Tic Tac Toe

Contents

[1. Overview 2](#_Toc178697556)

[2. Requirement Gathering 2](#_Toc178697557)

[Requirement from Visualization: 2](#_Toc178697558)

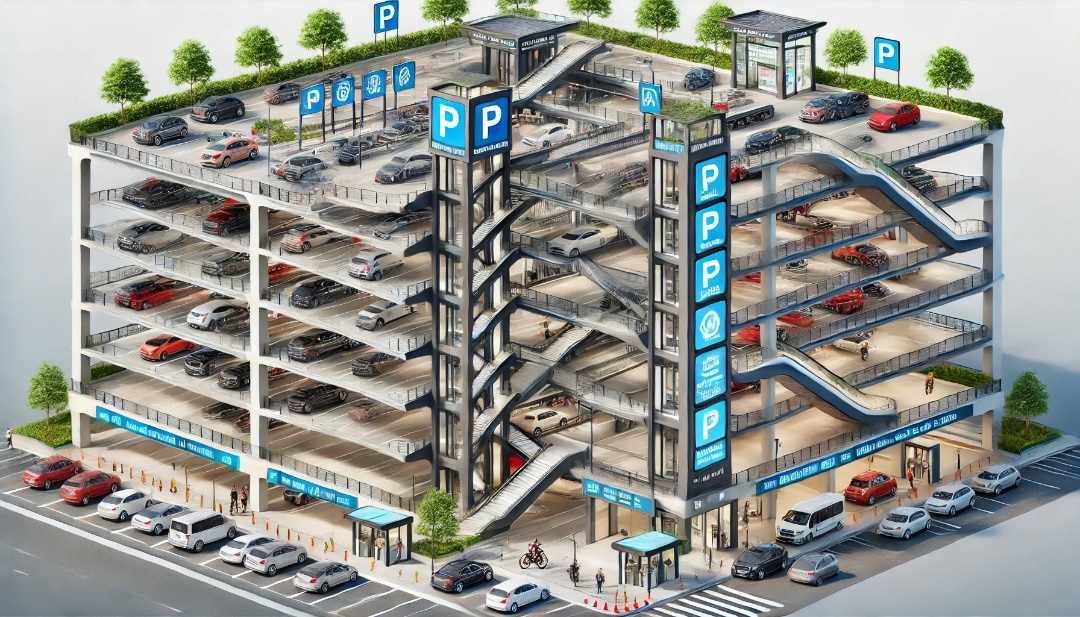
[Requirements from User Journey 3](#_Toc178697559)

[Clarify Requirements 3](#_Toc178697560)

Code and Notes are @ <https://github.com/nishithjain/Design_TicTacToe>

# Overview

* There are 2 different paths to get the overview of the system.
  + We know the system.
  + We don't know the system.
* Assume, we don't know the system.
* If we don't know the system, as questions. What kind of parking lot you are looking for?
* In response to this question, we get...
  + Design a parking management system for public parking lots.
  + For example, Airport, Hospitals, Malls.
  + We need to design Multi-Level Car Parking.



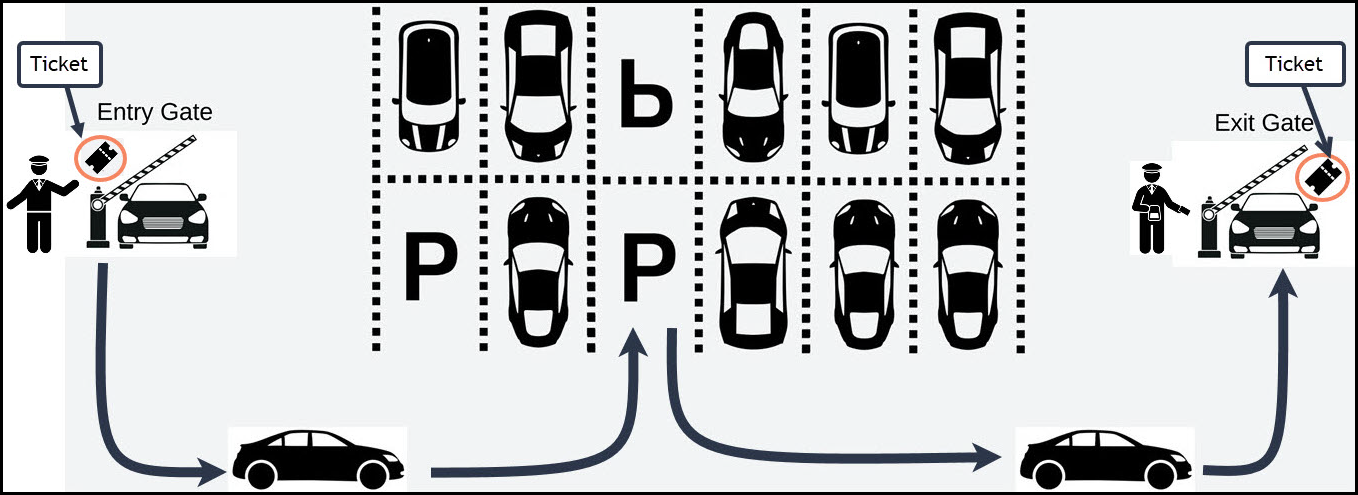
# Requirement Gathering

* Once we get the overview, let's talk about the requirements.
* If we remember, from the design point of view/class diagram point of view, we have to come from outside of the system to inside...

## Requirement from Visualization:

* Parking lot will have multiple floors.
* Multiple entry points.
* Multiple exit points.
* Multiple types of vehicles. (2-wheeler, 3-wheeler, cars, buses)
* Different types of parking spots for different types of vehicles.

## Requirements from User Journey

* At the entry gate, a ticket will be generated.
* Where the vehicle gets parked?
  + User will decide where to park the vehicle.
  + Operator will assign a parking spot at the entry gate.
* Spot assignment will happen at the entry gate.
* Store the operator details for entry gate and exit gate.
* Payment will happen at exit gate.
* User can make the payment via cash/UPI. For all online transactions, we will use Razor pay.
* Ticket should contain the vehicle information, entry time, entry gate details.

Note: Electric Vehicle parking spots are also supported.

## Clarify Requirements

* Parking Spot assignment can be done in multiple ways.
  + Near to lift lobby.
  + Near to exit gate.
  + Particular floor.
  + EV Parking Spot.
* We can use Strategy design pattern to assigning parking spot.
* Fee Calculation can be done in multiple ways.
  + Charges per hour

|  |  |  |
| --- | --- | --- |
| Hours | | Charge/Hour |
| Base Price | 0 to 4 | 50 |
|  | 5 to 10 | 1.2 X Base Price |
| 11 to 15 | 1.5 X Base Price |

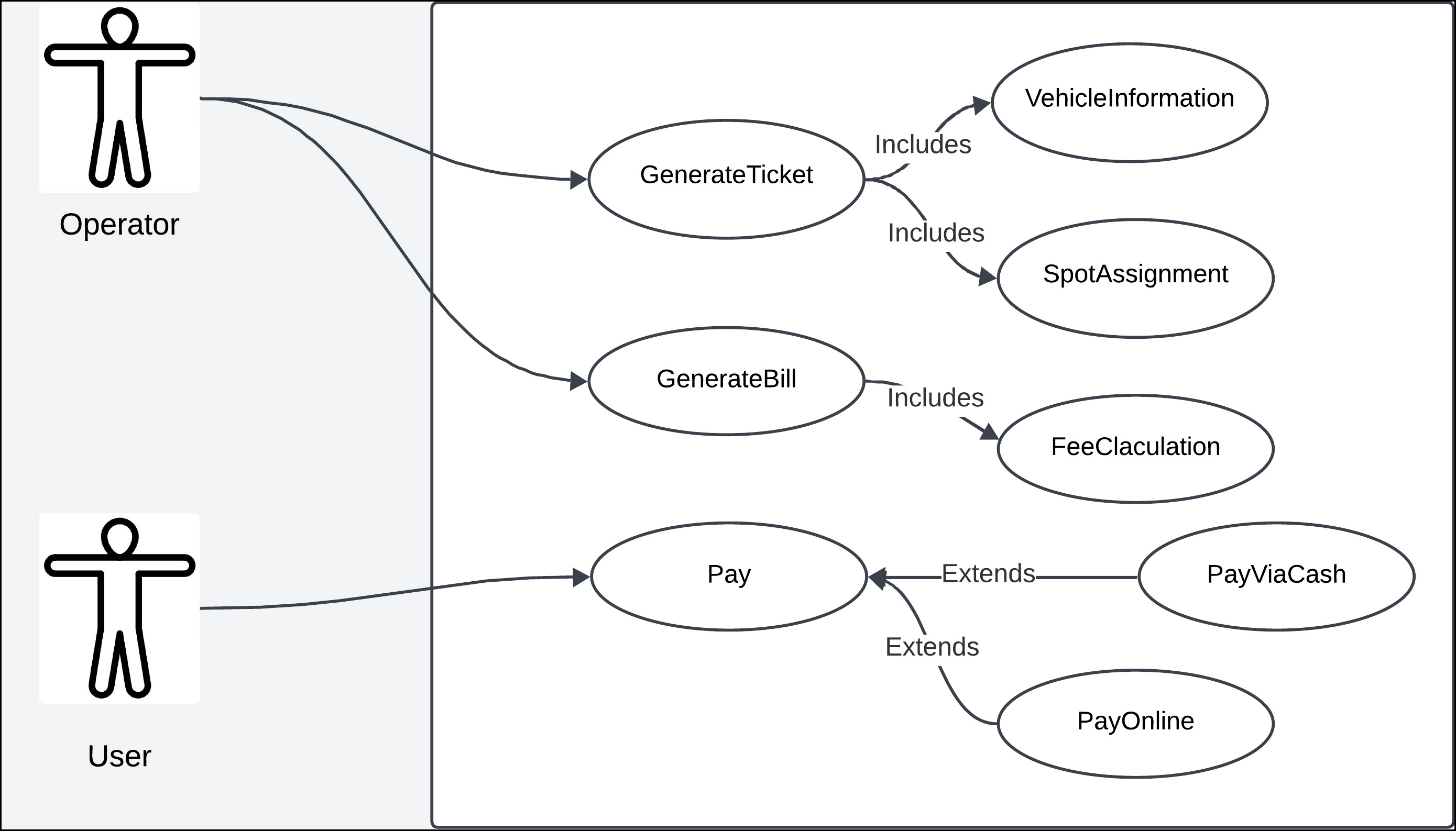
Example: For 12 hours of parking…

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hours | | Charge/Hour | For 12 Hours | |
| Base Price | 0 to 4 | 50 | 50 X 5 | 250 |
|  | 5 to 10 | 1.2 X Base Price | 1.2 X 5 X 5 | 300 |
|  | 11 to 15 | 1.5 X Base Price | 1.5 X 5 X 2 | 150 |
| Total | | | | 700 |

* + Flat charge/day
* We can use Strategy design pattern to calculate fee.
* Partial payment feature.
  + User can pay the fee from different modes of payment.
  + Example: If fee is 500Rs, 200 from wallet + 300 from UPI.
* For EV, if the vehicle is charged, then total fee = parking fee + electricity consumed.

# Use Case & Class Diagram

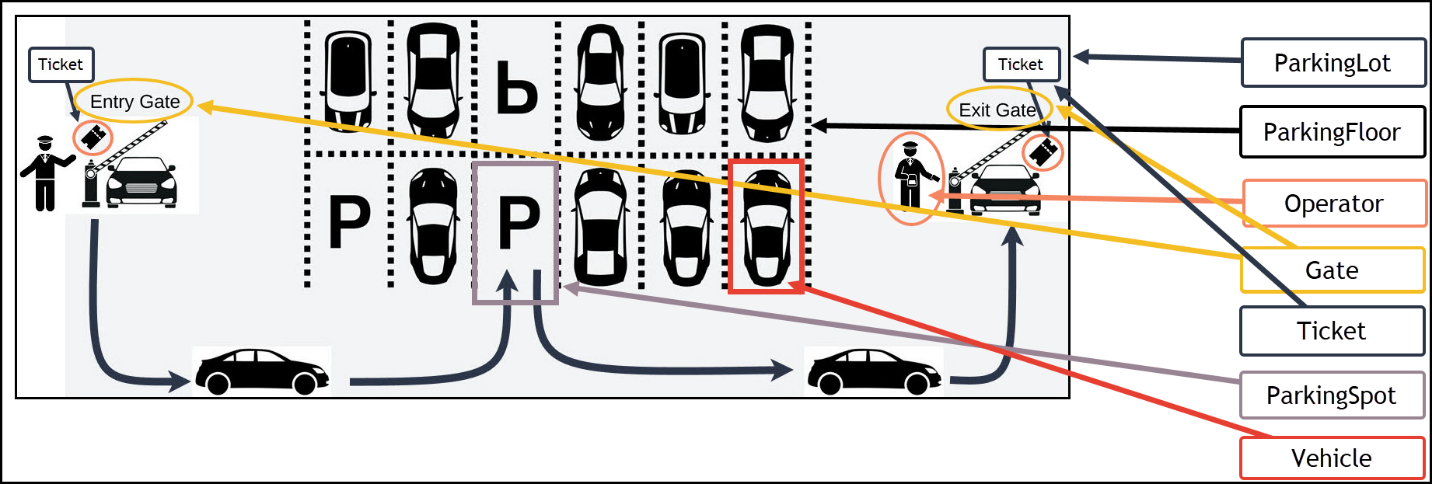
## Use Case Diagram



## Class Diagram

* To come up with class diagram,
  + Visualize the system.
  + Identify the nouns in the requirement.

### Visualize the system

* Let’s visualize the system and list all the nouns first.
* Let’s come from Outside while identifying the nouns…
  + ParkingLot class
  + ParkingFloor class
  + ParkingSpot class
    - ParkingSpot is internal to a floor. So, ParkingFloor has list of ParkingSpot.
    - ParkingFloor is internal to parking lot. So, ParkingLot has list of ParkingFloor.
  + GateOperator class
  + Gate class
    - ParkingLot should have all the gates which belong to that parking lot. Hence, we need list of Gate.
    - ParkingLot might also store all gate operator details. Hence, we need list of GateOperator.
  + Ticket class
  + Vehicle class

### Nouns from the requirement

* Let’s go through the requirement again and identify the nouns…