# Vague Prompt:

Crafting a prompt that is too vague or broad can often lead to responses from ChatGPT that are off-topic or not useful for the game. For example, if you ask something like "What happens next?" without providing enough context, the model might produce a response that doesn't fit the narrative. To overcome this, it's essential to provide sufficient context and details in the prompt. This could involve specifying the characters involved, their current situation, their surroundings, or their intended goals.

# ChatGPT Has a Lying Problem

It's hard to believe that we've only been living with ChatGPT for a matter of months. In that time, I'm sure you've heard how everyone's favorite LLM (large language model) will shred the fabric of modern life — by destroying white-collar jobs, making human creativity redundant, and giving teens the tools to abolish homework forever.

But beneath all the hype and occasional silliness, there are some serious problems that LLMs are about to unleash on our world. The most likely scenario is that the information space of the future — web content, comments, social media, and everywhere else that text exists — will be massively polluted by people misusing AI tools.

# Simplify Relational Database + Elasticsearch architecture with TiDB

As a developer, you know that a relational database (RDS) such as MySQL, or PostgreSQL is great for storing structured data and handling online transactional workloads. However, when it comes to handling analytics, and indexing large amounts of data, a different storage engine such as Elasticsearch is often needed. It is a popular choice for online services with hybrid workloads.

Developers need to handle two types of API: SQL and Elasticsearch API. This is not only about different wired protocols, but also different behaviors while processing data. Developers need to build two modules to interact with both sides. This would introduce complexity to the code.

TiDB provides a highly available, horizontally scalable, and fault-tolerant architecture that allows it to handle high write and read loads. It also has built-in support for distributed indexes, which eliminates the need for separate indexing and analytics engines like Elasticsearch. Additionally, it's compatible with the MySQL protocol, making it easy for developers to use and integrate with their existing applications.

# gRPC with microservice

I will talk about the work that we have done to improve communication performance between services, in a project that was developed with microservice architecture by the TDesk team in Trendyol.

TDesk is an application that was located behind an API gateway and has so many domain services that developed different languages. The communication is provided with REST architecture and HTTP way between API gateway and services.

After some investigation to improve communication performance, we decided to make a POC about gRPC and we would like to share our experience about this process.

# Event-Driven Architecture

Event-driven architecture is a popular software design pattern that enables a system to react to various events that occur in the application. In this pattern, there are two types of objects: event emitters and event listeners. Event emitters trigger events, while event listeners listen and react to those events.

Event-driven applications are common for use cases including IoT, fraud detection, payment processing, website monitoring, and real-time marketing. Event-driven applications often treat data as immutable, or unchangeable, making it easy to look up the values of data at previous points in time.

It can handle heavy traffic with a large number of events with low latency. Message broker makes it an ideal choice for applications that need to process a large number of messages in real-time, such as financial trading systems, social media applications, and gaming platforms.

# Caching Strategies to Remember while designing Cache System

The goal of Caching is to reduce the number of times data needs to be fetched from its original source, which can result in faster processing and reduced latency.

Caching can be implemented at various levels of architecture, including in-memory caching, disk caching, database caching, and CDN caching.

Data can be cached using different techniques, each with its own benefits. In-memory caching stores data in the computer's primary memory, enabling fast access compared to disk storage.

Disk caching, on the other hand, stores data on the hard disk, which is slower than primary memory but faster than accessing data from a remote source.

With database caching, frequently accessed data is stored within the database, reducing the need to retrieve data from external storage.

Lastly, CDN caching involves storing data on a distributed network of servers, reducing latency when accessing data from remote locations.

# Saying Goodbye to Ingress: Embracing the Future of Kubernetes Traffic Management with Gateway API and Cilium

In the dynamic world of Kubernetes, managing ingress traffic efficiently and securely has always been a crucial challenge.

However, a groundbreaking solution has emerged, promising to transform the way we handle traffic routing within Kubernetes clusters. Meet the Gateway API: a long-term replacement for Kubernetes Ingress that brings a wealth of benefits to operators, including role-based access control, portability, and extensibility.

Gateway API is an open-source project managed by the SIG-NETWORK community. It is a collection of resources that model service networking in Kubernetes. These resources — GatewayClass, Gateway, HTTPRoute, TCPRoute, Service, etc - aim to evolve Kubernetes service networking through expressive, extensible, and role-oriented interfaces that are implemented by many vendors and have broad industry support.