Tiny URL

**Tiny URL - System Design**

<https://www.youtube.com/watch?v=Cg3XIqs_-4c&t=2s>

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a black and white screen

AI-generated content may be incorrect.

<https://www.hellointerview.com/learn/system-design/problem-breakdowns/bitly>

Bit.ly Entities

[**The API**](https://www.hellointerview.com/learn/system-design/in-a-hurry/delivery#4-api-or-system-interface)

The next step in the delivery framework is to define the APIs of the system. This sets up a contract between the client and the server, and it's the first point of reference for the high-level design.

Your goal is to simply go one-by-one through the core requirements and define the APIs that are necessary to satisfy them. Usually, these map 1:1 to the functional requirements, but there are times when multiple endpoints are needed to satisfy an individual functional requirement.

9/10 you'll use a REST API and focus on choosing the right HTTP method or verb to use.

* **POST**: Create a new resource
* **GET**: Read an existing resource
* **PUT**: Update an existing resource
* **DELETE**: Delete an existing resource

System Design <https://www.youtube.com/watch?v=fhdPyoO6aXI&t=1706s>

<https://www.hellointerview.com/learn/system-design/problem-breakdowns/ticketmaster>

**1 – Design Ticket Master**

Approach

A diagram of a software development process

AI-generated content may be incorrect.

Functional Requirements & Non Functional Requirements

A screenshot of a computer screen

AI-generated content may be incorrect.  
Core Entities

A list of events written on a white surface

AI-generated content may be incorrect.

API

A screenshot of a computer

AI-generated content may be incorrect.

**High Level Design**

A screenshot of a diagram

AI-generated content may be incorrect.

**How to get available tickets:**

The reserved tickets are kept in TTL based redis. To Search for available tickets, query the DB in Ticket table with status == available. Get those id’s and get the ticket ids from redis, remove all the tickets from redis which are present from db query output

**Deep Dives**

How to do low latency search

Currently we do full table scan , use elastic search for querying – searching the events, benue, performer ..

When read queries >>> write , use redis for aching the reads and invalidate them when any db changes happening

A screenshot of a diagram

AI-generated content may be incorrect.

The 4 Is for read query