# — DAY 2 ASSIGNMENT —

**LetsUpgrade** **Pratap Kumar Bal**

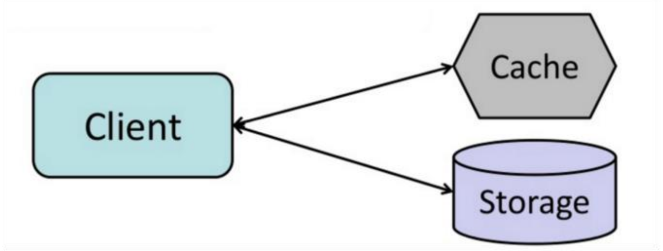
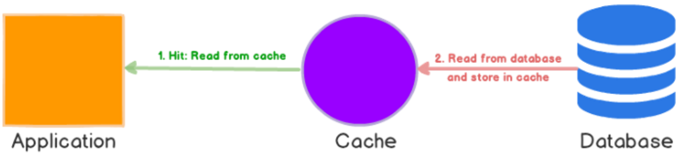
## "System Design: Zero to Hero"

### Q1: Define the following cache strategies:

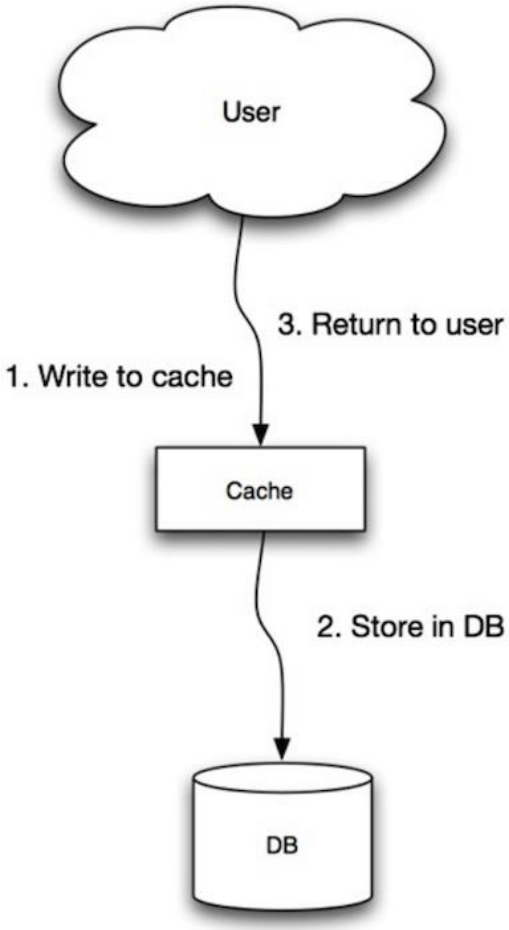
- Cache Aside
- Read Through
- Write Through
- Write Behind

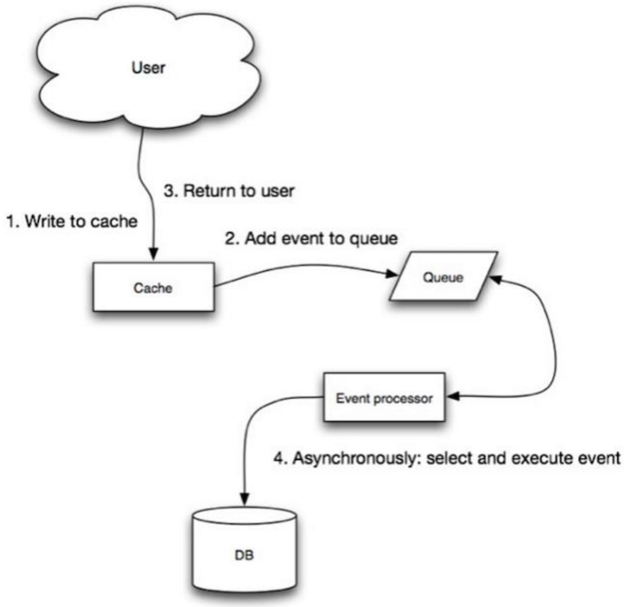
**A1:**

A Cache is a reserved storage location that collects temporary data to help websites, browsers, and apps load faster. Caching improves latency and can reduce the load on your servers and databases.

| ***Cache Aside:*** | ***Read Through:*** |
|---|---|
|  |   1. Hit: Read from cache<br>2. Read from database and store in cache<br>Application<br>Cache<br>Database |
| This is the most commonly used cache update strategy in applications. In this update strategy, cache sits aside and an application talks to cache and data store directly. It is also known as lazy-loading. Application logic first checks in the cache before hitting the database. It is mostly used with an application with read-heavy workloads. | Read-through cache strategy is quite similar to cache-aside accept instead of managing both data store and the cache, delegates data store synchronization to the cache provider. Both cache-aside and read-through loads data lazily on the first read. |

# — DAY 2 ASSIGNMENT —

<table>
<tr><th><em>Write Through:</em></th><th><em>Write Behind:</em></th></tr>
<tr><td>


</td><td>


</td></tr>
<tr><td>In this strategy, an application uses the cache as the main data source to read and write. Cache sits between application and data store. The cache is responsible for reading and writing to the database.</td><td>In write-behind sometimes also known as write-back cache strategy, data is first written to the cache and then asynchronously data is updated to the data store, improving write performance.</td></tr>
</table>

# — DAY 2 ASSIGNMENT —

## Q2: Explain Message Queues and PUB-SUB Queues. Mention where they are used.

**A2:**

A queue is a line of things waiting to be handled — in sequential order starting at the beginning of the line.

***Message Queues:***

The message queuing pattern takes on a point-to-point approach. A message within the queue will be deleted once consumed, similar to the Post Office Protocol, where a message is deleted from the server once it is delivered. These queues enable asynchronous messaging.

In the event of a network issue that causes a delay in a message's delivery, like if a consumer is unreachable, that message will stay in the queue until it can be delivered. This means messages aren't necessarily delivered in any particular order. They are instead delivered on a first-available basis, which can boost efficiency in certain environments.

***PUB-SUB Queues (Public Subscribe Queues):***

The publish-subscribe pattern, often called the pub-sub pattern involves publishers that produce ("publish") messages in different categories and subscribers who consume published messages from various categories they are subscribed to. Unlike point-to-point messaging, a message will only be deleted if it's consumed by all subscribers to the category.

Some message systems, such as Kafka, have a retention policy that ensures messages stay in the queue for a specified amount of time, even after they are consumed by all subscribers. Other providers include Google Cloud Pub/Sub messaging, which offers similar functionality. Both are examples of middleware message delivery services that offer scalability and decoupling potential for real-time data streams and workflows.