1. What is API? Give 3 real-life examples of usage of API.

API:

API means Application Programming Interface, which is **a software intermediary that allows two applications to interact with each other**. Each time you use an app like google map, send an instant message, or check the weather on your phone, you're using an API.

3 Real-life examples:

a) Whenever we are using a web browser to access local weather, we enter "what is today's weather?". The web browser collects data about weather from different weather reports providing sites and displays us options of choices. That is an example of using an API.
b) If you want to book a flight, you submit a query on your web browser saying "flights from NY to Florida". The web browser will act as a client and will collect desired information or data from various air ticket providing sites. Then it will display an interface to choose desired flight details. This interface is an API.
c) Last year while we went to vacation in poconos.we had no idea about the areaand where we can go to eat.we just ask to google map API to find a halal restaurant near us and using the API easily we found couple of best restaurants in pocons.Thanks to google API, they are able to easily display business hours, reviews, phone numbers, and even times they are likely to be busy.

2. What is API testing and name some types of API testing?

API Testing:

API TESTING is a software testing type that validates Application Programming Interfaces (APIs). The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces. In API Testing, instead of using standard user inputs(keyboard) and outputs, you use software to send calls to the API, get output, and note down the system's response. API tests are very different from GUI Tests and won't concentrate on the look and feel of an application. It mainly concentrates on the business logic layer of the software architecture

Name some types of API testing

1. Validation testing
2. Functional testing
3. UI testing
4. Security testing

3. What is the difference between REST and SOAP APIs?

REST and **SOAP** are 2 different approaches to online data transmission. Specifically, both define how to build application programming interfaces (APIs), which allow data to be communicated

between web applications. ***Representational state transfer (REST)*** is a set of architectural principles. ***Simple object access protocol (SOAP)*** is an official protocol maintained by the World Wide Web Consortium (W3C). The main difference is that SOAP is a protocol while REST is not. Typically, an API will adhere to either REST or SOAP, depending on the use case and preferences of the developer.

Many legacy systems may still adhere to SOAP, while REST came later and is often viewed as a faster alternative in web-based scenarios. REST is a set of guidelines that offers flexible implementation, whereas SOAP is a protocol with specific requirements like XML messaging.

REST APIs are lightweight, making them ideal for newer contexts like the Internet of Things (IoT), mobile application development, and serverless computing. SOAP web services offer built-in security and transaction compliance that align with many enterprise needs, but that also makes them heavier. Additionally, many public APIs, like the Google Maps API, follow the REST guidelines.

4. What are common HTTP methods? Explain with examples.

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

HEAD

The HEAD method asks for a response identical to a GET request, but without the response body.

POST

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

PUT

The PUT method replaces all current representations of the target resource with the request payload.

DELETE

The DELETE method deletes the specified resource.

Example of a method:

https://go.postman.co/workspace/New-Team-Workspace~38b295e5-e8ff-4088-bd2e-368f2dc5a7e3/collection/20615829-0e87e85f-3071-485b-8ce7-681c0a6e3b04?action=share&creator=20615829

5. What is the difference between POST AND put?

PUT	POST
This method is idempotent.	This method is not idempotent.
PUT method is called when you have to modify a single resource, which is already a part of resource collection.	POST method is called when you have to add a child resource under resources collection.
RFC-2616 depicts that the PUT method sends a request for an enclosed entity stored in the supplied request URI.	This method requests the server to accept the entity which is enclosed in the request.
PUT method syntax is PUT /questions/{question-id}	POST method syntax is POST /questions
PUT method answer can be cached.	You cannot cache PUT method responses.
PUT /vi/juice/orders/1234 indicates that you are updating a resource which is identified by “1234”.	POST /vi/juice/orders indicates that you are creating a new resource and return an identifier to describe the resource.
If you send the same request multiple times, the result will remain the same.	If you send the same POST request more than one time, you will receive different results.
PUT works as specific.	POST work as abstract.
We use UPDATE query in PUT.	We use create query in POST.

Explain with example

post example

6. Name 5 main categories of HTTP status codes

1. Informational responses (100–199)
2. Successful responses (200–299)
3. Redirection messages (300–399)
4. Client error responses (400–499)
5. Server error responses (500–599)

Explain 5 status codes from each category

1)101 Switching Protocols

This code is sent in response to an Upgrade request header from the client and indicates the protocol the server is switching to.

2)201 Created

The request succeeded, and a new resource was created as a result. This is typically the response sent after POST requests, or some PUT requests.

3)301 Moved Permanently

The URL of the requested resource has been changed permanently. The new URL is given in the response.

4)404 Not Found

The server can not find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of 403 Forbidden to hide the existence of a resource from an unauthorized client. This response code is probably the most well known due to its frequent occurrence on the web.

5)502 Bad Gateway

This error response means that the server, while working as a gateway to get a response needed to handle the request, got an invalid response.

7. What are the main components of HTTP request and HTTP response?

HTTP request is divided into 3 parts

1. A request line
2. A set of header fields
3. A body, which is optional

HTTP response components

- Status Line
- Headers
- Body (Optional)

8. What are the main components of the HTTP request header and response header?

Components of HTTP request header

. Host
. Content-Type
. Cache-Control.

Components of response header

- `Accept-Patch`
- `Accept-Ranges`
- `Age`

- ***Allow***
- ***Alt-Svc***
- ***Cache-Control***
- ***Connection***
- ***Content-Disposition***
- Content-Encoding
- Content-Language
- Content-Length
- Content-Location
- Content-Range
- Content-Type
- Date
- Delta-Base
- ETag
- Expires
- IM
- Last-Modified
- Link
- Location
- Pragma
- Proxy-Authenticate
- Public-Key-Pins
- Retry-After
- Server

9. What is the difference between authentication and authorization?

Authentication	Authorization
Authentication validate who the user is.	Authorization determines what resources a user can access.
Authentication works through passwords, one-time pins, biometric information, and other information provided or entered by the user.	Authorization works through settings that are implemented and maintained by the organization.
Authentication is the first step of a good identity and access management process.	Authorization always takes place after authentication.
Authentication is visible to and partially changeable by the user.	Authorization isn't visible to or changeable by the user.
Example: By verifying their identity, employees can gain access to an HR application that includes their personal pay information, vacation time, and 401K data.	Example: Once their level of access is authorized, employees and HR managers can access different levels of data based on the permissions set by the organization.

10. Briefly explain common API Authentication Methods.

Common API authentication methods

While there are dozens of open and proprietary API authentication methods available, there are four common methods most administrators will see and interact with over the course of their professional career.

1. HTTP basic authentication

If a simple form of HTTP authentication is all an app or service requires, HTTP basic authentication might be a good fit. It uses a locally derived username and password and relies on Base64 encoding. Authentication uses the HTTP header, making it easy to integrate. Because this method uses shared credentials, however, it's important to rotate passwords on a regular basis.

2. API access tokens

API keys have unique identifiers for each user and for every time they attempt to authenticate. Access tokens are suitable for applications where many users require access. Access tokens are secure and easy to work with from an end-user perspective.

3. OAuth with OpenID

Despite having *auth* in the name, OAuth is not an authentication mechanism. Instead, it provides authorization services to determine which users have access to various corporate resources. OAuth is used alongside OpenID, an authentication mechanism. Using OAuth and OpenID together provides authentication and authorization. With OAuth 2.0, OpenID can authenticate users and devices using a third-party authentication system. This combination is considered one of the more secure authentication/authorization options available today.

4. SAML federated identity

Security Assertion Markup Language (SAML) is another tokenlike authentication method often used in environments that have federated single sign-on (SSO) implemented. This XML-derived open standard framework helps seamlessly authenticate users through the organization's respective identity provider. For larger organizations working to consolidate the number of authentication mechanisms within the company, the use of SAML and federated SSO is a great fit.