**Stock Data:** Imagine you are building some sort of *service* that will be called by up to *1,000 client applications* to get simple *end-of day stock price* information (open, close, high, low). You may assume that you *already have the data*, and you can store it in any format you wish.

How would you design the client-facing service that provides the information to client applications? You are responsible for the development, rollout, and ongoing monitoring and maintenance of the feed.

Describe the different methods you considered and why you would recommend your approach. Your service can use any technologies you wish, and can distribute the information to the client applications in any mechanism you choose

**Solution:**

Major Functionality is that the web service based application will provide the client with the end of day stock data (open close high low) with the previously available data.

**Web Service:** REST API can be used as it tends to be more extensible even though the latency is high compared to the RPC and the GraphQL is quite complex. It works on a resource as a fundamental unit and so it is more like maintaining the object (Stock) and mostly we are in need to reuse this for various extensible operations

**1000 Client:** Since the amount of application which leverage the web service is large we could use the load balancer and the scale the application server so as to handle those huge volume of web service request. Also we cannot be sure about the security mechanism provided by the user and so we need to have security mechanism in place before we reach the application server to ensure no attack are waged. We could implement the sanitization of the user inputs at a server side to ensure that SQL Injection and the Cross Site Scripting attacks are thwarted and one another major attack would be to flood the web service requests and create a Denial of Service for the actual users of the application. We would require in this case maybe configure in such a way that the certificates are used in a two way authentication to ensure only valid user of the application are requesting the service

**Data format:** JSON – high speed and it is very much readable in comparison to XML based data. It will be much like the js objects

Database: We can have a relational database SQL as the functionality basically seem to revolve more around the Stock and so it is ideal if we use a ORM based framework and also leverage the features like ACID, transaction, indexing

Assumption #1: Read heavy as we are only proving the already available data to the application that invoke the request and only periodical data is need of updating, we can leverage the cache mechanism. i.e., when an application requests a stock of a company for the first time we would need to retrieve it and cache so that when change happens we can invalidate it (Cache Invalidation has to be handled)

Assumption #2: We assume that we will need to provide stock details of multiple companies.

**Cache**

**Clients 1000**

**1**

2

3

N

Internal implementation is a black box to external

**Load Balancer**

**Security Mechanism**

Sanitize the requests

Find the possible DOS and validate if necessary

**Hacker**

**Master - Master**

1 – First the load balancer checks the cache

2 – When the data is not available then it seeks the application server to respond to the request

3 – The application server which has the web service implementation will compute the data from the read replica and then provide the results

4 – The data is now stored and then returned to the user

Whenever the data changes in **Master** then we invalidate the cache

**Read Replica**

**1**

2

3

N

**Application Servers**