Activity 07 – Request Method GET using JSON Data Set

Course: SE/COM S 3190 – Construction of User Interfaces
Student: Ash Bhuiyan
ISU NetID: mbhuiyan
Email: mbhuiyan@iastate.edu
Date: November 14, 2025

1. Conceptual Questions

1.1 What is a REST application?

A REST (Representational State Transfer) application is a web application that follows the REST architectural style. Resources (such as users, products, or robots) are identified by URLs, and clients interact with those resources using standard HTTP methods like GET, POST, PUT, and DELETE. Each request is stateless – the server does not remember previous requests, so every request contains all the information needed for the server to understand and process it. The server responds with a representation of the resource, typically in formats such as JSON, HTML, or XML.

1.2 What is an API application?

An API (Application Programming Interface) application exposes a set of endpoints and rules that other software can use to communicate with it. Instead of a human clicking buttons in a graphical interface, another program sends structured requests (for example HTTP requests with JSON bodies) and receives structured responses. This lets different services and applications exchange data and functionality in a consistent, automated way.

2. Testing the HTTP Server with curl (Server Running)

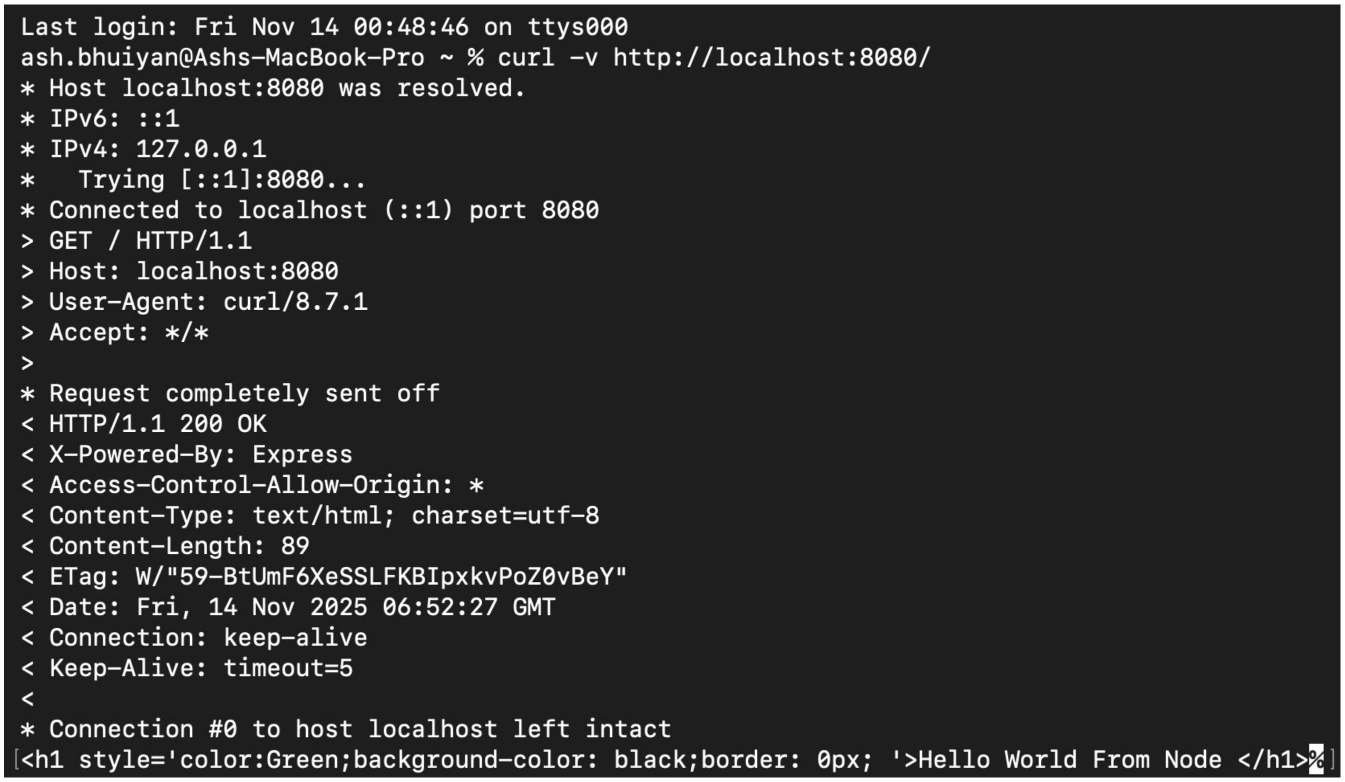
2.1 GET / with curl

Command executed:

```
curl -v http://localhost:8080/
```

This request sends an HTTP GET to the root path (/). The server responds with status 200 OK and the body is an HTML <h1> element that says "Hello World From Node" with green text on a black background. The verbose output also shows headers such as X-Powered-By: Express and Content-Type: text/html.

Screenshot is on the next page



```
Last login: Fri Nov 14 00:48:46 on ttys000
ash.bhuiyan@Ashs-MacBook-Pro ~ % curl -v http://localhost:8080/
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* Connected to localhost (::1) port 8080
> GET / HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Access-Control-Allow-Origin: *
< Content-Type: text/html; charset=utf-8
< Content-Length: 89
< ETag: W/"59-BtUmF6XeSSLFKBIpxkvPoZ0vBeY"
< Date: Fri, 14 Nov 2025 06:52:27 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
* Connection #0 to host localhost left intact
<h1 style='color:Green;background-color: black;border: 0px; '>Hello World From Node </h1>%
```

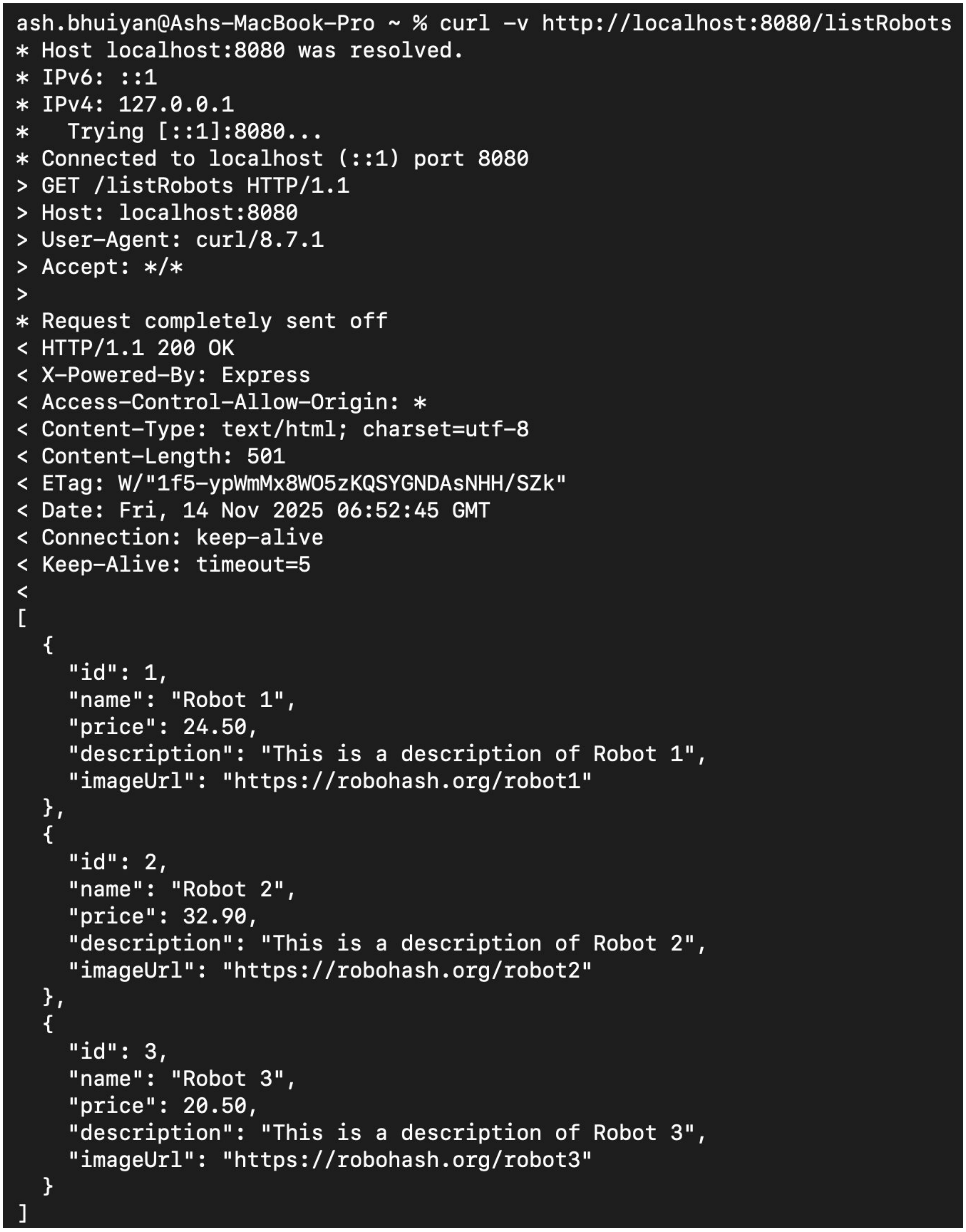
Figure 1: curl -v http://localhost:8080/ (server running)

2.2 GET /listRobots with curl (status 200)

Command executed:
curl -v http://localhost:8080/listRobots

This request calls the /listRobots endpoint, which reads the robots.json file on the server and sends its contents back to the client. The response status is 200 OK and the body is a JSON array with three robot objects (id, name, price, description, imageUrl). The verbose output shows the request line GET /listRobots HTTP/1.1 and response headers including Content-Type and Content-Length.

Screenshot is on the next page



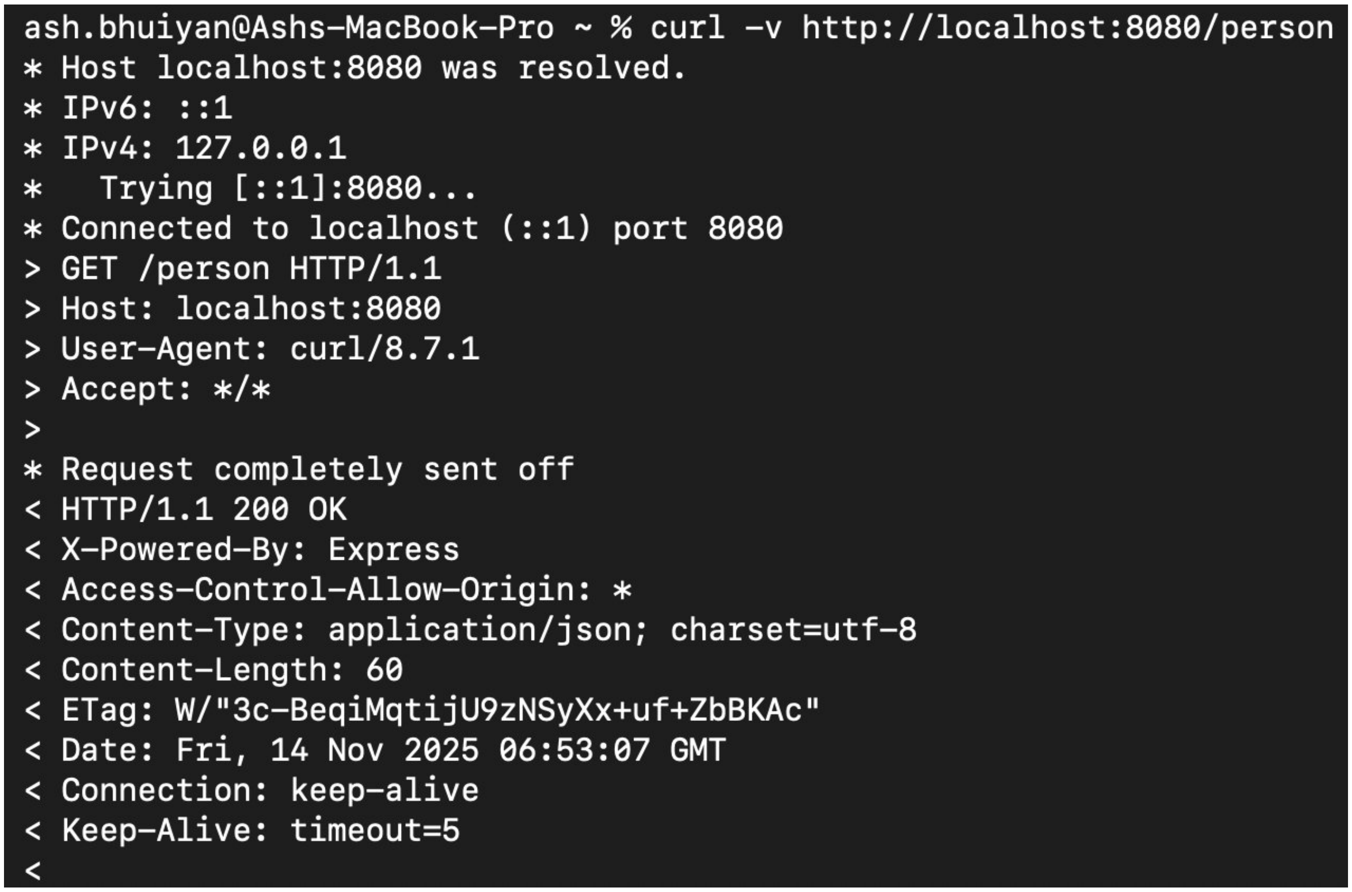
```
ash.bhuiyan@Ashs-MacBook-Pro ~ % curl -v http://localhost:8080/listRobots
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* Connected to localhost (::1) port 8080
> GET /listRobots HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Access-Control-Allow-Origin: *
< Content-Type: text/html; charset=utf-8
< Content-Length: 501
< ETag: W/"1f5-ypWmMx8WO5zKQSYGNDAsNHH/SZk"
< Date: Fri, 14 Nov 2025 06:52:45 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
[
  {
    "id": 1,
    "name": "Robot 1",
    "price": 24.50,
    "description": "This is a description of Robot 1",
    "imageUrl": "https://robohash.org/robot1"
  },
  {
    "id": 2,
    "name": "Robot 2",
    "price": 32.90,
    "description": "This is a description of Robot 2",
    "imageUrl": "https://robohash.org/robot2"
  },
  {
    "id": 3,
    "name": "Robot 3",
    "price": 20.50,
    "description": "This is a description of Robot 3",
    "imageUrl": "https://robohash.org/robot3"
  }
]
```

Figure 2: curl -v http://localhost:8080/listRobots (200 OK)

2.3 GET /person with curl

Command executed:
curl -v http://localhost:8080/person

The /person endpoint creates a JavaScript object with fields name, email, and job and sends it back as JSON. The response status is 200 OK and the Content-Type header is application/json. This shows how a Node/Express API can send structured JSON data to a client instead of HTML.



```
ash.bhuiyan@Ashs-MacBook-Pro ~ % curl -v http://localhost:8080/person
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* Connected to localhost (::1) port 8080
> GET /person HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Access-Control-Allow-Origin: *
< Content-Type: application/json; charset=utf-8
< Content-Length: 60
< ETag: W/"3c-BeqiMqtijU9zNSyXx+uf+ZbBKAc"
< Date: Fri, 14 Nov 2025 06:53:07 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
```

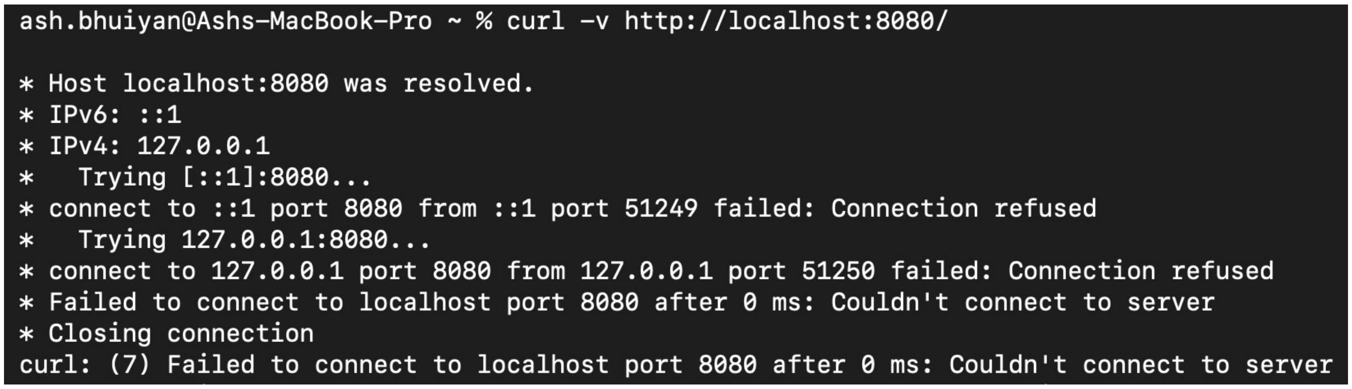
Figure 3: curl -v http://localhost:8080/person (200 OK, JSON)

3. curl Tests with the Server Stopped

3.1 GET / when the server is NOT running

Command executed after stopping the Node.js server:
curl -v http://localhost:8080/

In this case there is no process listening on port 8080. curl first tries to connect via IPv6 (::1) and then IPv4 (127.0.0.1), but both connections are refused by the operating system. Because the HTTP request never reaches a server, there is no HTTP status line; instead, curl prints an error such as "Failed to connect to localhost port 8080: Couldn't connect to server".



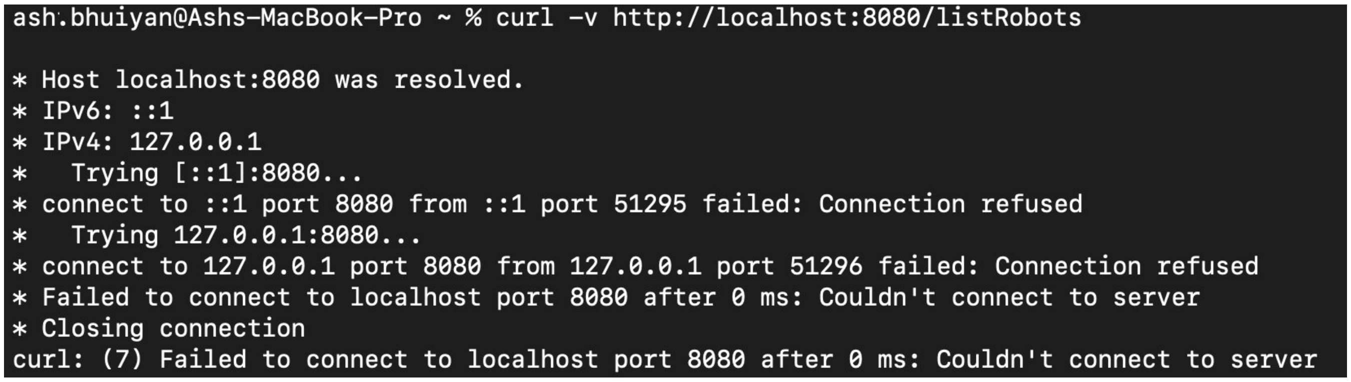
```
ash.bhuiyan@Ashs-MacBook-Pro ~ % curl -v http://localhost:8080/

* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* connect to ::1 port 8080 from ::1 port 51249 failed: Connection refused
*   Trying 127.0.0.1:8080...
* connect to 127.0.0.1 port 8080 from 127.0.0.1 port 51250 failed: Connection refused
* Failed to connect to localhost port 8080 after 0 ms: Couldn't connect to server
* Closing connection
curl: (7) Failed to connect to localhost port 8080 after 0 ms: Couldn't connect to server
```

Figure 4: curl -v http://localhost:8080/ (connection refused)

3.2 GET /listRobots when the server is NOT running

Command executed after stopping the Node.js server:
 curl -v http://localhost:8080/listRobots

The behavior is the same as in 3.1: curl cannot establish a TCP connection to port 8080. The output shows attempts to connect to ::1 and 127.0.0.1, followed by "Connection refused" messages and the same curl error code 7. This demonstrates that HTTP cannot happen at all if there is no listening server.



```
ash.bhuiyan@Ashs-MacBook-Pro ~ % curl -v http://localhost:8080/listRobots

* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* connect to ::1 port 8080 from ::1 port 51295 failed: Connection refused
*   Trying 127.0.0.1:8080...
* connect to 127.0.0.1 port 8080 from 127.0.0.1 port 51296 failed: Connection refused
* Failed to connect to localhost port 8080 after 0 ms: Couldn't connect to server
* Closing connection
curl: (7) Failed to connect to localhost port 8080 after 0 ms: Couldn't connect to server
```

Figure 5: curl -v http://localhost:8080/listRobots (connection refused)

4. Changing /listRobots Status Code from 200 to 404

4.1 curl -v http://localhost:8080/listRobots with status 404

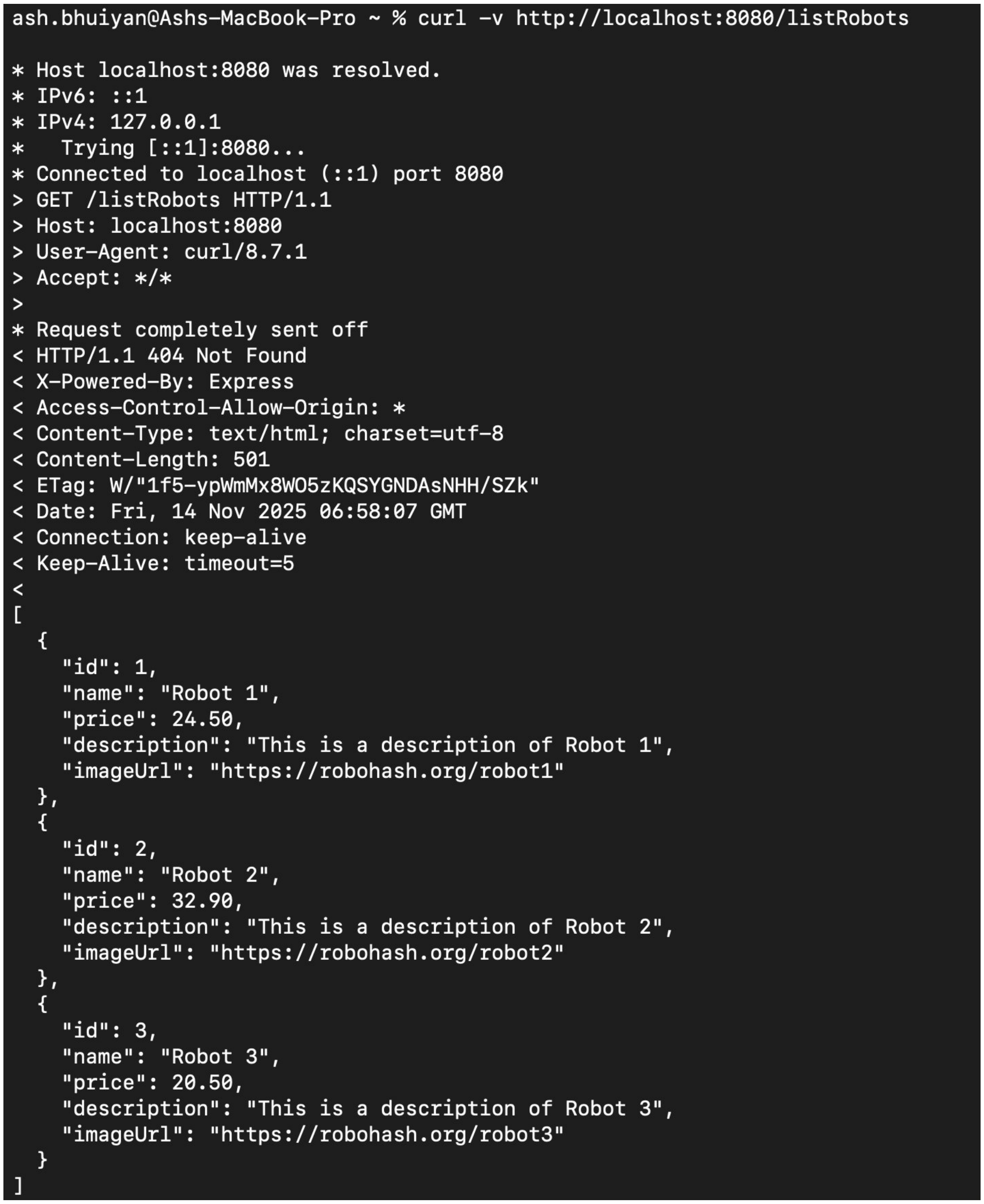
In mbhuiyan_Activity07.js, inside the /listRobots route, I temporarily changed the status code from 200 to 404:

```
res.status(200);   // original
// res.status(404);   // temporary change
```

After changing to status 404 and restarting (or letting nodemon restart) the server, I ran:

    curl -v http://localhost:8080/listRobots

The response now shows the status line HTTP/1.1 404 Not Found, but the body is still the same JSON array from robots.json. This illustrates that the status code is metadata that tells the client how to interpret the result (in this case, as a not-found error), even if a body is still sent.



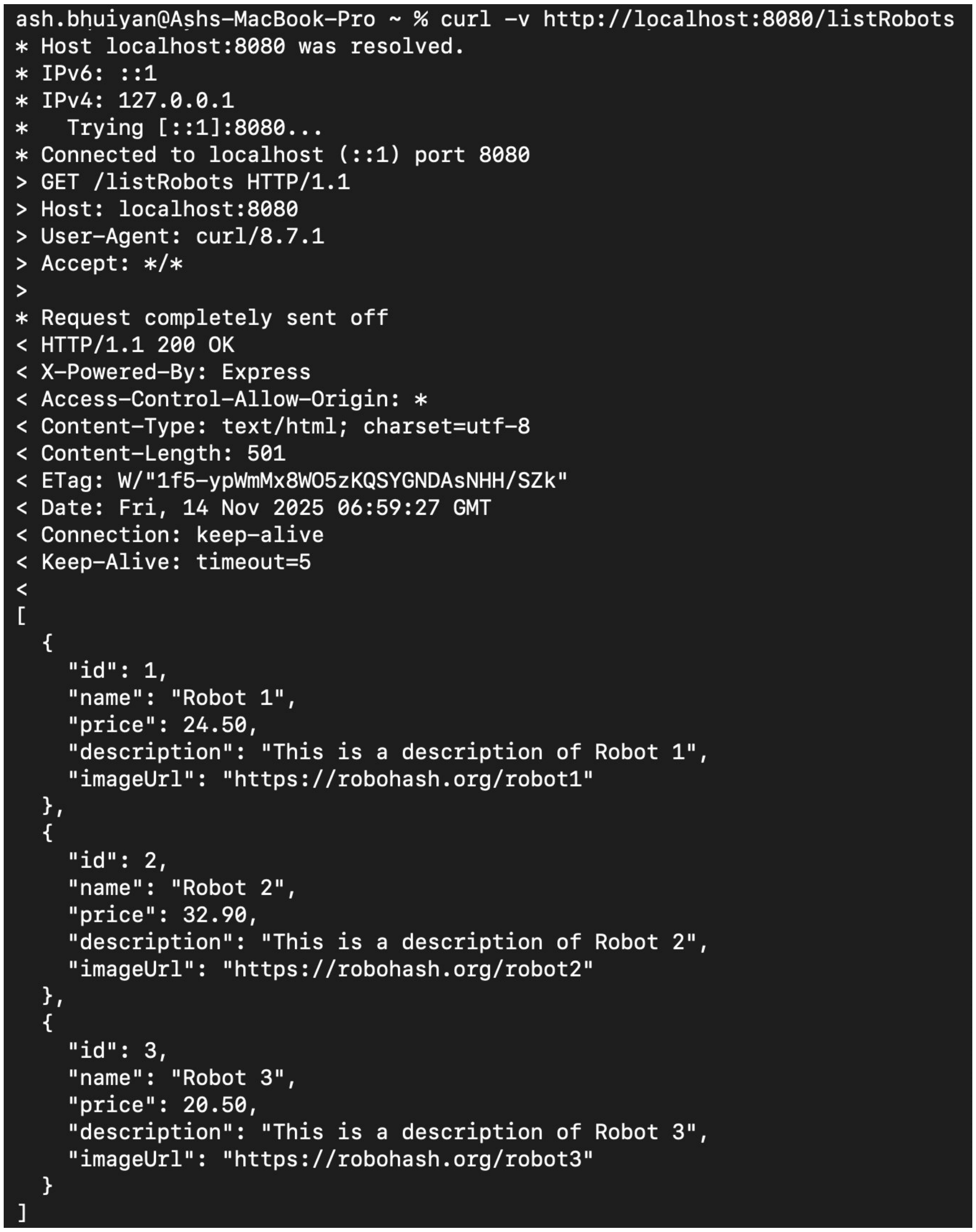
```
ash.bhuiyan@Ashs-MacBook-Pro ~ % curl -v http://localhost:8080/listRobots

* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* Connected to localhost (::1) port 8080
> GET /listRobots HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 404 Not Found
< X-Powered-By: Express
< Access-Control-Allow-Origin: *
< Content-Type: text/html; charset=utf-8
< Content-Length: 501
< ETag: W/"1f5-ypWmMx8WO5zKQSYGNDAsNHH/SZk"
< Date: Fri, 14 Nov 2025 06:58:07 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
[
  {
    "id": 1,
    "name": "Robot 1",
    "price": 24.50,
    "description": "This is a description of Robot 1",
    "imageUrl": "https://robohash.org/robot1"
  },
  {
    "id": 2,
    "name": "Robot 2",
    "price": 32.90,
    "description": "This is a description of Robot 2",
    "imageUrl": "https://robohash.org/robot2"
  },
  {
    "id": 3,
    "name": "Robot 3",
    "price": 20.50,
    "description": "This is a description of Robot 3",
    "imageUrl": "https://robohash.org/robot3"
  }
]
```

Figure 6: curl -v http://localhost:8080/listRobots (404 Not Found)

4.2 Switching back to 200 OK

After the experiment, I changed the status code back to 200 in the /listRobots route and re-ran:
curl -v http://localhost:8080/listRobots

The response again shows HTTP/1.1 200 OK along with the robots JSON body. This is the behavior the REST API should have in normal operation.



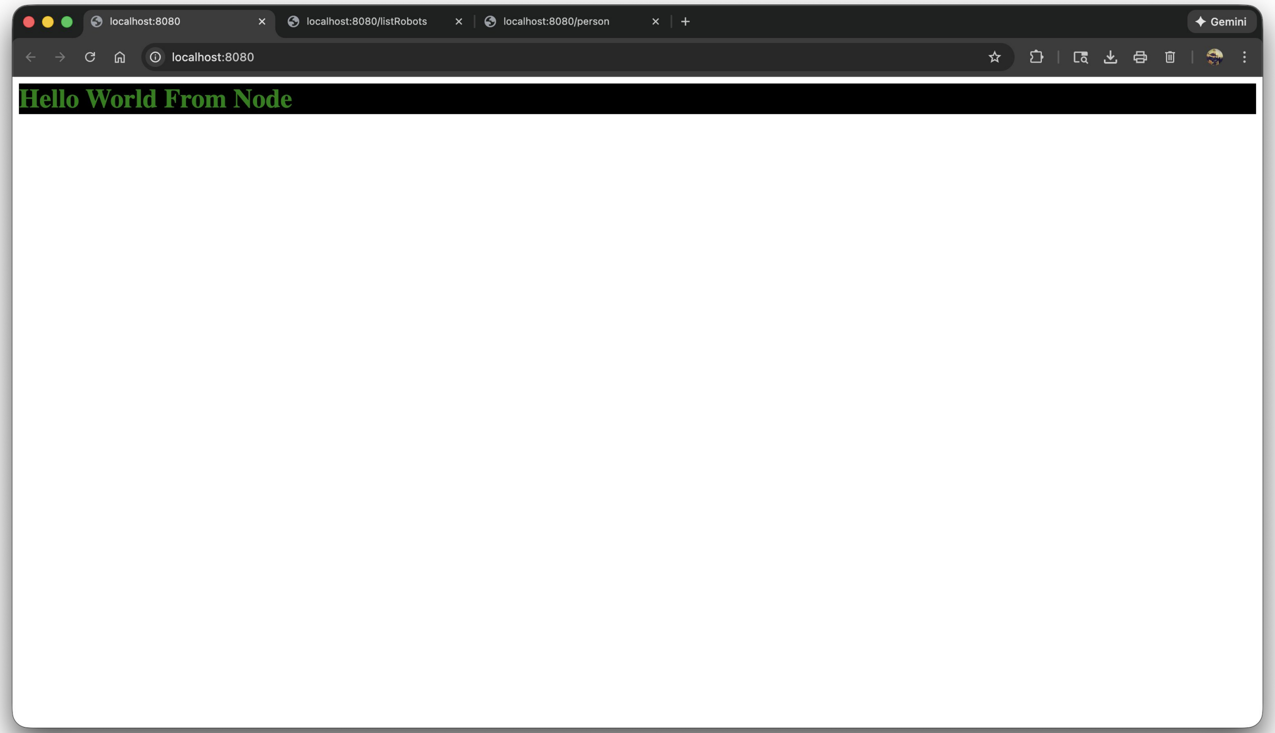
```
ash.bhuiyan@Ashs-MacBook-Pro ~ % curl -v http://localhost:8080/listRobots
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* Connected to localhost (::1) port 8080
> GET /listRobots HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Access-Control-Allow-Origin: *
< Content-Type: text/html; charset=utf-8
< Content-Length: 501
< ETag: W/"1f5-ypWmMx8WO5zKQSYGNDAsNHH/SZk"
< Date: Fri, 14 Nov 2025 06:59:27 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
[
  {
    "id": 1,
    "name": "Robot 1",
    "price": 24.50,
    "description": "This is a description of Robot 1",
    "imageUrl": "https://robohash.org/robot1"
  },
  {
    "id": 2,
    "name": "Robot 2",
    "price": 32.90,
    "description": "This is a description of Robot 2",
    "imageUrl": "https://robohash.org/robot2"
  },
  {
    "id": 3,
    "name": "Robot 3",
    "price": 20.50,
    "description": "This is a description of Robot 3",
    "imageUrl": "https://robohash.org/robot3"
  }
]
```

Figure 7: curl -v http://localhost:8080/listRobots (200 OK, after reverting)

5. Browser Tests and Visual Output

5.1 http://localhost:8080/ – HTML output

When I open http://localhost:8080/ in the browser, the page displays a black bar with green text "Hello World From Node". This matches the HTML string returned in the root route. In the browser's Network tab, the request to / shows method GET, status 200, and Content-Type text/html.



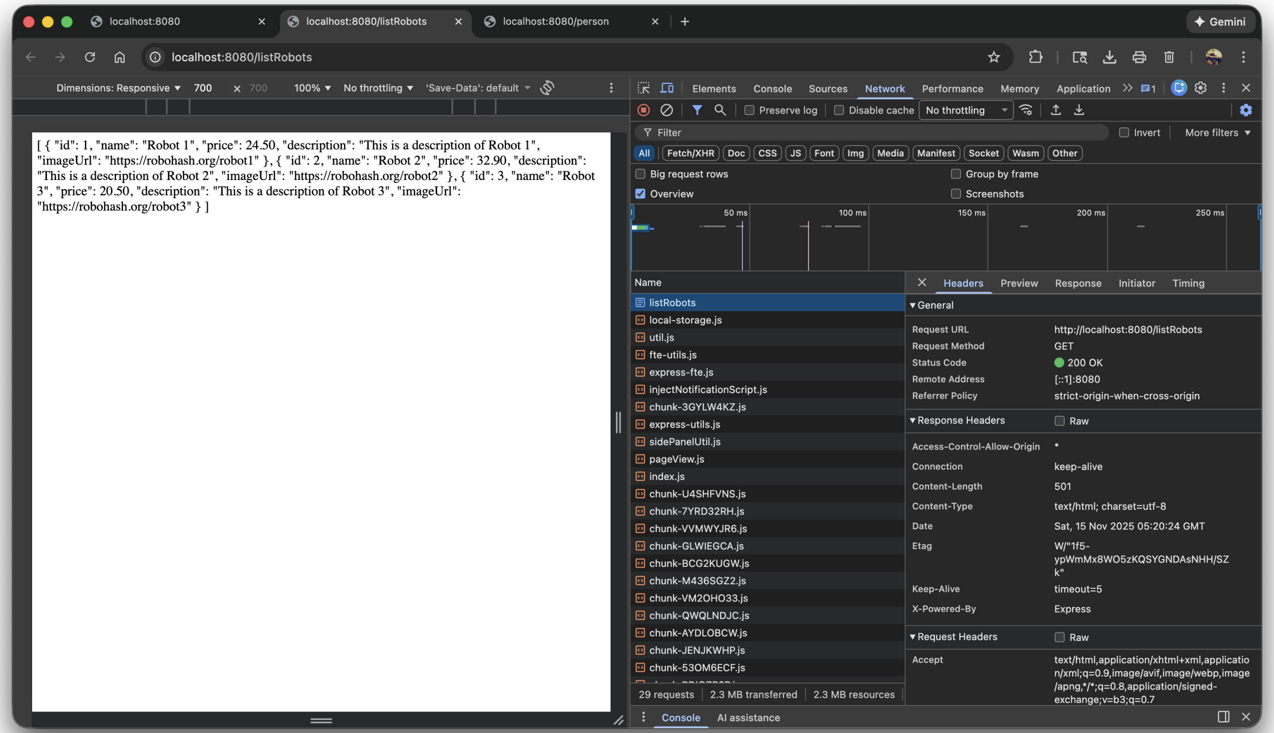
Hello World From Node

Figure 8: Browser showing, "Hello World From Node at /"

5.2 http://localhost:8080/listRobots – robots JSON

Opening http://localhost:8080/listRobots in the browser shows the JSON array with the three robots. The browser simply renders the raw JSON text returned by the server. In the Network tab, the request has method GET, status 200, and a response body equal to the contents of robots.json.

Screenshot is on the next page


localhost:8080
localhost:8080/listRobots
localhost:8080/person
Gemini
Dimensions: Responsive 700 x 700 100% No throttling 'Save-Data': default
Elements Console Sources Network Performance Memory Application
Preserve log Disable cache No throttling
Filter Invert More filters
All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other
Big request rows Group by frame
Overview Screenshots
50 ms 100 ms 150 ms 200 ms 250 ms
Name
listRobots
local-storage.js
util.js
fte-utils.js
express-fte.js
injectNotificationScript.js
chunk-3GYLW4KZ.js
express-utils.js
sidePanelUtil.js
pageView.js
index.js
chunk-U4SHFVNS.js
chunk-7YRD32RH.js
chunk-VVMWYJR6.js
chunk-GLWIEGCA.js
chunk-BCG2KUGW.js
chunk-M436SGZ2.js
chunk-VM2OHO33.js
chunk-QWQLNDJC.js
chunk-AYDLOBCW.js
chunk-JENJKWHP.js
chunk-53OM6ECF.js
29 requests 2.3 MB transferred 2.3 MB resources
Headers Preview Response Initiator Timing
Console AI assistance

```
[ { "id": 1, "name": "Robot 1", "price": 24.50, "description": "This is a description of Robot 1",
"imageUrl": "https://robohash.org/robot1" }, { "id": 2, "name": "Robot 2", "price": 32.90, "description":
"This is a description of Robot 2", "imageUrl": "https://robohash.org/robot2" }, { "id": 3, "name": "Robot
3", "price": 20.50, "description": "This is a description of Robot 3", "imageUrl":
"https://robohash.org/robot3" } ]
```

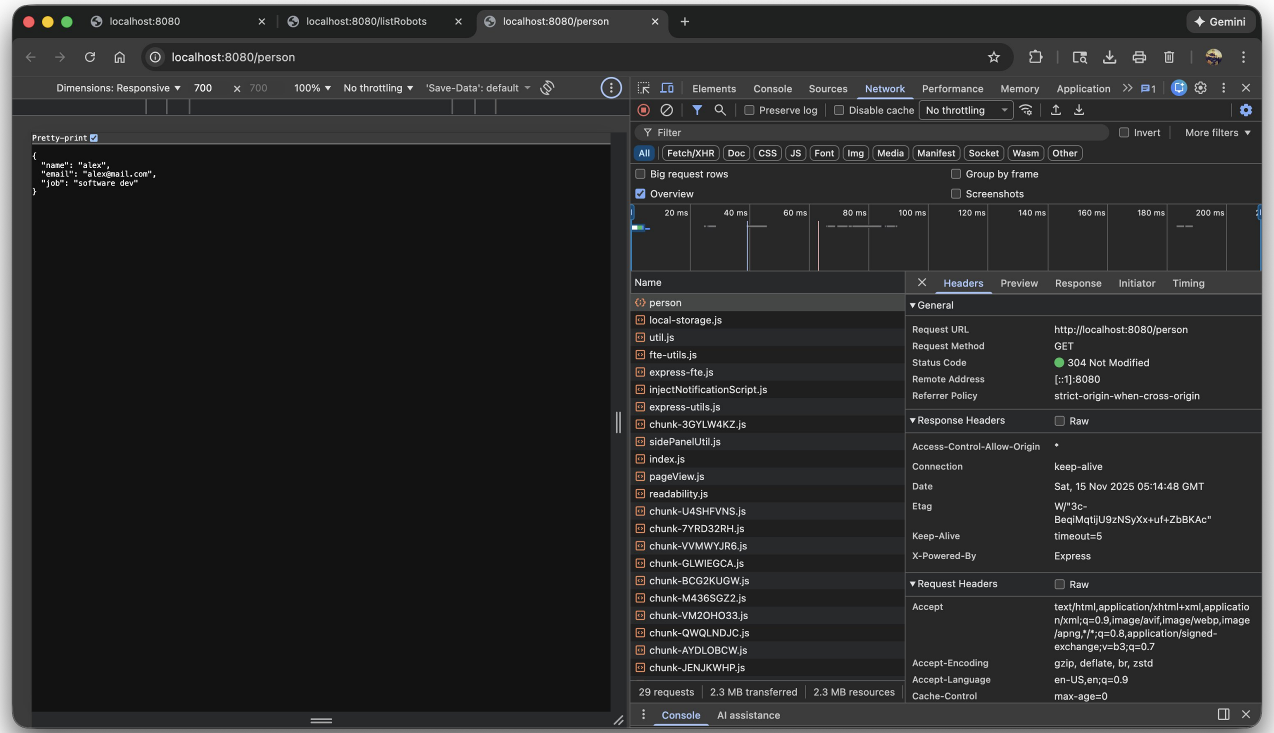
```
▼General
Request URL                  http://localhost:8080/listRobots
Request Method               GET
Status Code                  ● 200 OK
Remote Address               [::1]:8080
Referrer Policy              strict-origin-when-cross-origin
▼Response Headers            Raw
Access-Control-Allow-Origin  *
Connection                   keep-alive
Content-Length               501
Content-Type                 text/html; charset=utf-8
Date                         Sat, 15 Nov 2025 05:20:24 GMT
Etag                         W/"1f5-
                             ypWmMx8WO5zKQSYGNDAsNHH/SZ
                             k"
Keep-Alive                   timeout=5
X-Powered-By                 Express
▼Request Headers             Raw
Accept                       text/html,application/xhtml+xml,applicatio
                             n/xml;q=0.9,image/avif,image/webp,image
                             /apng,*/*;q=0.8,application/signed-
                             exchange;v=b3;q=0.7
```

Figure 9: Browser showing, "JSON array from /listRobots"

5.3 http://localhost:8080/person – person object as JSON

Visiting http://localhost:8080/person in the browser shows the person object as formatted JSON with fields name, email, and job. The developer tools Network tab shows that this response also has method GET, status 200, and Content-Type application/json. This confirms that the API is returning structured JSON data for this endpoint.

Screenshot is on the next page



localhost:8080/person

Pretty-print ✓

```
{
  "name": "alex",
  "email": "alex@mail.com",
  "job": "software dev"
}
```

```
Name
person
local-storage.js
util.js
fte-utils.js
express-fte.js
injectNotificationScript.js
express-utils.js
chunk-3GYLW4KZ.js
sidePanelUtil.js
index.js
pageView.js
readability.js
chunk-U4SHFVNS.js
chunk-7YRD32RH.js
chunk-VVMWYJR6.js
chunk-GLWIEGCA.js
chunk-BCG2KUGW.js
chunk-M436SGZ2.js
chunk-VM2OHO33.js
chunk-QWQLNDJC.js
chunk-AYDLOBCW.js
chunk-JENJKWHP.js
29 requests | 2.3 MB transferred | 2.3 MB resources
```

```
▼General
Request URL                   http://localhost:8080/person
Request Method                GET
Status Code                   ● 304 Not Modified
Remote Address                [::1]:8080
Referrer Policy               strict-origin-when-cross-origin
▼Response Headers             ☐ Raw
Access-Control-Allow-Origin   *
Connection                    keep-alive
Date                          Sat, 15 Nov 2025 05:14:48 GMT
Etag                          W/"3c-
                              BeqiMqtijU9zNSyXx+uf+ZbBKAc"
Keep-Alive                    timeout=5
X-Powered-By                  Express
▼Request Headers              ☐ Raw
Accept                        text/html,application/xhtml+xml,applicatio
                              n/xml;q=0.9,image/avif,image/webp,image
                              /apng,*/*;q=0.8,application/signed-
                              exchange;v=b3;q=0.7
Accept-Encoding               gzip, deflate, br, zstd
Accept-Language               en-US,en;q=0.9
Cache-Control                 max-age=0
```

Figure 10: Browser showing, "JSON object from /person"

6. Comparing /listRobots and /person

Both /listRobots and /person use the HTTP GET method and return JSON data, but they serve different purposes:

- /listRobots reads data from a local JSON file (robots.json) and returns an array of multiple robot objects. It behaves like a typical REST endpoint that returns a list of resources.
- /person constructs a single JavaScript object in code and returns it as JSON. It behaves like an endpoint that returns one resource instance.

From the curl output, both endpoints show status 200 OK when the server is running. However, /listRobots originally reported Content-Type as text/html because the response was sent as a string read from a file, while /person used application/json because Express detected that an object was being sent. In the browser, /listRobots displays a JSON array, and /person displays a single JSON object.

7. Summary

In this activity I set up a Node.js and Express server that exposes three GET endpoints: /, /listRobots, and /person. I created a robots.json file and used fs.readFile to send its contents as the response to /listRobots. I used curl to observe HTTP requests and responses, including the behavior when the server is stopped and when the status code for /listRobots is changed from 200 to 404. I also viewed the responses in the browser and the Network tab, which helped reinforce how REST APIs use URLs, HTTP methods, status codes, and JSON to communicate between client and server.