Log In Join

Back To Module Home

Design Problems

0% completed

Search Module

Design a Collaborative Document Editing Service / Google Docs

Conclusion

Mark Module as Completed

Requirements of Uber’s Design

Learn about the requirements to design an Uber service.

We'll cover the following

Requirements#

Let’s start with the requirements for designing a system like Uber.

Functional requirements#

The functional requirements of our system are as follows:

- **Update driver location**: The driver is a moving entity, so the driver’s location should be automatically updated at regular intervals.
- **Find nearby drivers**: The system should find and show the nearby available drivers to the rider.
- **Request a ride**: A rider should be able to request a ride, after which the nearest driver should be notified about the rider’s requests.
- **Manage payments:** At the start of the trip, the system must initiate the payment process and manage the payments.
- **Show driver estimated time of arrival (ETA):** The rider should be able to see the estimated time of arrival of the driver.
- **Confirm pickup:** Drivers should be able to confirm that they have picked up the rider.
- **Show trip updates**: Once a driver and a rider accept a ride, they should be able to constantly see trip updates like ETA and current location until the trip finishes.
- **End the trip:** The driver marks the journey complete upon reaching the destination, and they then become available for the next ride.

Point to Ponder

Question

What if two drivers are at the same distance from the rider? How will we select the driver to whom we’ll send the request?

Show Answer

Non-functional requirements#

The non-functional requirements of our system are as follows:

- **Availability:** The system should be highly available. The downtime of even a fraction of a second can result in a trip failure, in the driver being unable to locate the rider, or in the rider being unable to contact the driver.
- **Scalability:** The system should be scalable to handle an ever-increasing number of drivers and riders with time.
- **Reliability:** The system should provide fast and error-free services. Ride requests and location updates should happen smoothly.
- **Consistency:** The system must be strongly consistent. The drivers and riders in an area should have a consistent view of the system.
- **Fraud detection:** The system should have the ability to detect any fraudulent activity related to payment.

Resource estimation#

Now, let’s estimate the resources for our design. Let’s assume it has around 500 million riders and about five million drivers. We’ll assume the following numbers for our estimates:

- We have 20 million daily active riders and three million daily active drivers.
- We have 20 million daily trips.

- All active drivers send a notification of their current location every four seconds.

Storage estimation#

Let’s estimate the storage required for our system:

Rider’s metadata#

Let’s assume we need around 1,000 Bytes to store each rider’s information, including ID, name, email, and so on, when the rider registers in our application. To store the 500 million riders, we require 500 GB of storage:

$$500 \times 10^6 \times 1000 = 500\ GB$$

Additionally, if we have around 500,000 new riders registered daily, we’ll need a further 500 MB to store them.

Driver’s metadata#

Let’s assume we need around 1,000 Bytes to store each driver’s information, including ID, name, email, vehicle type, and so on, when the driver registers in our application. To store the five million drivers, we require 5 GB of storage:

$$5 \times 10^6 \times 1000 = 5\ GB$$

Additionally, if we have around 100,00 new drivers registered daily, we’ll need around 100 MB to store them.

Driver location metadata#

Let’s assume we need around 36 Bytes to store the driver’s location updates. If we have five million drivers, we need around 180 MB of storage just for the drivers’ locations.

Trip metadata#

Let’s assume we need around 100 Bytes to store single trip information, including trip

ID, rider ID, driver ID, and so on. If we have 20 million daily rides, we need around 2 GB of storage for the trip data.

Let’s calculate the total storage required for Uber in a single day:

Storage Capacity Estimation

Number of drivers (millions)	5
Storage required to store a driver's location (Bytes)	36
Total storage required to store drivers’ locations (MB per day)	*f* 180
Number of trips (millions)	20
Storage required to store a trip (Bytes)	100
Total storage required to store trips (GB per day)	*f* 2
Storage required for new riders daily (MB per day)	500
Storage required for new drivers daily (MB per day)	100
Total storage (GB per day)	*f* 2.78

Total storage required by Uber in an year

> **Note:** We can adjust the values in the table to see how the estimations change.

Bandwidth estimation#

We’ll only consider drivers’ location updates and trip data for bandwidth calculation since other factors don’t require significant bandwidth. We have 20 million daily rides, which means we have approximately 232 trips per second.

$$\frac{20000000}{86400} \approx 232\ trips\ per\ second$$

Now, each trip takes around 100 Bytes. So, it takes around 23 KB per second of bandwidth.

$$232 \times 100\ =\ 23\ KB \times 8\ =\ 185\ kbps$$

As already stated, the driver’s location is updated every four seconds. If we receive the driver’s ID (3 Bytes) and location (16 Bytes), our application will take the following bandwidth:

$$3M\ active\ drivers \times (3+16)B = 57MB \times 8\ =\ \frac{456}{4}\ = 114\ Mbps$$

These bandwidth requirements are modest because we didn’t include bandwidth needs for maps and other components that are present in the real Uber service.

Bandwidth Requirements

Trips per second	232
Bandwidth for each trip (Bytes)	100
Total bandwidth for trips (kilo bits per second)	*f* 185.6
Active drivers (millions)	3
Bandwidth for each driver (Bytes)	19
Bandwidth for drivers (Megabits per second)	*f* 114

Total bandwidth (Megabits per second)	*f* 114.19

Total bandwidth required by Uber

> **Note:** We can adjust the values in the table to see how the requirements change.

We’ve ignored the bandwidth from the Uber service to users because it was very small. More bandwidth will be required for sending map data from the Uber service to users, which we’ve also discussed in the Google Maps chapter.

Number of servers estimation#

We need to handle concurrent requests coming from 20 million daily active users. We’ll use the following formula to estimate a pragmatic number of servers. We established this formula in the Back-of-the-envelope Calculations chapter:

$$\frac{Number\ of\ daily\ active\ users}{RPS\ of\ a\ server} = \frac{20 \times 10^6}{8000} = 2500$$

Number of servers required for the Uber service

Estimating the Number of Servers

Daily active users (millions)	20
RPS of a server	8000
Number of servers required	2500

> **Note:** We can adjust the values in the table to see how the estimations change.

Building blocks we will use#

The design of Uber utilizes the following building blocks:

Building blocks in high-level design of Uber

- **Databases** store the metadata of riders, drivers, and trips.
- **A cache** stores the most requested data for quick responses.
- **CDNs** are used to effectively deliver content to end users, reducing delay and the burden on end-servers.
- **A load balancer** distributes the read/write requests among the respective services.

Riders’ and drivers’ devices should have sufficient bandwidth and GPS equipment for smooth navigation with maps.

> **Note:** The information provided in this chapter is inspired by the engineering blog of Uber.

System Design: Uber

Next

High-level Design of Uber

Mark as Completed

Report an Issue