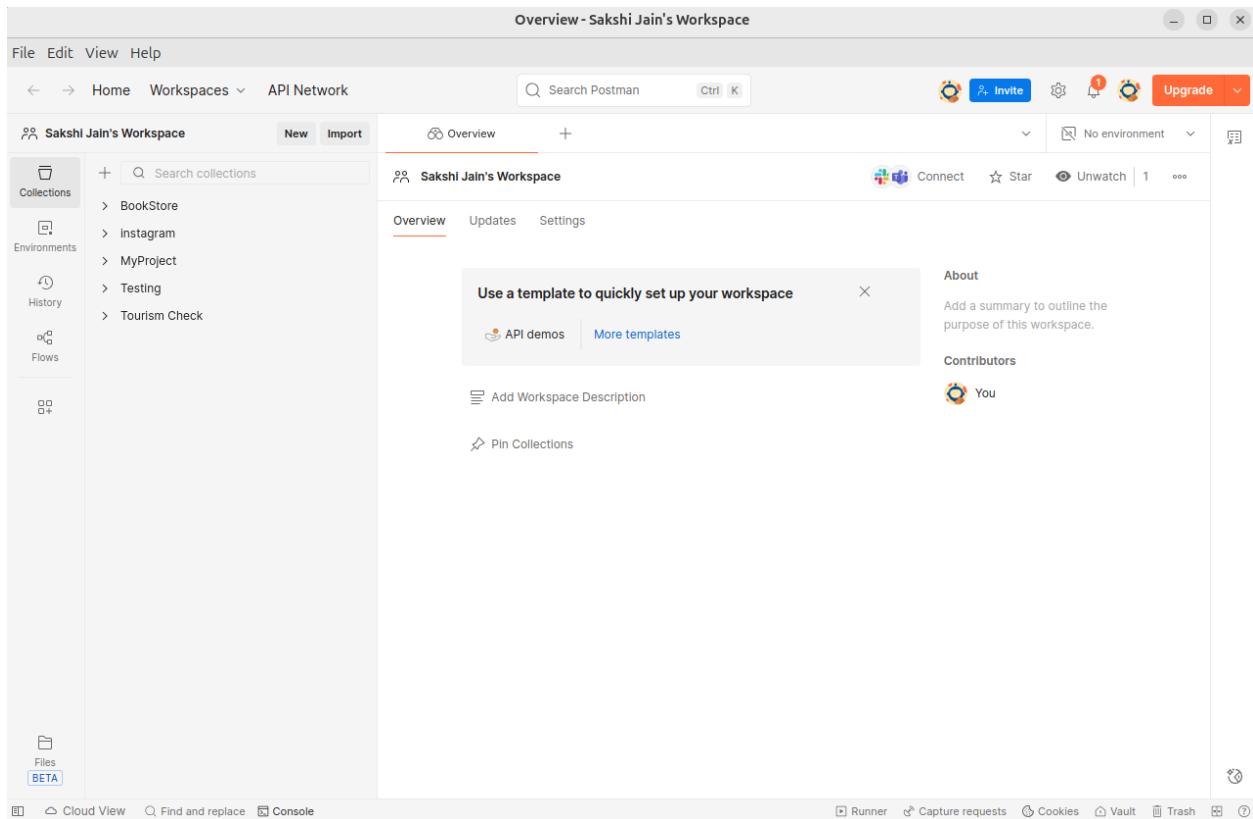


Task 1- Install and open Postman (or use an alternative like Hoppscotch).

- Will be using Postman



Task -2] Make a GET request to a public API

(e.g., <https://jsonplaceholder.typicode.com/posts/1>).

- I have researched about public apis on website - <https://apipheny.io/free-api/>
- Selected the public REST API for get request : Predicting the age based on name
- API URL: <https://api.agify.io?name=sakshi>
- Output:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (BookStore, Instagram, MyProject, Testing, Tourism Check), 'Environments', 'History', and 'Flows'. The main area has tabs for 'Overview' and 'TRaining-Session'. Under 'TRaining-Session', it shows a 'GET' request to 'https://api.agify.io?name=sakshi'. The 'Params' tab is selected, showing 'name' with value 'sakshi'. Below that is a table for 'Query Params' with one row for 'Key' and 'Value'. The 'Body' tab shows the response: a 200 OK status with a response time of 3.77s and a size of 605 B. The response body is a JSON object:

```

1  {
2   "count": 287,
3   "name": "sakshi",
4   "age": 19
5 }

```

At the bottom, there are buttons for 'Runner', 'Capture requests', 'Cookies', 'Vault', 'Trash', and a help icon.

Now trying GraphQL Free public api:

I used : <https://countries.trevorblades.com/>

And Query Schema:

```

query {
  countries {
    code
    name
    capital
    emoji
  }
}

```

The screenshot shows the Postman application interface with the title "Training-GraphQL - Sakshi Jain's Workspace". The left sidebar displays collections like "BookStore", "Instagram", "MyProject", "Testing", and "Tourism Check". The main workspace is titled "Training-GraphQL" and contains a query for "countries.trevorblades.com". The query is:

```

query Countries {
  countries {
    capital
    code
    emoji
    currency
    name
  }
}

```

The results section shows a table with the following data:

	capital	code	emoji	currency	name
0	Andorra la Vella	AD	🇪🇸	EUR	Andorra
1	Abu Dhabi	AE	🇦🇪	AED	United Arab Emirates
2	Kabul	AF	🇦🇫	AFN	Afghanistan
3	Saint John's	AG	🇦🇬	XCD	Antigua and Barbuda
4	The Valley	AI	🇦🇮	XCD	Anguilla
5	Tirana	AL	🇦🇱	ALL	Albania

The status bar at the bottom indicates "200 OK" with a response time of "606.91 ms" and a size of "9.82 KB".

Again I checked - Rick and Morty API
URL: <https://rickandmortyapi.com/graphql>

SCHEMA

```

query {
  characters(page: 1) {
    results {
      name
      species
      status
    }
  }
}

```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Sakshi Jain's Workspace' containing collections like BookStore, Instagram, MyProject, Testing, and Tourism Check. The main area has tabs for Overview, GET TTraining-Session, Training-GraphQL, and the current tab, https://rickandmortyapi.com/graphql. The URL bar shows 'https://rickandmortyapi.com/graphql'. The query editor contains the following GraphQL code:

```

query Characters {
  characters {
    results {
      name
      status
      species
    }
  }
}

```

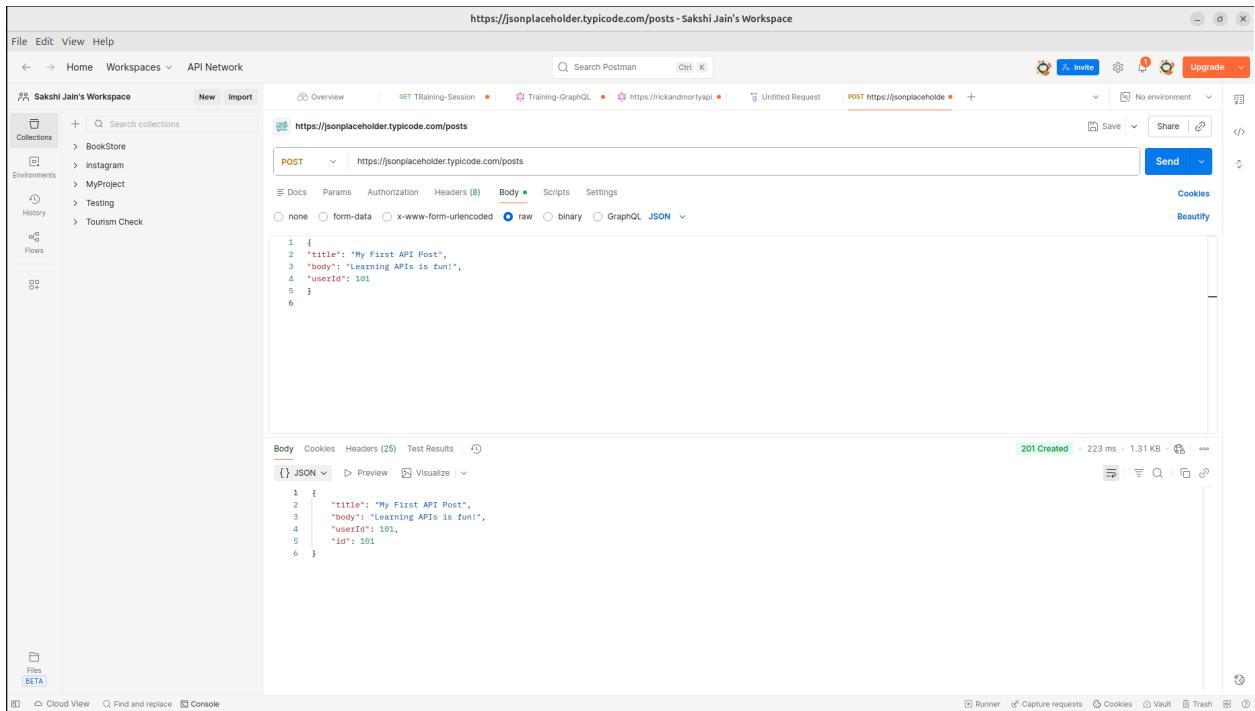
Below the code, under the 'Body' tab, is a table showing the results of the query:

	name	status	species
0	Rick Sanchez	Alive	Human
1	Morty Smith	Alive	Human
2	Summer Smith	Alive	Human
3	Beth Smith	Alive	Human
4	Jerry Smith	Alive	Human
5	Abadango Cluster Princess	Alive	Alien

The status bar at the bottom indicates a 200 OK response with 1723.63 ms and 1.26 KB.

Task 3: Make a POST request to <https://jsonplaceholder.typicode.com/posts> with a JSON body:

```
{
  "title": "My First API Post",
  "body": "Learning APIs is fun!",
  "userId": 101
}
```



Task 4: In your browser DevTools Network tab, trigger a request to any website (e.g., your own site or google.com) and inspect:

- URL
- Method
- Request headers
- Response headers
- Body (if any)

Write a short explanation (5–8 lines) describing what happened from request → response.

When I first loaded <https://bharat-yatra-frontend.vercel.app/>, the browser already had a cached version of the page. So it sent a GET request with cache validation headers (**if-none-match** and **if-modified-since**).

The server responded with 304 Not Modified, indicating the content had not changed and no response body was sent.

After disabling cache and reloading, the browser requested the page again without using cached data.

This time the server returned 200 OK along with the full HTML document. The browser then rendered the page and loaded additional assets like JavaScript and CSS.

What I observed in each part

URL : <https://bharat-yatra-frontend.vercel.app/>

Method : GET

Headers

- 304 request included cache headers like if-none-match and if-modified-since
- 200 response included content-type, content-length, etag, and caching headers

Body

- 304 → No response body (browser reused cached HTML)
- 200 → HTML content was returned in the response body

Basic difference between 304 and 200

- 304 Not Modified → Content unchanged, load from cache, faster, no body
- 200 OK → Fresh content sent by server, full response body included

While observing and analysing I got to know about: if-none-match and if-modified-since

And then i checked about it how this works

if-none-match

This works with something called an ETag (entity tag).

The server previously sent an ETag like:
"d00238250aee6422ae39b18d3e775984"

The browser stores it.

On the next request, the browser sends:
if-none-match: "d00238250aee6422ae39b18d3e775984"

If the current version does not match this tag, then send me the new content.

If the tag matches, the server replies 304 Not Modified.
If it doesn't, the server sends 200 OK with new content.

if-modified-since

This is a **time-based check**.

- The server earlier said:
last-modified: Thu, 08 Jan 2026 07:11:47 GMT
- The browser later sends:
if-modified-since: Wed, 07 Jan 2026 03:49:13 GMT

Only send the page again if it was changed after this time.

If nothing changed since that timestamp → **304**
If it changed → **200** with content

Lesser precise than if-none-match