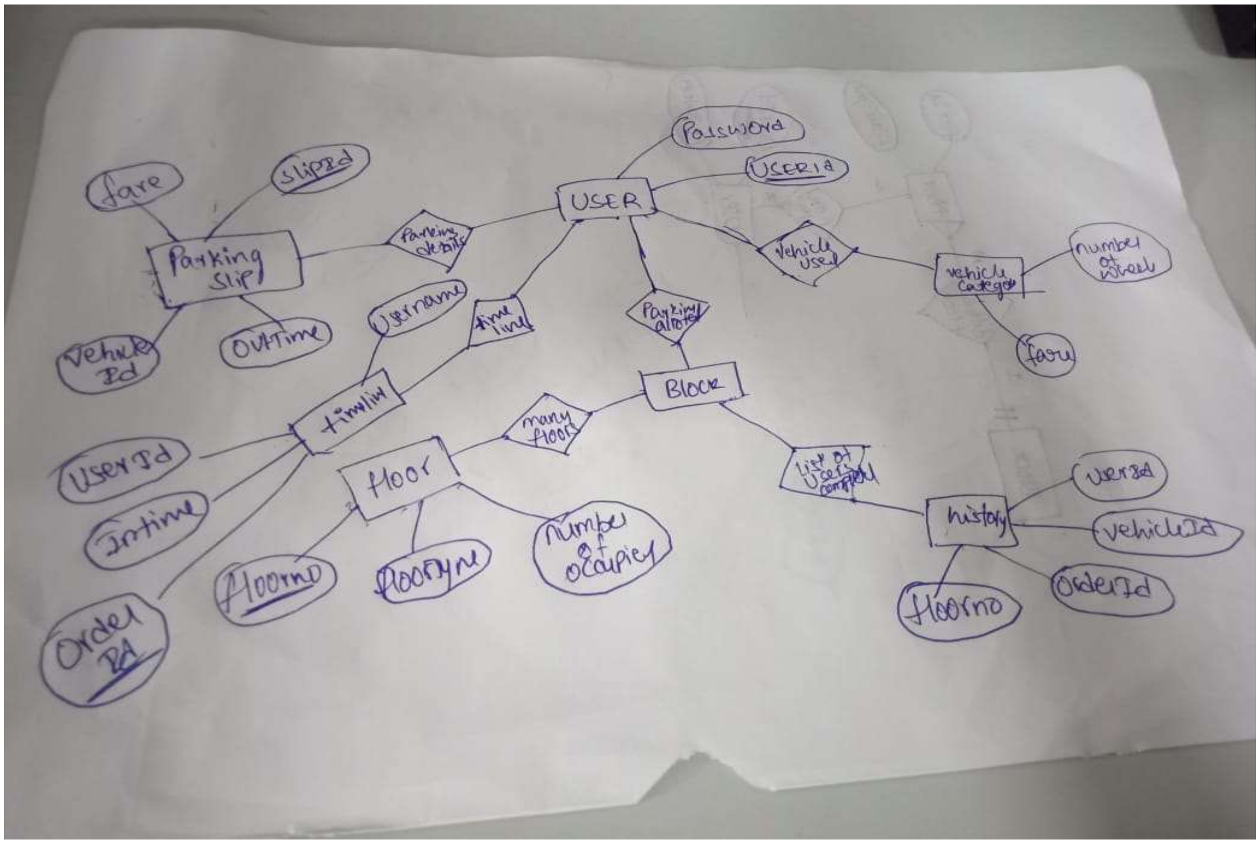
Incase if a customer wants to extend his/her time slot, customer can do it only if his/her slot vacant after his stipulated time or if there’re any vacant slots in the floor.

E-R analysis

Entities

- **user**
- **Block**
- **Floor**
- **timeline**
- **history**
- **vehiclecategory**
- **parking slip**

E-R Diagram



Password
USERid
USER
fare
slipId
Parking Slip
Parking details
Vehicle Id
OutTime
Username
time line
Parking allotted
Vehicle used
vehicle category
number of wheels
fare
timeline
Block
many floor
UserId
Intime
Order Id
floor
floorno
floortype
number of occupied
list of users completed
history
UserId
vehicleId
OrderId
floorno

Tables

users(userId,password)

Block(Floorno,numberofoccupied,floorType
)

timeline(userId,vehicleId,username,vehicleT
ype,orderId,inTime,floorno)

history(userId,vehicleId,vehicleType,orderId
,inTime,outTime,floorno)

vehiclecategory(numberofWheels,fare)

parkingslip(slipId,userId,vehicleId,vehicleTy
pe,inTime,outTime,fare)

FUNCTIONAL DEPENDENCIES

userId→username (non trivial)
vehicleId→vehicleType
floorno→floorType
floorType→vehicleType
orderId→userId
slipId→userId
userId→vehicleId

INFERENCE RULES:

floorno→vehicleType(IR3)
floorno→floortypevehicletype(IR4)
orderId→username(IR3)

slipId$\rightarrow$username(IR3)
orderIdslipId$\rightarrow$userId(union)
userId$\rightarrow$vehicletype(IR3)

TRIGGERS CREATED:

There are 3 triggers which are:

1) As soon as a row entered in timeline table the same row values will be inserted into history and parking slip tables
2) If there is a vehicle added in the parking lot then there will be an auto incrementation in the column consisting of occupancy of vehicles
3) If the service of the particular user is completed then the row

consisting the details of that customer will be deleted in the timeline table and the outTime of the same orderId in history table will be updated and fare,outTime in the parking slip table

SCHEMA:

```
create database if not exists
backend;
use backend;

create table timeLine(orderId
char(100) primary key,userId
char(100),vehicleId
char(20),userName
char(15),vehicleType int,inTime
timestamp default
current_timestamp,floorNo int);
```

```
create table block1(floorNo int
primary key,numberOfOccupied
bigint default 0,floorType int);


create table history(orderId
char(100) primary key,userId
char(100),vehicleId
char(20),userName
char(100),vehicleType int,inTime
timestamp default
current_timestamp,outTime
timestamp default
current_timestamp,floorNo int);


#drop table vehicleCategory;
```

```
create table
vehicleCategory(numberOfWheels
bigint,fare double);

SET SQL_SAFE_UPDATES = 0;

create table slip(orderId
char(100),userId char(100),
vehicleId char(100),userName
char(100),vehicleType int,
inTime timestamp default
current_timestamp,outTime
timestamp default
current_timestamp,fare bigint
default 0);


insert into vehicleCategory value
(1,.25);
```

```
select * from history;

#drop table history;

SELECT
dateadd(day,datediff(day,0,GETDA
TE()),0);

SELECT * FROM timeLine order by
inTime DESC;
#WHERE
DATEDIFF(date(timeLine.inTime)
,current_date) = -1;

select * from history order by
outTime DESC;

show tables;

drop table users;
```

```
select * from slip;

insert into
timeLine(orderId,userId,vehicleId,
userName,vehicleType,floorNo)
values("f23","helloooo","sdfg","sd
",1,1);

select * from timeLine;

select count(distinct orderId) as
count from history where
DATEDIFF(date(history.inTime),dat
e(current_date))>=-1;

#delete from  where 1;

select * from history;

select * from block1;
```

```
insert into block1 values
(1,0,1),(2,0,1),(3,0,2),(4,0,2);


#drop trigger incrementFloor;

show triggers;

delimiter #;

create trigger addVehicle after
insert on timeLine for each row
begin
insert into
history(orderId,userId,vehicleId,us
erName,vehicleType,floorNo)
values(new.orderId,new.userId,ne
w.vehicleId,new.userName,new.v
ehicleType,new.floorNo);
end;
delimiter;
```

```
use backend;


delimiter #;

create trigger incrementFloor after
insert on timeline for each row
begin
update block1 set
block1.numberOfOccupied=block1
.numberOfOccupied+1
where
block1.floorNo=new.floorNo;
end;

delimiter;
```

```
delimiter #;

create trigger addToSlip after
insert on timeline for each row
begin
insert into
slip(orderId,userId,vehicleId,userN
ame,vehicleType)
values(new.orderId,new.userId,ne
w.vehicleId,new.userName,new.v
ehicleType);
end;

delimiter;




delimiter #;

create trigger bill after delete on
timeline for each row
```

```
begin
update slip set
outTime=current_timestamp(),fare
=(select
timestampdiff(minute,slip.inTime,
slip.outTime)) where
slip.orderId=old.orderId;
end;

delimiter;
```