Facebook Graph API:

API Reference: Facebook Graph API Documentation
Understanding: The Facebook Graph API allows developers to access and manipulate data on Facebook. It provides endpoints to retrieve user information, posts, photos, videos, pages, groups, and more.
Parameters + Endpoints: The API uses HTTP GET requests to access various endpoints. For example, you can use /me to get current user's data, /user-id/posts to retrieve a user's posts, or /page-id/photos to fetch photos from a Facebook page.
Business Utilities: The Facebook Graph API can be used to integrate Facebook functionality into your applications, such as fetching user data, posting content, interacting with groups and pages, and analyzing user engagement for marketing purposes.

LinkedIn API:

API Reference: LinkedIn API Documentation
Understanding: The LinkedIn API provides endpoints for accessing professional networking data. It allows you to retrieve user profiles, connections, companies, jobs, and more.
Parameters + Endpoints: Similar to other REST APIs, you can make HTTP requests to various endpoints. For instance, you can use /v2/me to get the authenticated user's profile, /v2/people/{personId} to retrieve a specific person's profile, or /v2/companies/{companyId} to fetch company details.
Business Utilities: The LinkedIn API can be used to integrate LinkedIn features into your applications, such as displaying professional profiles, sharing job listings, fetching company data, and networking analysis.

2. `DB Schema`

A database schema is a logical blueprint that defines the structure, organization, relationships, and constraints of a database. It serves as a high-level representation of how data is stored and accessed within a database management system (DBMS). A schema includes details about tables, columns, data types, keys, indexes, and relationships between different elements.

Types of Database Schemas:

1. **Physical Schema:**
    - The physical schema describes how data is physically stored on storage devices, such as hard drives or SSDs.
    - It includes details about file organization, indexing methods, and data storage formats.
    - This schema is of particular concern for database administrators and storage management.

2. LOGICAL SCHEMA

- The logical schema defines the structure of the data, including tables, columns, relationships, and constraints.
- It represents the database's overall logical design, without concerning itself with how data is physically stored.
- This schema is of interest to developers, analysts, and designers who work with the data model.

3. Conceptual Schema:

- The conceptual schema provides a high-level, abstract representation of the entire database.
- It focuses on the overall view of the database and its major entities, relationships, and constraints.
- This schema is useful for understanding the database from a business or user perspective.

4. External Schema (View):

- An external schema defines how specific users or applications view the data.
- It provides a subset of the database that is tailored to the needs of a particular user group.
- External schemas allow for data access control and customization for different user roles.

3. **`What is High Availability Architecture? what happens in High Availability Architecture`**

High Availability (HA) architecture refers to a design approach in computing and systems engineering that aims to ensure the continuous and reliable operation of a system, application, or service. The goal of high availability architecture is to minimize downtime and provide seamless access to resources, even in the face of hardware failures, software issues, or other disruptions.

In a high availability architecture, the system is designed to handle failures and maintain service availability by employing various strategies and technologies. The exact implementation can vary based on the specific requirements of the application or service, but here are some common features and concepts:

Redundancy: High availability systems often incorporate redundancy by duplicating critical components. This can include redundant servers, storage devices, network links, and power supplies. If one component fails, the redundant component can take over seamlessly to ensure continuous operation.

Failover: Failover is the process of switching from a failed component to a redundant one without disrupting service. When a failure is detected, the system automatically redirects traffic or operations to the backup component. This can be done at various levels, such as server-level failover or application-level failover.

1.Load Balancing: Load balancers distribute incoming network traffic across multiple servers or resources. This not only improves performance by evenly distributing the load but also enhances availability. If one server becomes unavailable, the load balancer can route traffic to other healthy servers.

Automatic Recovery: High availability systems often have mechanisms in place to automatically recover from failures. This can involve restarting failed services, reestablishing database connections, or restoring lost data from backups.

2.Data Replication: Data replication involves maintaining copies of data across multiple locations or servers. In case of a failure, the system can switch to a replica and continue operations without data loss.

Geographic Redundancy: Some high availability architectures use geographic redundancy by spreading resources across different physical locations or data centers. This approach helps mitigate the impact of regional failures or disasters.

3.Monitoring and Alerts: Continuous monitoring and proactive alerts are crucial for detecting and responding to potential failures. Monitoring tools can identify performance degradation or hardware issues, allowing administrators to take action before a complete failure occurs.

Scalability: High availability architectures are often designed to be scalable, meaning they can handle increased workloads by adding resources or scaling out horizontally.

4.Isolation: Isolating different components or services can prevent a failure in one part of the system from affecting others. This can be achieved through containerization, microservices architecture, or virtualization.

Testing and Simulation: Regular testing and simulation of failures are important to ensure that the high availability mechanisms work as expected. This can involve planned downtime for maintenance and failover testing.

High availability architectures are essential for critical systems where downtime can result in significant financial losses, reputation damage, or safety risks. They are commonly used in applications such as e-commerce websites, financial services, healthcare systems, and telecommunications networks. By ensuring continuous access to resources and services, high availability architectures contribute to a more reliable and resilient computing environment.

4. QUEUE ENDPOINTS AND TOPIC ENDPOINTS

Queue endpoints and topic endpoints are concepts that belong to the realm of publish-subscribe messaging patterns, which is a type of messaging system. Specifically, they are associated with the Publish-Subscribe messaging pattern.

Publish-Subscribe (Pub-Sub) is a messaging pattern where senders (publishers) of messages do not need to know the receivers (subscribers) of the messages. Instead, messages are published to "topics," and subscribers who are interested in specific topics receive the messages.

In the context of Pub-Sub:

- Queue Endpoint: In some messaging systems, particularly those that support a "fan-out" mechanism, a queue endpoint represents a point of delivery for messages from a topic. Each subscriber has its own queue, and when a message is published to a topic, it is replicated to the queues of all subscribers interested in that topic. Each subscriber processes messages independently from its own queue. This provides a form of load balancing and enables parallel processing for subscribers.
- Topic Endpoint: A topic endpoint represents a channel or category to which messages are published. Subscribers interested in a specific topic can subscribe to the topic endpoint. When a message is published to a topic, it is delivered to all active subscribers of that topic.