

# Design Leetcode

Members: Tuan Le Hoang - Long Nguyen Hoang

# Agenda

1. Clarify Requirements
2. Back of the Envelope Estimation
3. API Design
4. Data Model Design
5. High-level Design
6. Detailed Design
7. Identify and Discuss Bottle necks

# 1. Clarify Requirements

# Functional Requirements

- View Problems
- Submit Solutions in multiple languages
- Join Coding Contests
- Post and discuss solution

# Non-Functional Requirements

- Availability: 99.9% uptime
- Scalability: 10K+ current submissions
- Latency: Code execution < 10s, leaderboard update: < 5s
- Security: Isolated code execution, prevent malicious code

# Out of Scope

- User Authentication
- Payment processing
- User analytics / User management

## 2. Back of the Envelope Estimation

- Daily Active Users: 500,000
- Contest participants: 10,000
- Problems in DB: 3000+, growth: 8 problems/week => 418 problems/year
- Leetcode has 26.3 million monthly visitors  
=>  $26.3 * 10^6 / 30 / 86000 = 10$  QPS
- Support 10+ popular programming Languages: Python 3, python 2, Java, C++, C, C#, C, Javascript, Typescript, Go, Swift, Rust, PHP, Kotlin...

# Submission Estimation

Daily submissions:

$500,000 \text{ DAU} * 3 = 1.5 \text{ M submissions/day}$

$\Rightarrow 1.5 * 10^6 / 86,000 = 17 \text{ submissions/second}$

$\Rightarrow 2 \text{ submissions/language/second}$

Peak submissions (contest):

$10,000 * 20 \text{ submissions} = 200,000 \text{ in 90 minutes}$

$\Rightarrow 2 * 10^5 / 5400 = 37 \text{ submissions/second}$

$\Rightarrow 4 \text{ submissions/language/second}$



# Storage Estimation

Per submission:

Code: ~10 KB

Metadata: ~1KB

Results: ~2 KB

=> Total: ~13 KB

=> Daily storage:  $1.5 \text{ M} * 13 \text{ KB} = 20 \text{ GB/day}$

=> Monthly storage growth: 600 GB/month

Problems + Test cases:

$3,000 * 50 \text{ test cases} * 10 \text{ KB} = 1.5 \text{ GB}$

# Bandwidth Estimation

Incoming (submissions):

$1.5 \text{ M} * 10 \text{ KB} = 15 \text{ GB/day}$

Outgoing

Problem views:  $10 \text{ M} * 50 \text{ KB} = 500 \text{ GB/day}$

Results:  $1.5 \text{ M} * 5 \text{ KB} = 7.5 \text{ GB/day}$

=> Total:  $\sim 520 \text{ GB/day} \approx 6 \text{ MB/s}$  average

## 3. API Design

# Problem APIs

```
GET /problems?page=1&limit=100  
-> Partial<Problem>[] : [{id, title, difficulty, tags...}]
```

```
GET /problems/{problem_id}  
-> {id, title, description, examples, constraints, starter_code, difficulty, tags, acceptance_rate...}
```

```
GET /problems/{problem_id}/solutions?sort=votes&page=1  
->
```

# Submission APIs

```
POST /problems/{problem_id}/run
```

```
Body: {code, language, test_input}
```

```
-> {results, runtime_ms, memory_kb}
```

```
POST /problems/{problems_id}/submit
```

```
Body: {code, language}
```

```
-> {submission_id}
```

```
GET /submissions/{submission_id}
```

```
-> {status: "pending" | "running" | "accepted" | "wrong_answer", results, runtime_ms, memory_kb}
```

# Contest APIs

GET /contests

-> {upcoming: [...], ongoing: [...], past: [...]}

GET /contests/{contest\_id}

-> {id, title, start\_time, end\_time, problems: [...]}

POST /contests/{contest\_id}/register

-> {success: true}

GET /contests/{contest\_id}/leaderboard?page=1&limit=50

{rankings: [{rank, user, score, finish\_time}]}

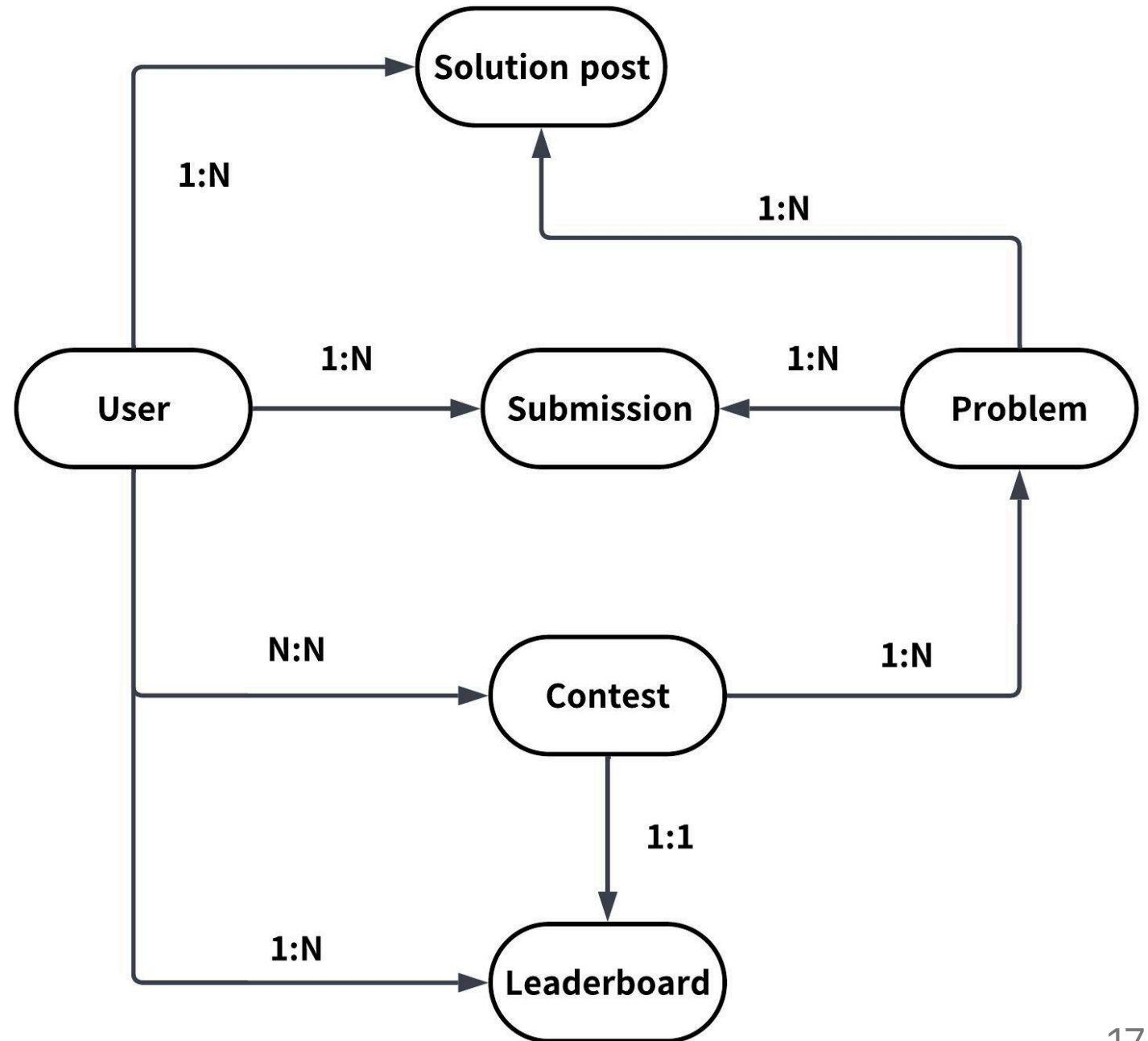
# User APIs

**GET:** profile, submissions, progress.

## 4. Data Model Design



## Entity Relationship Diagram



## 5. High level Design