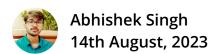




## **HLD - 7**



# Designing LeetCode Platform Continued: Storage, Caching, and Rate Limiting

**Storage Considerations:** 

Data to Store:



video URLs for tutorials.

- User Profile Database:
- Store user-specific information such as user ID, associated question IDs, preferred coding language, status, and code solutions.

#### **Databases Schema:**

### **Questions Database:**

- question\_id: Identifier for questions.
- test\_cases: Predefined test cases for code evaluation.
- code\_url: URL to access the code of the question.
- video\_url: URL to access tutorials for the question.

#### **User Profile Database:**

- user\_id: Unique identifier for each user.
- question\_id: IDs of the questions attempted by the user.
- language: Preferred coding language.
- status: Status of the code solution (submitted, in progress, etc.).
- code\_solutions: User-submitted code solutions.

## **Indexing and Caching:**

## **Indexing:**

• Create an index on user\_id in the user profile database to optimize user data retrieval.



- Implement client-side caching to store frequently accessed user data locally.
- Use server-side caching to store static data like question details.

## SQL vs. NoSQL:

### SQL:

- Use SQL databases for structured and relational data, such as user profiles and question details.
- Facilitates complex queries and transactions.

## NoSQL:

- Utilize NoSQL databases for flexible schema needs, like storing user code submissions and test cases.
- Provides fast retrieval of unstructured data.

## Rate Limiting and Avoiding Attacks:

## Rate Limiting:

- Implement rate limiting to prevent excessive API requests from a single user, ensuring fair resource distribution.
- Set limits based on user roles (registered user, premium user, etc.).

#### **DDoS and DoS Attacks:**



Use Content Delivery Networks (CDNs) to absorb traffic spikes.

## Persistent vs. Non-Persistent Storage:

#### Persistent Storage (Hard Disk):

- Store critical and structured data, such as user profiles and question details.
- Ensures data integrity and durability.

#### Non-Persistent Storage (RAM):

- Implement caching mechanisms using RAM for faster data retrieval.
- Suitable for frequently accessed static data.

#### **Server Estimation:**

- Estimate the number of servers based on:
- Expected user traffic.
- Storage requirements.
- Complexity of queries and data processing.

## **Designing Google Drive:**

## Requirement Gathering:

## **Storage and CRUD Operations:**



• Design an intuitive interface for seamless CRUD operations.

## **Data Sharing:**

- Determine mechanisms to share files and folders with others.
- Set permissions and access levels for shared content.

#### File Size and User Limits:

- Specify the maximum size of uploaded files.
- Define per-user quotas for data storage.

## **Availability and Performance:**

• Address potential slow download issues and ensure data availability.

#### **Data Loss Tolerance:**

• Decide if data loss is acceptable and if redundant copies are needed.

## **Storage Estimation:**

• Estimate storage needs based on the maximum file size, the number of users, and their quotas.

## **Creating Files/Folders:**

 Define user-friendly processes to create and organize files and folders efficiently.



Assessment: <a href="https://www.bosscoderacademy.com/blog/hld-7-assessment">https://www.bosscoderacademy.com/blog/hld-7-assessment</a>

