**Parking Lot System**

**1. Floor**

* floor\_id (PK)
* floor\_number (e.g., 1, 2, B1)

**2. Slot**

* slot\_id (PK)
* floor\_id (FK → Floor)
* vehicle\_type\_id (FK → VehicleType)
* is\_occupied (boolean)

**3. VehicleType**

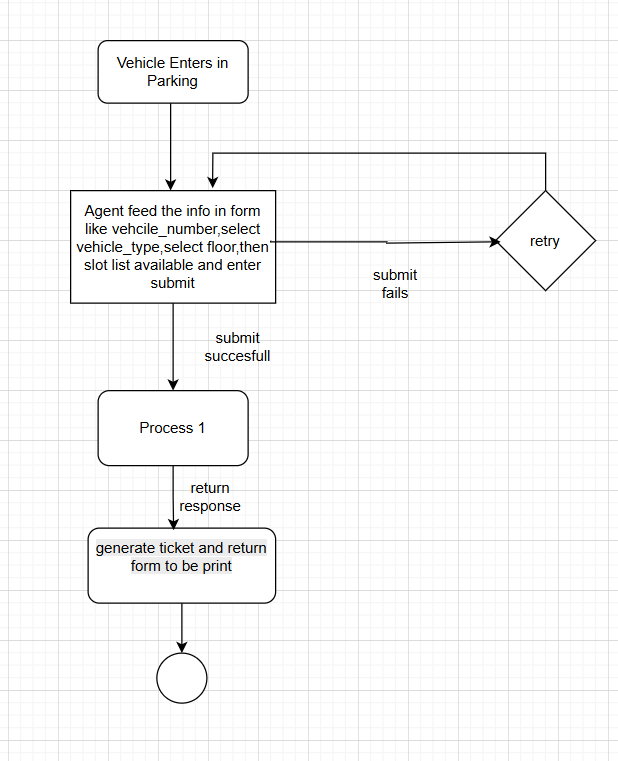
* vehicle\_type\_id (PK)
* name (Car, Bike, Truck, EV, etc.)

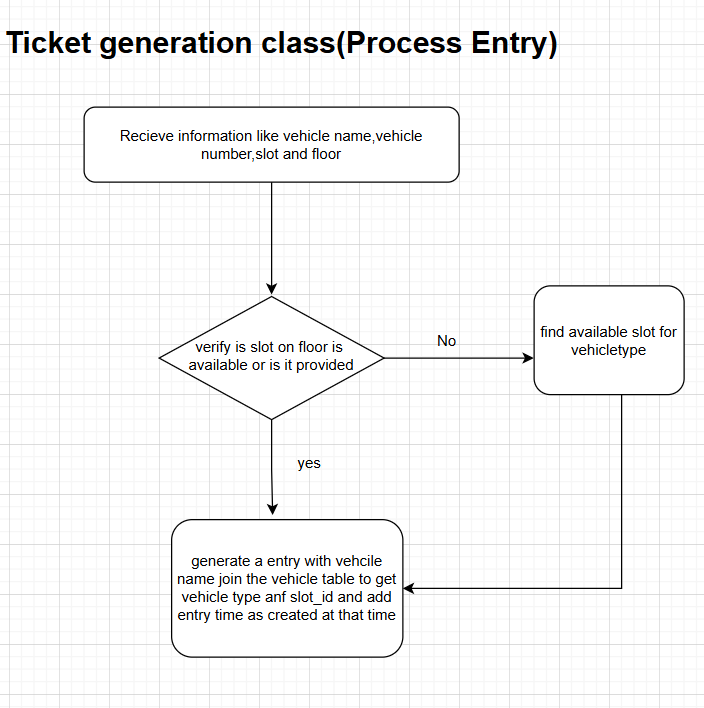
**4. Pricing**

* pricing\_id (PK)
* vehicle\_type\_id (FK → VehicleType)
* price\_per\_minute
* price\_per\_hour

**5. Ticket (Entry/Exit transaction)**

* ticket\_id (PK)
* slot\_id (FK → Slot)
* vehicle\_type\_id (FK → VehicleType)
* vehicle\_number (license plate)
* entry\_time
* exit\_time (nullable until exit)
* total\_price (calculated at exit)
* status (Active, Closed, Cancelled)





**Flow of GenerateToken Class**

1. Input Received
   1. vehicle\_number (e.g., DL10AB1234)
   2. vehicle\_type (Car, Bike, etc.)
   3. (Optional) slot\_number, floor\_number
2. Validate Slot & Floor
   1. If slot\_number and floor\_number are provided from UI: ✅ Use them directly.
   2. If null → call Slot service/class to:
      1. Find an available slot on a floor where:
         1. slot.vehicle\_type\_id == vehicle\_type\_id
         2. slot.is\_occupied == false
3. Resolve VehicleType ID
   1. Query VehicleType table by name (Car, Bike, etc.) → get vehicle\_type\_id.(Vehicle class)
4. Create Ticket Entry (Ticket class)
   1. Generate ticket\_id (GUID/UUID).
   2. Insert a record into Ticket table:
      1. ticket\_id → random GUID
      2. slot\_id → from UI or selected by Slot class
      3. vehicle\_type\_id → from VehicleType
      4. vehicle\_number → input value
      5. entry\_time → DateTime.Now
      6. exit\_time → NULL (until exit happens)
      7. total\_price → 0 initially
      8. status → Active
5. Update Slot Occupancy
   1. Update the chosen slot.is\_occupied = true.(Slot class)
6. On Exit
   1. When vehicle exits, fetch ticket by ticket\_id. (Ticket class)
   2. Calculate duration = exit\_time - entry\_time.(Pricing class)
   3. Fetch pricing from Pricing table using vehicle\_type\_id.
   4. Compute total\_price.
   5. Update ticket.exit\_time, ticket.total\_price, and ticket.status = Closed.(Generate token class)
   6. Free the slot → slot.is\_occupied = false.(slot class)