# API Notes

**API** stands for **Application Programming Interface**. It's a **set of rules or protocols** that allow different software applications to **communicate** with each other.

## ✅ Why is API Important?

APIs are important because they:

### 1. **Enable Software Integration**

* APIs let different systems and apps work together (e.g., Google Maps inside a food delivery app).

### 2. **Speed Up Development**

* Developers don’t need to build everything from scratch—they can use APIs to tap into existing services (like payment gateways or weather data).

### 3. **Enable Automation**

* APIs allow programs to talk to each other without human intervention—great for automation and scaling.

### 4. **Improve Security**

* APIs can control data access, acting as a gatekeeper between your app and the outside world (with authentication and rate limits).

### 5. **Power the Web & Mobile Apps**

* Nearly every mobile app or website uses APIs to fetch or send data (e.g., login, show a product list, post a comment).

### 📦 Real-World Examples

* **Twitter API**: Used to post tweets automatically or display feeds in another app.
* **Spotify API**: Lets developers build apps that fetch playlists, play music, or analyze listening habits.

## 🧩 **Types of APIs**

APIs come in several types based on how and where they’re used:

### 1. **By Accessibility**

* **Open/Public APIs**: Available to external developers (e.g., Twitter API, Weather API)
* **Partner APIs**: Shared with specific partners under agreements (e.g., payment processors)
* **Private/Internal APIs**: Used only within an organization (e.g., microservices in a company)

**By Architecture**:

| **API Type** | **Description** | **Use Case Example** |
| --- | --- | --- |
|  |  |  |
| **REST** | Uses HTTP, stateless, resource-based (GET, POST, etc.) | Most web/mobile apps |
| **SOAP** | XML-based, strict protocols, used in enterprise systems | Banking, legacy systems |
| **GraphQL** | Query language for APIs, fetch exactly what you need | Facebook, Shopify |
| **gRPC** | High-performance, binary protocol (uses Protocol Buffers) | Microservices, internal APIs |
| **WebSockets** | Persistent connection, real-time data transfer | Chat apps, live dashboards |
| **Webhooks** | Event-driven callbacks (push, not poll) | GitHub notifying Jenkins on push |
| **Falcor** (Netflix) | Like GraphQL but with a focus on a virtual JSON graph | Netflix services |

## 🛠️ **Modes of API Usage**

These define how APIs are used by applications:

### 1. **Synchronous APIs**

* The client waits for the response.
* Examples: REST, GraphQL

### 2. **Asynchronous APIs**

* The request is sent, and the response comes later (non-blocking).
* Examples: Webhooks, WebSockets, message queues (Kafka, RabbitMQ)

| **Aspect** | **Example / Tool** | **Use Case** |
| --- | --- | --- |
| API Design | OpenAPI, Swagger | Standardized API specs |
| Communication | REST, GraphQL, gRPC | Varies by architecture |
| Auth | OAuth 2.0, JWT | Secure access |
| Management | Kong, Apigee | Monitoring, scaling |
| Testing | Postman, Pact | Contract and integration testing |
| Deployment | Docker, Kubernetes, AWS Gateway | Scaling APIs in production |

## 📚 Summary Table