HTTP (Hypertext Transfer Protocol) is like a language that your web browser and websites use to talk to each other. It's a set of rules that helps them understand how to request and exchange information. There are several different methods in HTTP, and each one serves a specific purpose:

GET: Think of it like asking for a page from a book. When you type a web address into your browser and hit "Enter," your browser sends a GET request to the website's server. The server then sends back the web page you requested.

POST: This is like sending a letter. When you submit a form on a website, like when you create a user account or make an online purchase, your browser sends a POST request with the information you entered. The server receives this data and processes it, like storing your account details.

PUT: Imagine you have a blank piece of paper, and you want to replace the text on it. PUT is like saying, "Here's the new content for this page." It's often used to update existing information on a website.

DELETE: It's as simple as it sounds. DELETE tells the server to remove something, like a file or a record in a database. It's like telling the server, "Get rid of this."

PATCH: This method is like making a small fix or update to something. Instead of replacing the entire page (as with PUT), you're saying, "Change this specific part of the page."

HEAD: This is similar to a GET request, but with one key difference: it asks for the response headers only, not the actual content. It's like asking for the title of a book without getting the whole story.

OPTIONS: Think of OPTIONS as asking, "What can I do here?" When you send an OPTIONS request, the server tells you which HTTP methods are allowed for a particular resource. It's like checking the rules of a game before you start playing.

// just like connecting to a server CONNECT: This method is mainly used for establishing a network connection to a resource, typically for secure communication through a proxy.

TRACE: TRACE is a diagnostic method that echoes back the received request, which can be useful for debugging and understanding how a request is being processed by intermediaries.

These methods help web browsers and servers understand what kind of action to take when communicating with each other. It's like knowing whether to ask for information (GET), send data (POST), update something (PUT), remove something (DELETE), or perform other specific actions. The right method is chosen based on what you want to do with the web server or website you're interacting with.

Breif Documentation of concept looked out for the development of this project

Imagine Postman as a magical tool for computers that helps them talk to each other.

You know how people use letters or phones to send messages to each other? Well, computers also need a way to talk to each other, especially when they're on the internet. Postman is like a special messenger for computers, and it helps them send messages and understand each other better.

Here's how it works:

Making Requests (Sending Messages): Just like when you write a letter or send a text message, you have to tell Postman what you want to say. In the computer world, this is called a "request." You can tell Postman, "Hey, I want to talk to this website or computer, and I want to ask it for some information." Postman helps you write down this message.

Choosing How to Talk: Computers have different ways of talking to each other, kind of like speaking different languages. Postman helps you pick the right way to talk to the other computer. This is called choosing the "HTTP method," like saying "I want to ask for information" or "I want to send some data."

Adding Details: Sometimes, you need to include more information in your message, like your name and address when sending a letter. Postman lets you add extra details, like the website you want to talk to and any special instructions.

Sending the Message: Once you've filled out your message with Postman, you press a button, and Postman sends your message to the other computer over the internet. It's like dropping your letter in the mailbox or hitting the send button on your text message.

Receiving Responses: After the other computer gets your message, it sends a reply back. Postman helps you read this reply, so you can understand what the other computer is saying. It's like getting a letter or a reply to your text message.

Checking for Errors: Sometimes, things can go wrong, just like when you make a mistake in a letter. Postman helps you check if there are any mistakes in your message or if the other computer didn't understand you. It gives you hints on how to fix things.

Testing and Learning: Postman is also like a learning tool. It helps you practice talking to different computers and see how they respond. This way, you can learn more about how computers work together on the internet.

So, in simple terms, Postman is like a helpful friend that makes it easy for you to send messages to other computers on the internet. It helps you write the messages, send them, understand the replies, and learn more about how computers communicate with each other. It's a great tool for people who want to build, test, and work with computer programs and websites.

// This another research I got from chat gpt to fully understand the concept of API in a simplified manner so to give me more deeper understanding of what am working on.

Imagine APIs as a menu at a restaurant:

You know how when you go to a restaurant, you don't go into the kitchen and cook your own food, right? Instead, you look at a menu, which shows you all the dishes the restaurant can make. The menu tells you what choices you have.

APIs are a bit like that menu. They're like a list of things a computer can do or information it can share. So, when you use a computer program or an app, it might want to do things like get weather data, show you pictures, or send messages. Just like you choose a dish from a menu, the program uses an API to ask the computer for these things.

Here's how it works:

Ordering from the Menu (Request): When you order food from the menu, you're telling the restaurant what you want. In the computer world, this is like making a request. The program says, "Hey computer, I want some information or I want to do something."

Kitchen Makes the Food (Server): In a restaurant, the chefs in the kitchen make your food based on your order. In the computer world, there's a special computer called a server that takes the request from the program and does the job. It might gather data, like weather information or pictures.

Food Arrives at Your Table (Response): Just like the server brings your food to your table, the server computer sends back the information or action that the program asked for. This is the response. It's like the server saying, "Here's the information you wanted."

Enjoying Your Meal (Using the Data): You can now enjoy your meal at the restaurant. In the computer world, the program can use the data it got from the server to do what it needs to do. For example, it can show you the weather or display pictures on your screen.

So, APIs are like the menu that lets computer programs order and get things done without having to do all the work themselves. They're a way for different computer programs to talk to each other and share information, just like you talk to the restaurant through the menu to get the food you want.

More pratical

What is an API?

An API is like a special set of rules and tools that allow different computer programs to talk to each other and share information. It's a bit like a menu at a restaurant where you can order dishes. In this case, computer programs "order" data or services from each other using APIs.

How APIs are Created:

Creating an API involves a few steps:

Decide What to Share: Just like a restaurant menu lists the dishes it can serve, you decide what information or functions your program can share with others. For example, you might want to share weather data, pictures, or user accounts.

Design the API: This is like designing the menu. You create a list of things (like dishes) that your API can do. Each item on the list is called an "endpoint." Each endpoint is like a button or option that other programs can choose to use.

Define How to Use It: For each endpoint on your API menu, you specify how other programs can request that information or service. You define the rules, like what they need to ask for and how they should ask for it.

Build the API: This is where you do the technical work. You write the code that makes your API work according to the rules you defined. It's like creating the recipes and cooking the dishes in a restaurant.

Document It: Just like a menu has descriptions of dishes, your API needs documentation. This is like writing down instructions for how other programs should use your API. It helps them understand how to talk to your program.

Test It: Before you let others use your API, you need to make sure it works well. You test it to ensure that it provides the right data or services and that it follows the rules you set.

Share It: Once your API is ready and tested, you share it with other developers. They can now use your API to access the data or services your program offers.

Maintain and Update: Over time, you might need to make changes or improvements to your API. Just like a restaurant updates its menu or recipes, you'll need to maintain and update your API to keep it useful and reliable.

So, in simple terms, creating an API is like making a menu of things your program can do, writing down how others can order from that menu, and then building the kitchen (the technical part) to make those orders happen. It's a way for different computer programs to work together and share information or services.

What is a REST API?

A REST API is like a friendly waiter in a restaurant who listens to your orders, brings you food, and understands a special language that makes ordering easy.

How a REST API Works:

Imagine you're at a restaurant:

Menu (Resources): The restaurant has a menu with different dishes you can order. In the world of REST APIs, these dishes are like pieces of information or actions that you can request from a computer.

Order (HTTP Request): When you're ready to eat, you tell the waiter what dish you want. This is similar to sending a request to a REST API. You use a special language called HTTP (Hypertext Transfer Protocol) to make your request.

Kitchen (Server): The waiter takes your order to the kitchen, where the chef prepares your dish. In the world of REST APIs, the "kitchen" is a computer called a server that has the information or action you asked for.

Plate of Food (HTTP Response): The chef finishes making your dish and gives it to the waiter. The waiter brings your food back to your table. Similarly, the server sends back the information or action you requested as an HTTP response.

Enjoy Your Meal (Use the Data): You can now enjoy your delicious meal. In the world of REST APIs, the program that made the request can use the data from the response for whatever it needs to do, like displaying information on a website or saving it in a database.

How REST APIs are Created:

Creating a REST API involves these steps:

Design the Menu (Define Resources): You decide what kinds of information or actions your API will offer. These are like the dishes on the menu. For example, you might have "get weather" or "create user account."

Create Ordering System (Define Endpoints): You specify how others can order these items. Each item on the menu corresponds to an "endpoint," which is like a button that other programs can press to request something from your API.

Chef's Recipe (Server Code): You write the code on the server side that knows how to make each dish (fulfill each request). This code listens for incoming orders and prepares the responses.

Menu Descriptions (Documentation): You create a menu guide (documentation) that explains what each item on your API menu does and how others can order from it. This helps other developers understand how to use your API.

Test the Kitchen (Testing): Before opening the restaurant to customers, you make sure that your kitchen (server) works correctly. You test to ensure it can handle orders and provide the right dishes.

Open for Business (Launch): Once everything is ready, you share your API with other developers. They can now make orders (requests) and get the data or actions they need from your API.

Continuous Improvement (Maintenance): Over time, you may update your menu, improve recipes, and fix any issues to keep your API useful and reliable.

So, a REST API is like a restaurant where you can order data or actions using a special language (HTTP). It's created by designing the menu (defining resources), setting up an ordering system (defining endpoints), and making sure the kitchen (server) can prepare and serve the dishes (responses) to those who order from it.