PARKING MANAGEMENT SYSTEM

Expectation.

1. Able to assign parking spots on 1st come 1st serve basis.
2. 20 percent reservation for differently abled users.
3. Able to list available parking spots.
4. Able register user and vehicle information.
5. Extensible for future development.

Components:

Framework: django-python(quick development)
Option : nodejs(express)(Best io)
Database: AnyNoSql,or Sql(worked on orm, switch db will not be a issue)
Deployment: CI-CD(Best for Micro service).
Jenkins: to trigger
Monitoring: Aws services
Architecture: Preferred is microservice.

Services:

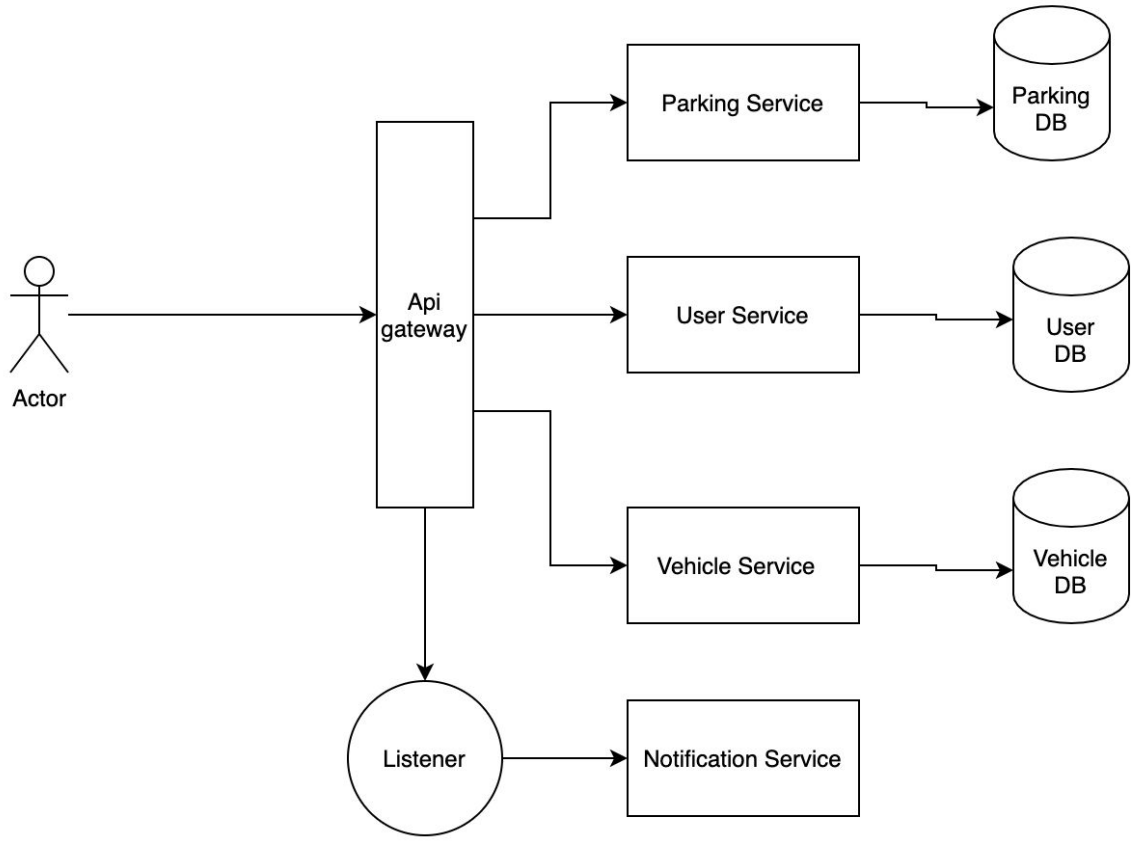
We can think of a microservice architecture as we want extensible features and separation of concerns. One service should not affect the operation of another service.

We can add extra features like

1. subscription model.
2. Monthly/daily pass.
3. Valet services
4. Pre booking.
5. Payment.
6. Reminders.

I this doc we covering api signatures and a high level approach to design a Parking management system.

You can find code on this link
https://github.com/Abhiintheweb/parking-management-system



User service.

Will store all information of the user. User personal information will be required.

Database: user_management_service

Table: user

Id: integer

Name: varchar(100)

phone_number: varchar(15)

role: varchar(20)

Is_Active: boolean

is_deleted: boolean

created_by: integer

updated_by: integer

created_at: datetime

updated_at: datetime

API:

1. **GET** <baseUrl> **v1/user/<id>**

pathParm: id(userId)

Response: {

Id: integer,

Name: integer,

phoneNumber: string,

Role: string

}

2. POST <baseUrl> v1/user

Request: {

Name: integer,

phoneNumber: string,

}

Response:

{

Id: integer,

Name: integer,

phoneNumber: string,

Role: string

}

Vehicle service:

Will store vehicle information.

Database: Vehicle_management

Table: Vehicle

Id: integer

Name: varchar(100)

Vehicle_number: varchar(15)

vehicle_type: varchar(20)

is_deleted: boolean

created_by: integer

updated_by: integer
created_at: datetime
updated_at: datetime
User_id: integer

API

1. GET <baseUrl>/v1/vehicle/<id>
   Response :{
   vehicleNo:string,
   vehicleType: string,
   userId: integer,
   vehicleType: string,
   }

2. POST: <baseUrl>/v1/vehicle/<id>
   Request:{
   name: string,
   vehicleNumber: string,
   userId: integer
   }
   Response:
   {
   vehicleNo:string,
   vehicleType: string,
   userId: integer,
   vehicleType: string,
   }

3. PUT <baseUrl>/v1/vehicle/<id>

   Request:{
   name: string,
   vehicleNumber: string,
   userId: integer
   }
   Response:
   {
   vehicleNo:string,
   vehicleType: string,
   userId: integer,
   vehicleType: string,

}

Parking service

We will store parking information.

Database: parking_management

Table: **parking**

id: integer,
name: varchar(20),
Is_reserved: bool,
Is_deleted: bool,
Is_active: bool,
is_blocked:bool,
blocked_by_vehicle_id: integer,
created_by: integer
updated_by: integer
created_at: datetime
updated_at: datetime

Parking_map_vehicle:

Id: number
Parking_id: number,
Vehicle_id: number,
Parking_start_datetime: datetime,
Praking_end_datetime: datetime,
Is_deleted: boolean

API

1. **Get Parking**

GET <baseUrl>/v1/parking?isBlocked=<bool>&id=1&limit&offset=

Response:

```
[{
      Id:integer,
      Name: string,
      isBlocked:false,
      blockedBy:integer,
      isReserved: bool
}]
```

2. Create Parking
   POST <baseUrl>/v1/parking
   Request: {
   Name: string,
   isReserved:true

   }
   Response:
   {
   Id:integer,
   Name: string,
   isBlocked:false,
   blockedBy:integer,
   isReserved: bool
   }

3. Update Parking
   PUT <baseUrl>/v1/parking

   Request: {
   Name: string,
   isReserved:true

   }

   Response:
   {
   Id:integer,
   Name: string,
   isBlocked:false,
   blockedBy:integer,
   isReserved: bool
   }
4. Parking Interaction
   POST <baseUrl>/v1/parking-interaction
   Request {
   parkingId: integer,
   vehicleId: integer,
   parkingStartTime: integer

   }
   Response:
   {

Message: Vehicle Parked.

}

5. Update Parking:
   PUT <baseUrl>/v1/parking-interaction/<id>
   Request {}
   response:{pakingTime:10 HRS.}