Parking Lot
(Low Level Design)

Flow:

Vehicle
- → Vehicle Id
- → Vehicle Type

Entrance Gate
- → Find Parking Spot
- → Book a Spot
- → Generate a ticket

Parking Spot
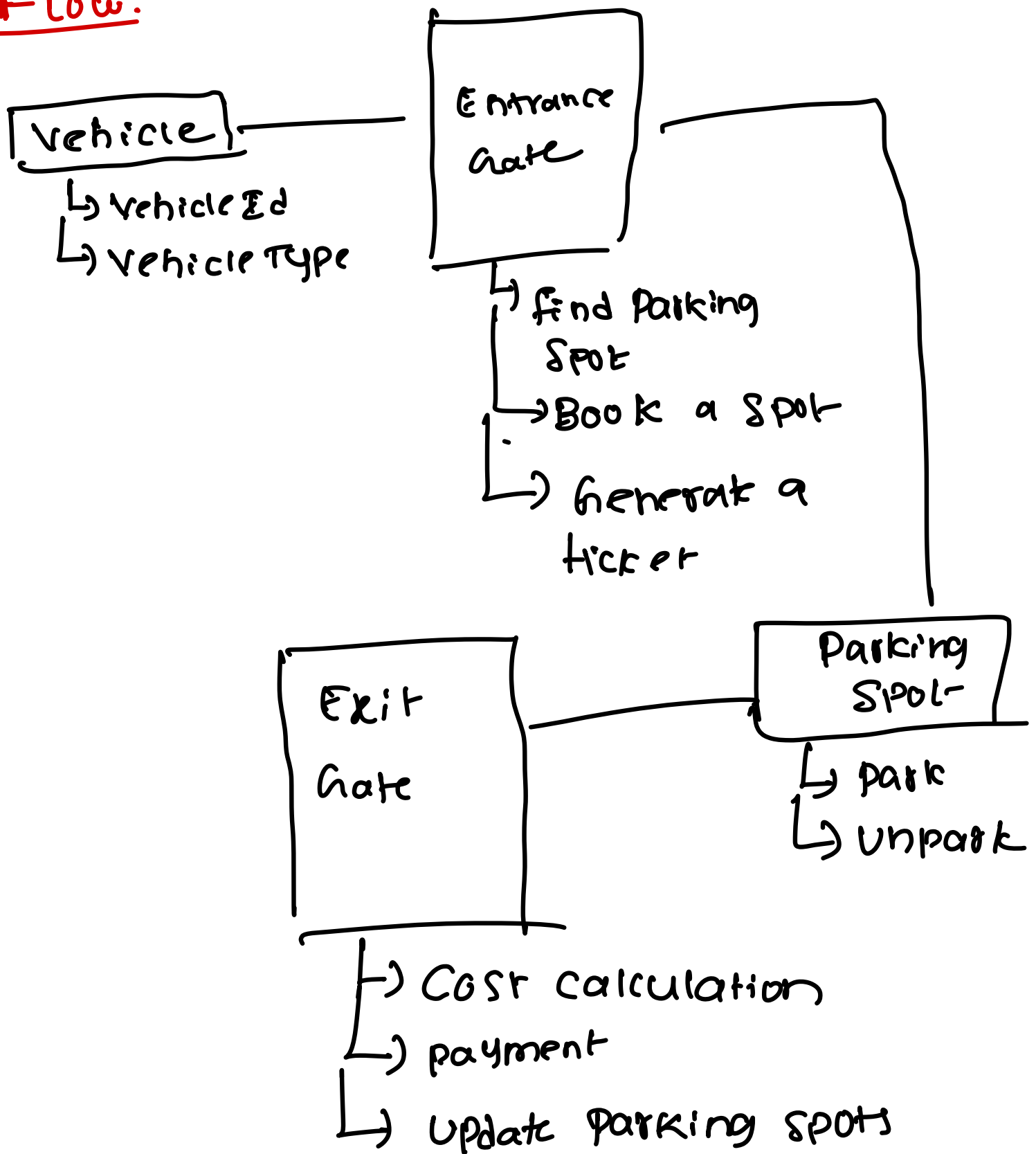- → park
- → unpark

Exit Gate
- → Cost calculation
- → payment
- → Update parking spots

Requirements:

1) Types of vehicles (Two wheeler, 4 wheeler)

2) How many Entrance Gates? " (1)

3) Nearest parking slot should be provided

4) payment (Hourly / minute based)

Low Level Design: (Bottom-Up Approach)

Parking Spot Manager

List < ParkingSpot >

---

addParkingSpot ( )
remove ParkingSpot ( )
find ParkingSpot ( )

Parking Spot

+ id : int
+ VehicleType :
    VehicleEnum
+ isAvailable :
    boolean
+ Price : int
+ Vehicle : Vehicle

+ ParkVehicle ( )
{
    isAvailable
        = false
    Vehicle = Vehicle
}

+ unpark ( )
{
    isAvailable
        = true
    Vehicle = null
( ) }

Two Wheeler Parking Spot Manager

---

const {
    super (list of
        two wheelers)

Four Wheeler Parking Spot Manager

Constructor {
    super (list of
    four wheelers)

Four Wheeler Parking Spot

@override
getPrice ( ) {
    150 per hour
}

Two Wheeler Parking Spot

@override
getPrice ( ) {
    50 per hour
}

ParkingFactory

getPSM (vehicle Type) {
return obj based on obj

Parking Spot Manager

List < ParkingSpot >

addParkingSpot ( )
remove ParkingSpot)
Find ParkingSpot)

Parking Spot

+ id : int
+ VehicleType :
  VehicleEnum
+ isAvailable :
  boolean
+ Price : int
+ Vehicle : Vehicle

+ ParkVehicle ( )
  {
    isAvailable
    = false
    Vehicle = vehicle
  }
+ unpark ( )
  {
    isAvailable
    = true
    vehicle=null
  }

Two wheeler Parking Spot Manager

Const
super (List of two wheelers)

Four wheeler Parking spot Manager

Constructor
super (List of four wheelers)

Ticket

entryTime
Parking Spot

bookTicket

Vehicle

id
Type

Enum

2 wheel
4 wheel

Four wheeler Parking Spot

@override
getPrice ( ) {
150 per hour
}

Two wheeler Parking spot

@override
getPrice ( ) {
50 per hour }

Entrance gate

ParkingFactory
       Obj
ParkingSpot Manager
       obj
Vehicle
Ticket

---

get parking spot
(vehicle, Strategy)
Book spot
(vehicle)
generateTicket ( )

Parking Strategy

default {
   50 Per hour
}

Near to Entrance

{

Near to Elevator

payment

Card   Cash

Exit Gate

Cost calculation
Factory
Cost calculation

getprice ())
payment();
Parking spot
update())

Cost calculation Factory

Cost calculation

Pricing strategy
Const (PS PS)
this. PS = PS;

getprice ( );
PS.

Two wheeler
hourly

4 wheeler
mins

Pricing Strategy

default {
50 rupees

}

Hourly

Price {
hours x price

minutes

Price {
mins x price

ParkingFactory

getPSM (vehicle
Type) {
return obj
based on obj

Parking Spot Manager
List < Parking Spots>

addParkingSpot ( )
remove ParkingSpot ( )
Find Parking Spot( )

Parking Spot
+ id : int
+ VehicleType:
  Vehicle enum
+ is Available:
  boolean
+ Price : int
+ Vehicle : Vehicle

+ Park Vehicle ( )
{ isAvailable
  = False
  Vehicle = Vehicle

+ UnPark ()
{ isAvailable
  = true
  Vehicle = null

Two wheeler
Parking spot
manager
Const :
super (Street
two wheeler)

Four Wheeler
Parking spot
manager
Constructor
super (three of
four Wheeler)

Ticket
entryTime
Parking
Spot
bookTicket

vehicle
id
Type

Enum
2 wheel
4 wheel

Four Wheeler
Parking Spot
@override
getPrice ( )
150 Per hour

Two Wheeler
Parking spot
@override
getPrice ( )
50 Per hour

Entrance gate
parkingFactory
obj
parkingSpot manager
obj
vehicle
Ticket

get parkingSpot
(vehicle, Strategy)
Book spot
(vehicle)
generateTicket ( )

Parking Strategy
default {
50 Per hour
}

Near to
Entrance
price
100

Near to
Elevator
price
120