System design

Project Name	EZParking
Team	Apollo
Data Produced	2022/11/20

Clarify the cope of the System

- **User cases** (description of sequences of events that, taken together, lead to a system doing something useful)
    - Who is going to use it?

      People who need to drive and can't find parking.
    - How are they going to use it?

      Enter the building you need to go to (eg.ED), and then the background will check the status of the parking lot and give suggestions to the user.

- **Constraints**
    - Mainly identify **traffic and data handling** constraints at scale.

      Use Redis to support large-scale reading and writing. It is faster to access the cache directly because it does not need to be stored on disk.
    - Scale of the system such as requests per second, requests types, data written per second, data read per second)

      About 9,000 times per second
    - Special system requirements such as multi-threading, read or write oriented.

      Spring boot, Mybaits-plus, multi-threading, Redis

High-level architecture design

- Sketch the important components and connections between them, but don't go into some details.
    - Application service layer (serves the requests)

      Linux

    - List different services required.

      Have a Jdk runtime environment

    - Data Storage layer

      Mysql, Redis, Mybatis

- **Component Design**
    - Component + specific **APIs** required for each of them.

      Login, Register, Curd of waypoint, Curd of parking lot, Set the destination, Curd Parking lot capacity

- **Object oriented design** for functionalities.
    - Map features to modules: One scenario for one module.
    - Consider the relationships among modules:
        - Certain functions must have unique instance (Singletons)

          Login， Register，parking lot curd

        - Core object can be made up of many other objects (composition).

          When creating waypoint, you can store the parking structure and route separately by using different instance,
          The increase and decrease of parking capacity,
          Set the destination

        - One object is another object (inheritance)

          Waypoint: set destination, parking lot

- **Database schema design.**

  Waypoint：

  Type_node: String

  Long: float

  Lat: float

  Neib：String

  User：

  Id: String

  Name: String

  Password: String

  Role: String

  Email: String

Bottlenecks

- Perhaps your system needs a load balancer and many machines behind it to handle the user requests. * Or maybe the data is so huge that you need to distribute your database on multiple machines. What are some of the downsides that occur from doing that?

  Consistency of data can be difficult, often using MQ to uniformly send messages to servers that process the same thing.

- Is the database too slow and does it need some in-memory caching?

Access to the cache is needed to reduce the number of times users have to access data to get parking lot information.

Scaling

- **Vertical scaling**
  - You scale by adding more power (CPU, RAM) to your existing machine.

    Kafka，Cpu，4G RAM

- **Horizontal scaling**
  - You scale by adding more machines into your pool of resources.

    Distribute computing

- **Caching**
  - Redis