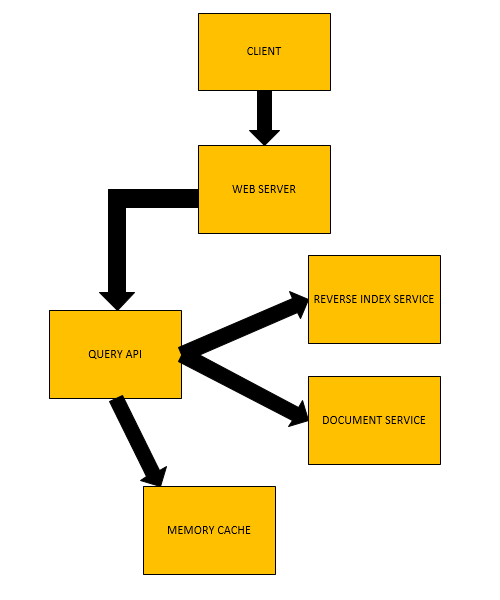
System Design:

Key-value collection service with a Query API in JSON that would provide the metrics for the response times for the apps.



**CLIENT:**

The client sends request to the web server, i.e. whenever a user interacts with an app, it goes through to a web server and then moves forward from there.

**WEB SERVER:**

The web server interacts with the clients and also with the query API service and forwards any requests from the client to the query API.

**QUERY API SERVER:**

Parses the query by the user and checks for any errors in the query and fixes them. If the user’s query results in a cache hit then it will check the Memory cache and update the contents of the cache to the front of the LRU list (least recently used).

If the query results in a cache miss, then it will use the reverse index service to check documents matching the query. The reverse index query will rank the results and return the ones at the top. It will also use the document service to return the title. Updating the memory cache is also done with the contents being placed at the front of the LRU list.

The cache should be updated when the page contents change or when a page is removed or added or when a page rank changes.

**Scalability:**

This design can work up until certain point and it would not work when we start scaling up the system. When the number of users start increasing, it would start crashing down since the query API would not be able to take the load. Also a single web server would not be able to take the load for multiple users at a single time. So for this purpose we need to introduce load balancers into the system. The load balancers can distribute the user requests to different application servers and databases.

Benefits with using load balancers also include decryption and encryption so that backend servers do not have to do those operations and also session persistence in case if web apps do not keep track of sessions.

To improve availability and performance also load balancers can be used for the purpose of horizontal scaling. It is more cost efficient than scaling a single server on expensive hardware.

The problem with using load balancers though, is that it becomes increasingly complex once the numbers increase. It might also be a performance bottleneck if the configuration is not done right.

The other aspect of scalability involves scaling the database and also involving the master slave architecture for the purpose of improved query performances as well as for the increase in the number of requests being handled. There are plenty of scale a relational database like master-slave replication, master-master replication, federation sharding, denormalization and SQL tuning.