สยามชำนาญกิจ
SIAM CHAMNAN KIT

Web API Design

Agenda day 1

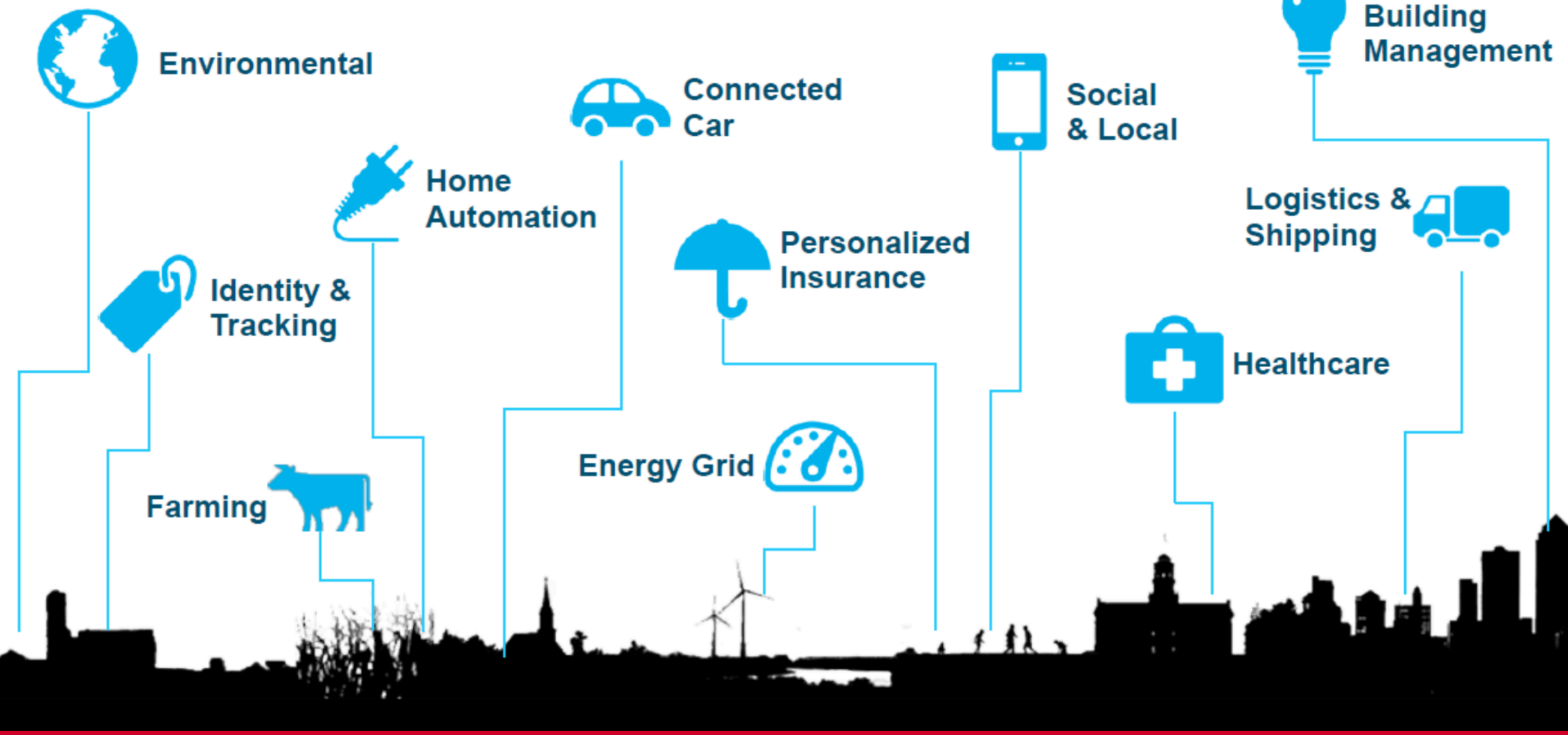
- Understanding APIs
- Principles of modern Web APIs
- Make a great APIs
- Modeling APIs
- From modelling to API design
- Workshop

Agenda day 2

- Working with REST
- Documenting your APIs
- Secure your APIs
- Testing your APIs
- Workshop

Understanding APIs

The world of APIs

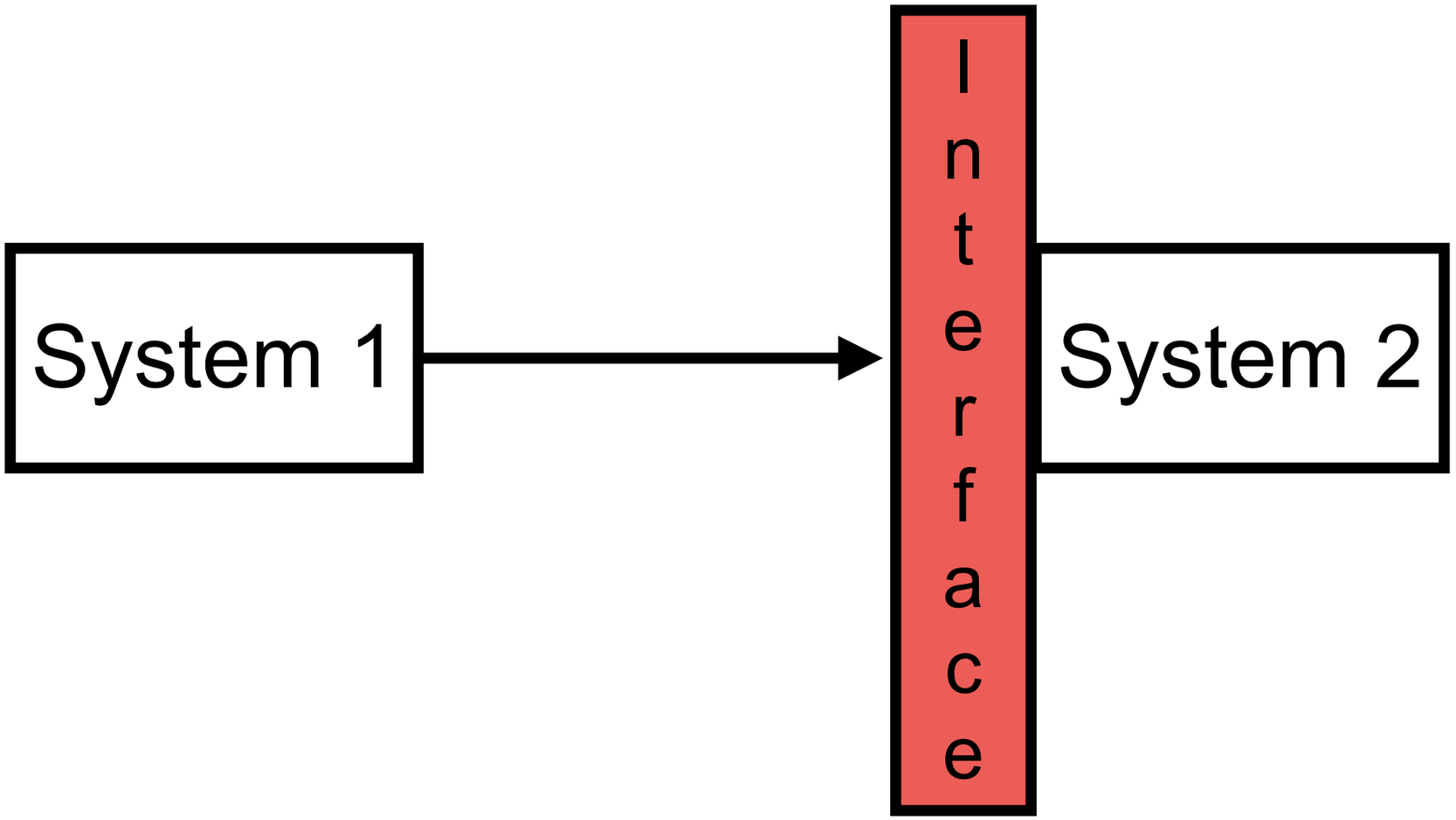


What is an APIs ?

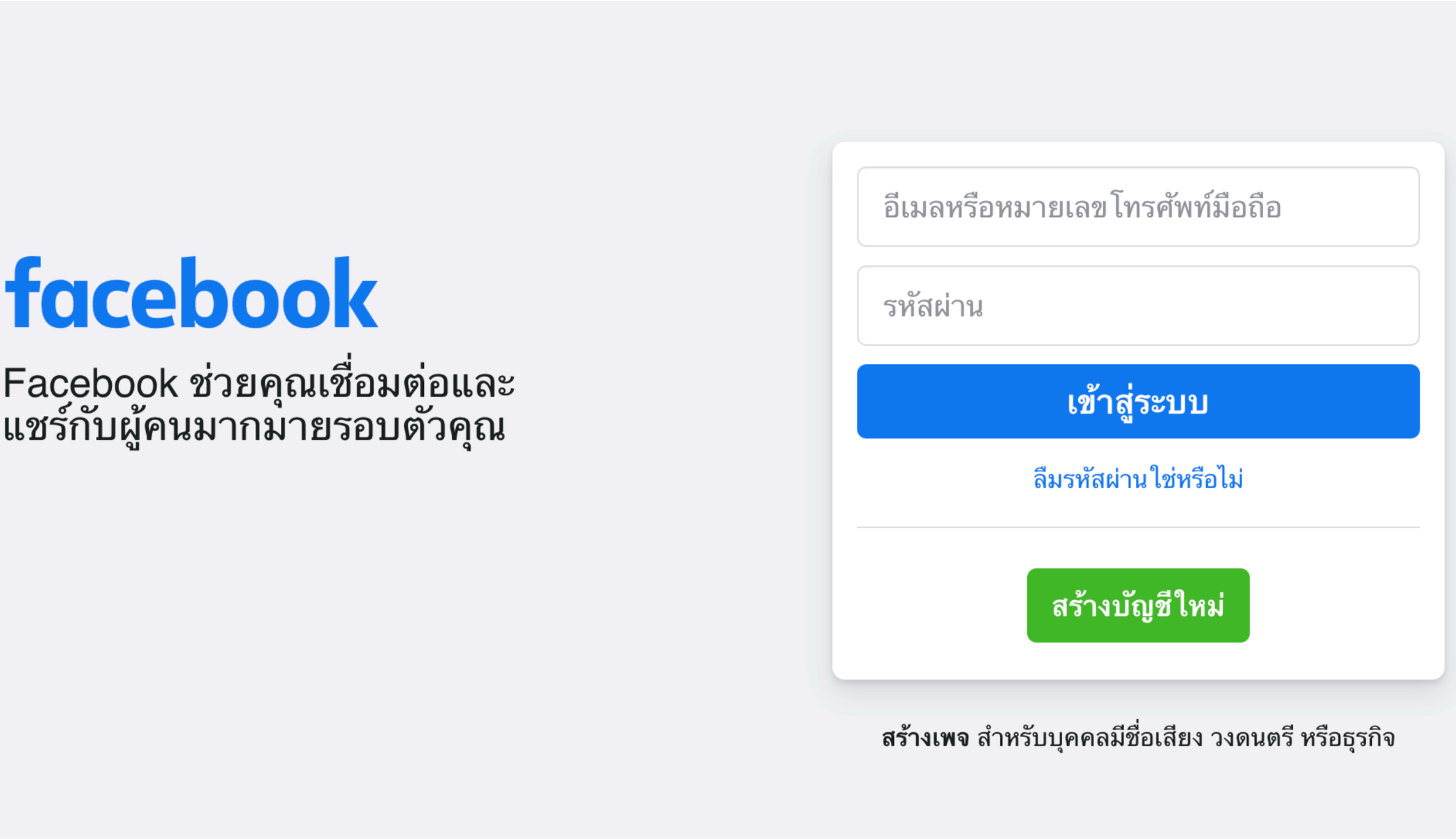
Application **P**rogramming Interfaces

What is an APIs ?

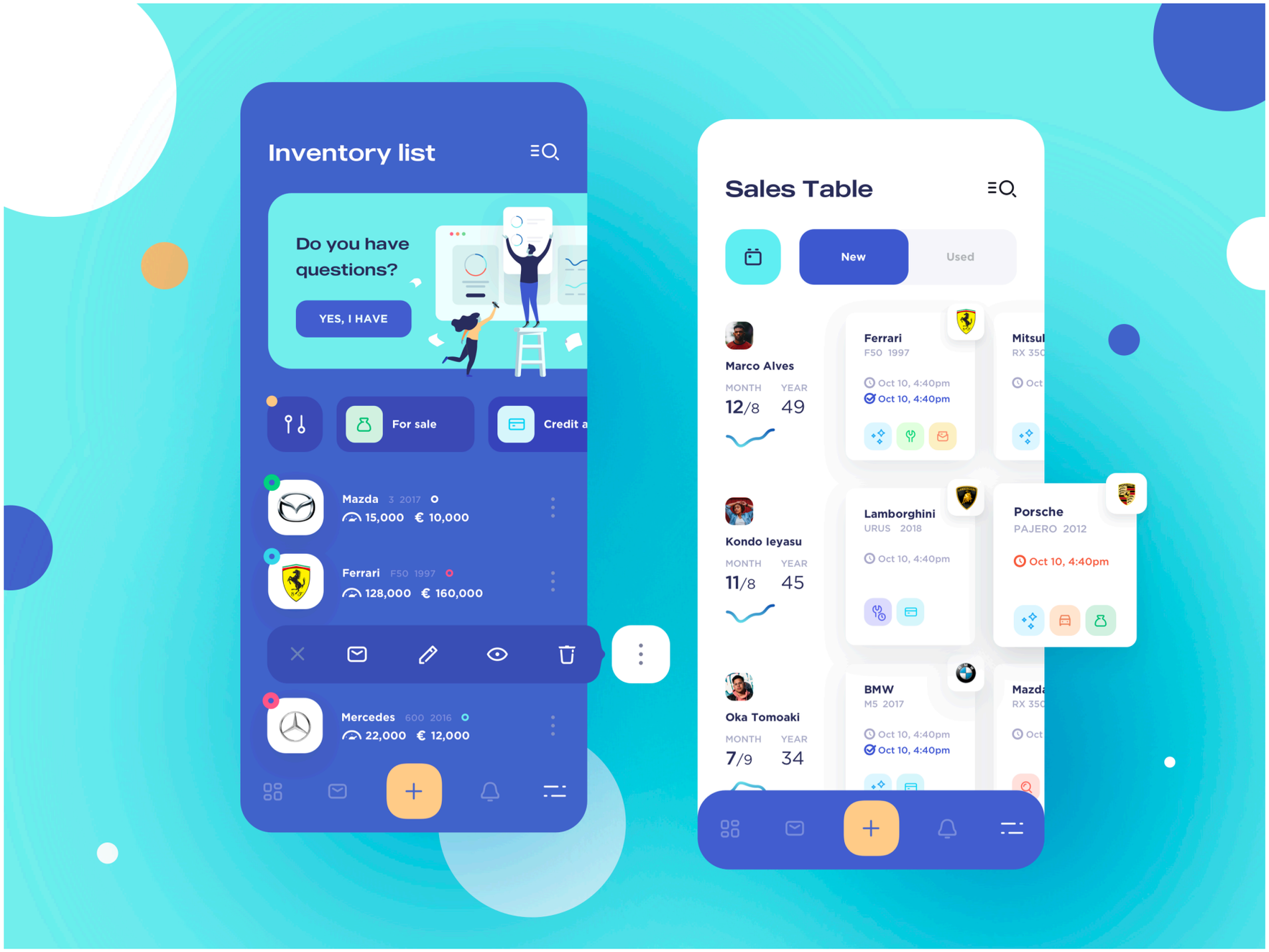
Software uses **interfaces** to communicate



User Interface



User Interface

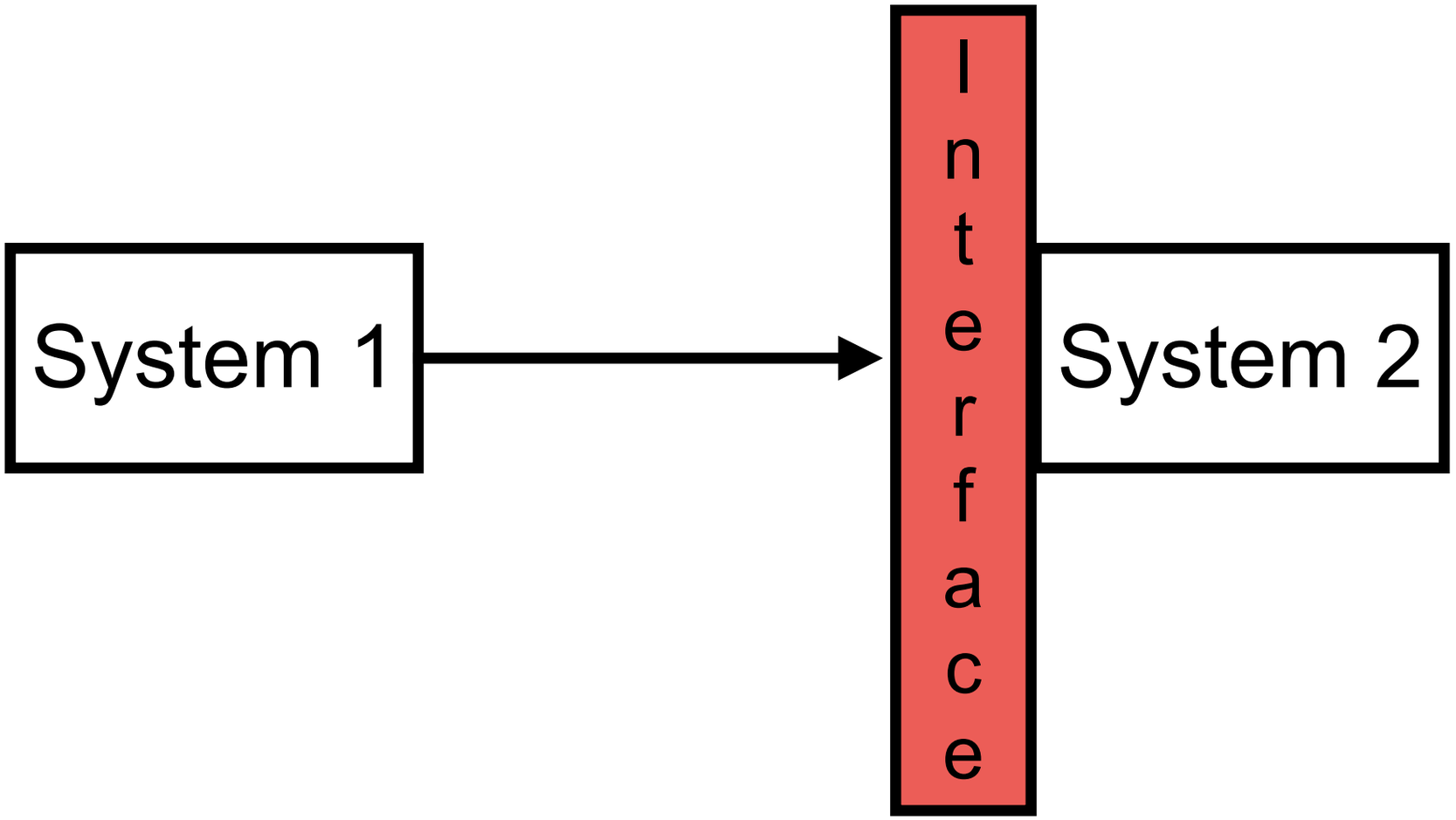


Interface

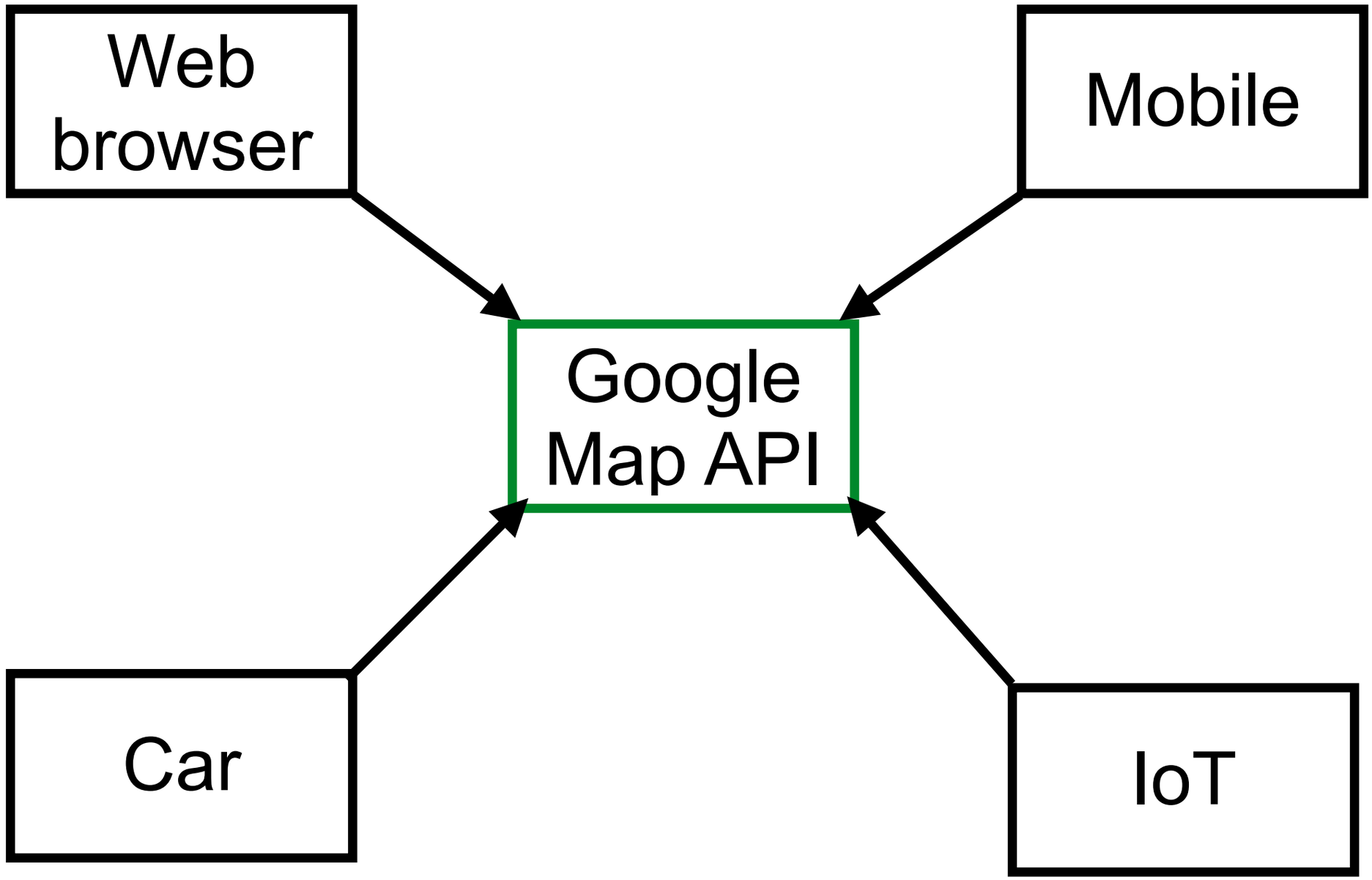

```
{
    age: "69",
    age_display: "69 year",
  + aliases: […],
    @context: "/terms/",
  + lab: {…},
  + biosample_ontology: {…},
    references: [ ],
    status: "released",
  + dbxrefs: […],
    internal_tags: [ ],
    description: "mammary gland, adenocarcinoma",
    @id: "/biosamples/ENCBS000AAA/",
    age_units: "year",
    uuid: "56e94f2b-25ac-4c58-9828-f63b66220999",
    parent_of: [ ],
  + submitted_by: {…},
  + documents: […],
    genetic_modifications: [ ],
  + organism: {…},
    health_status: "breast cancer (adenocarcinoma)",
  + @type: […],
    alternate_accessions: [ ],
    url: "http://www.atcc.org/Products/All/HTB-22.aspx",
```


What is an APIs ?

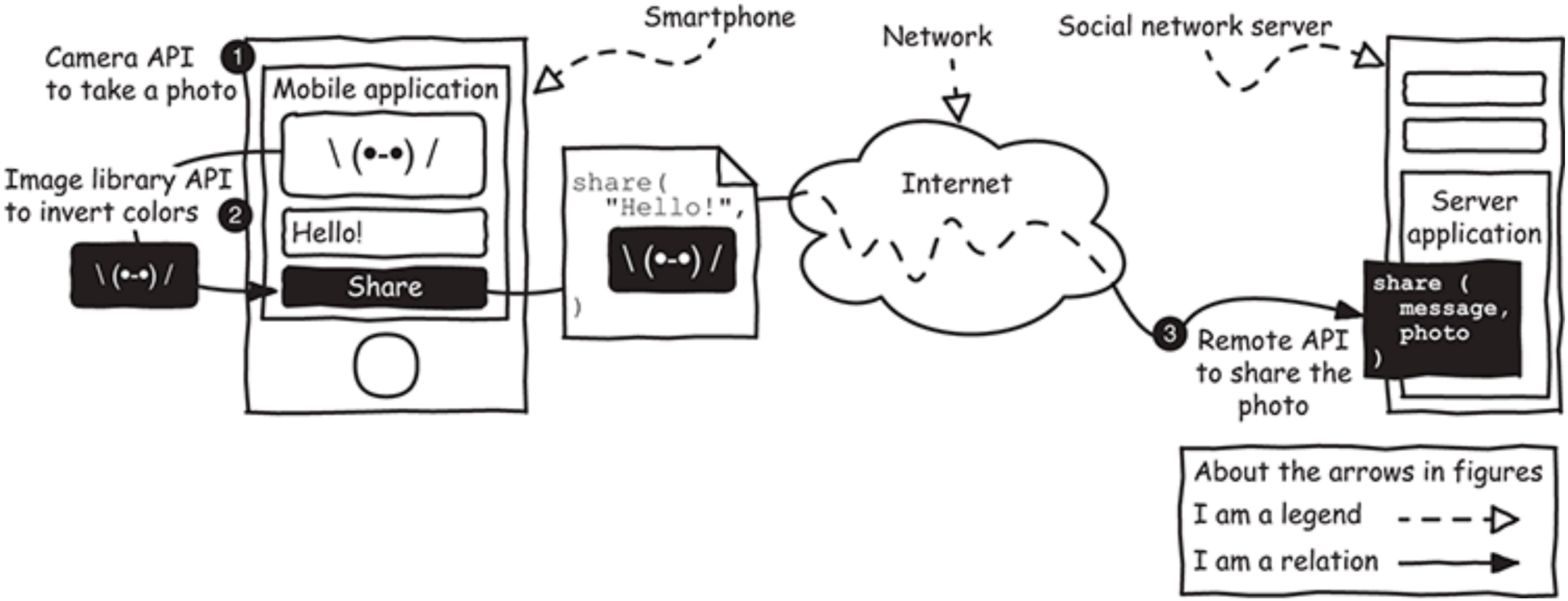
The interface that a software program presents to other programs, to **humans**



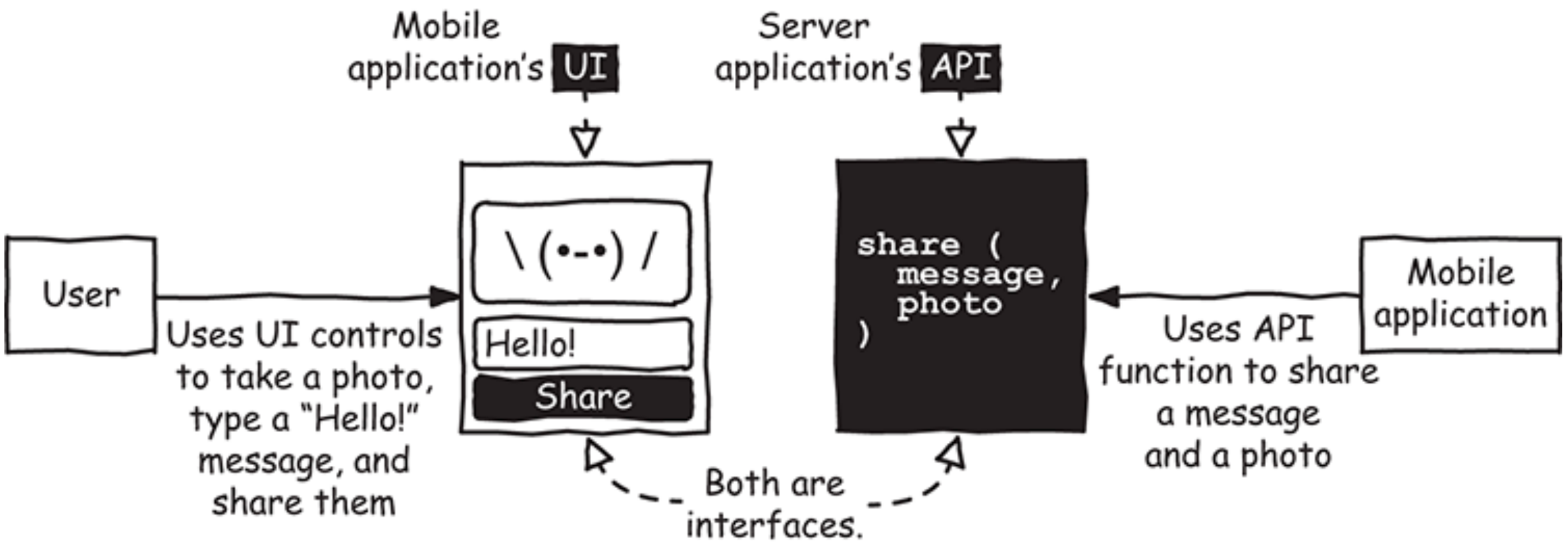
Google Map



Example of APIs with Mobile

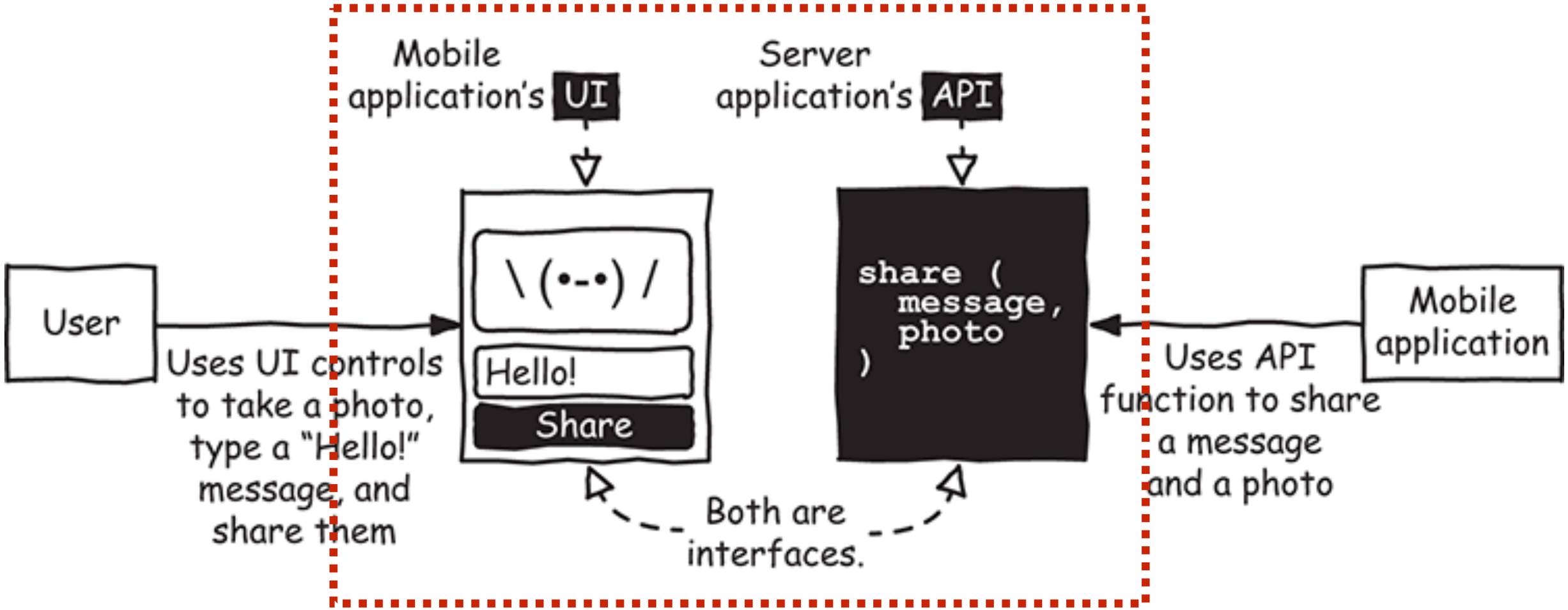


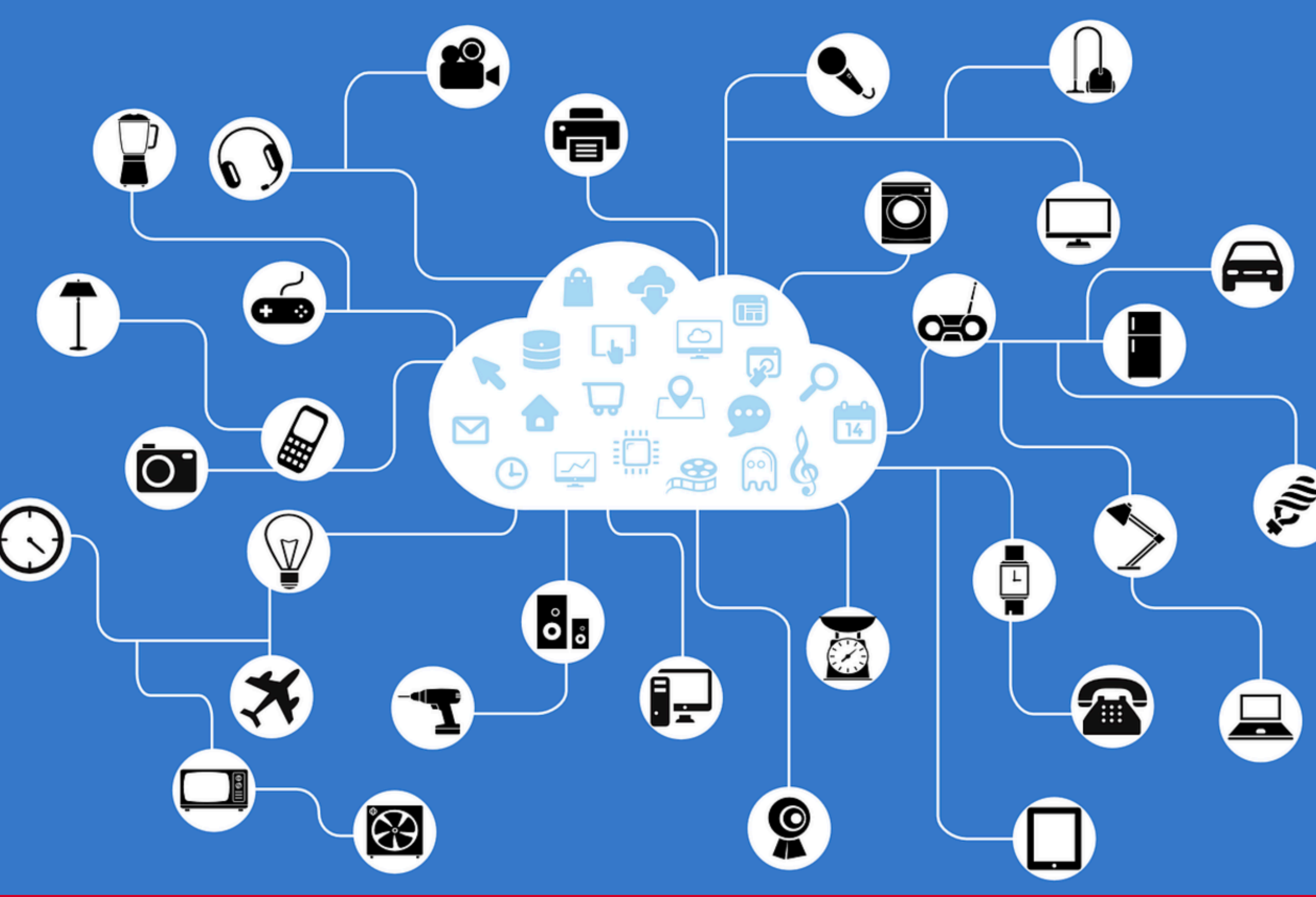
UI to API



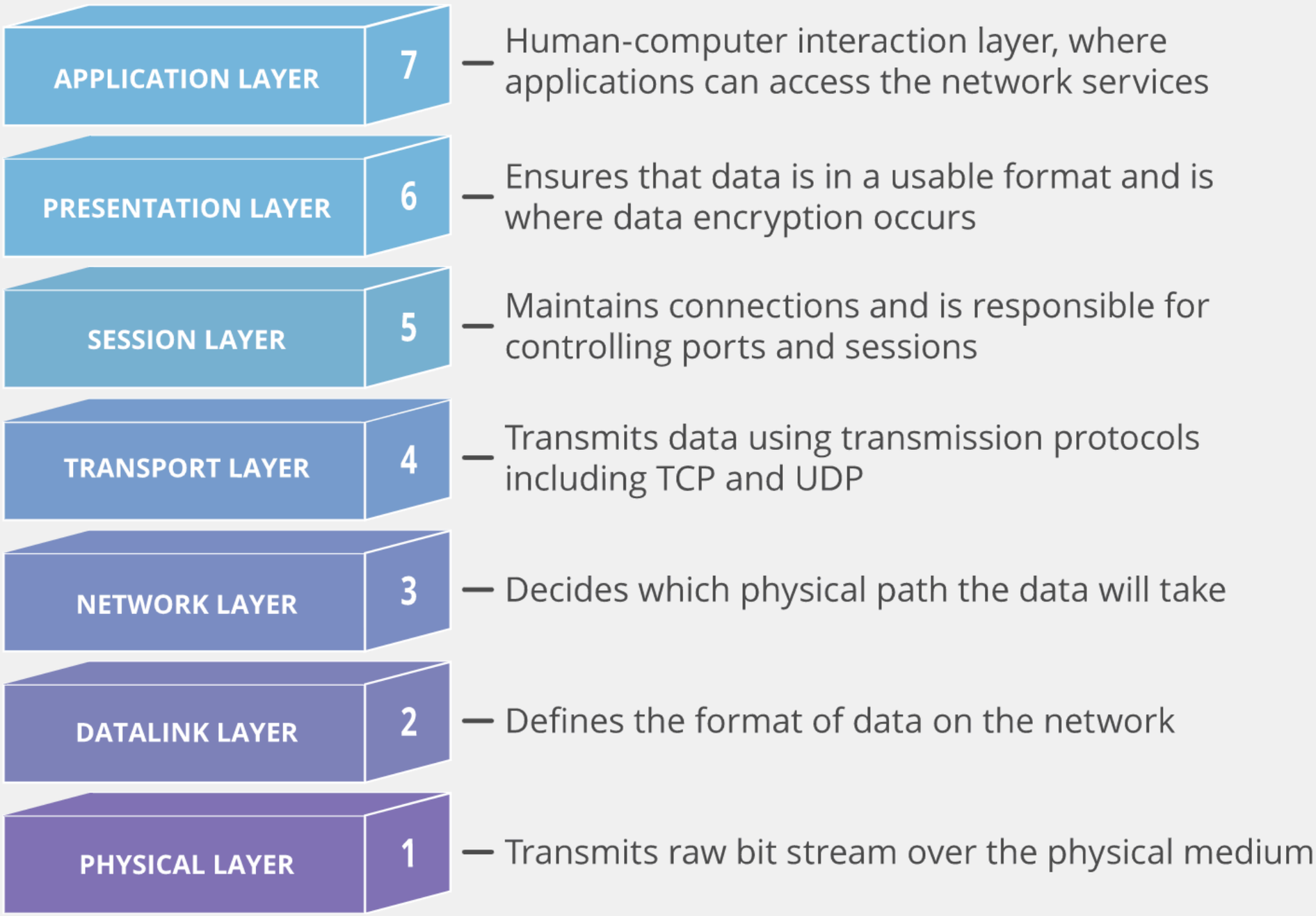
Communication protocol ?

Communication protocol ?





Open Systems Interconnection (OSI)



https://www.cloudflare.com/learning/network-layer/what-is-a-protocol/

Web APIs

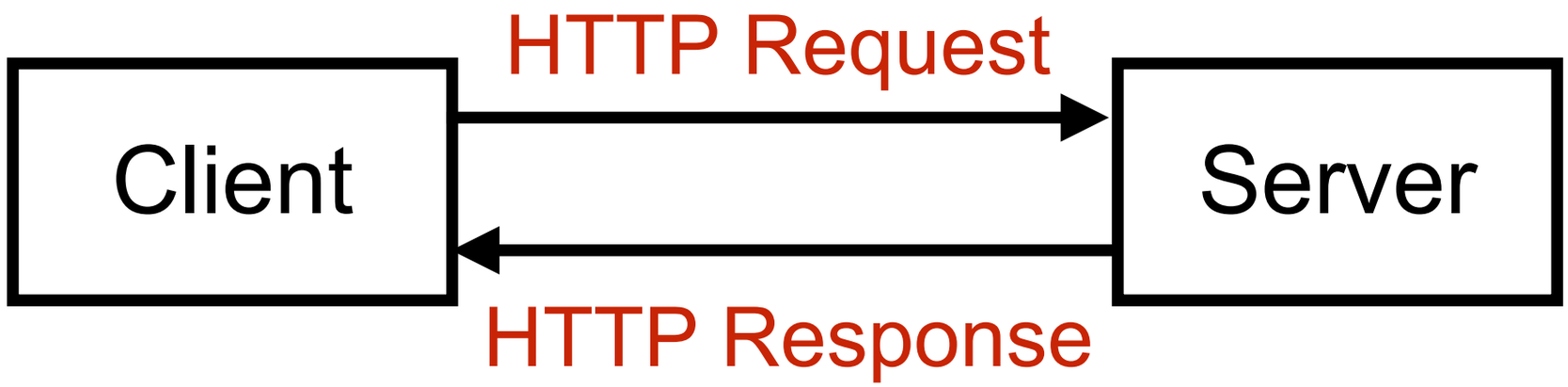
HTTP
Hyper Text Transfer Protocol

HTTP ?

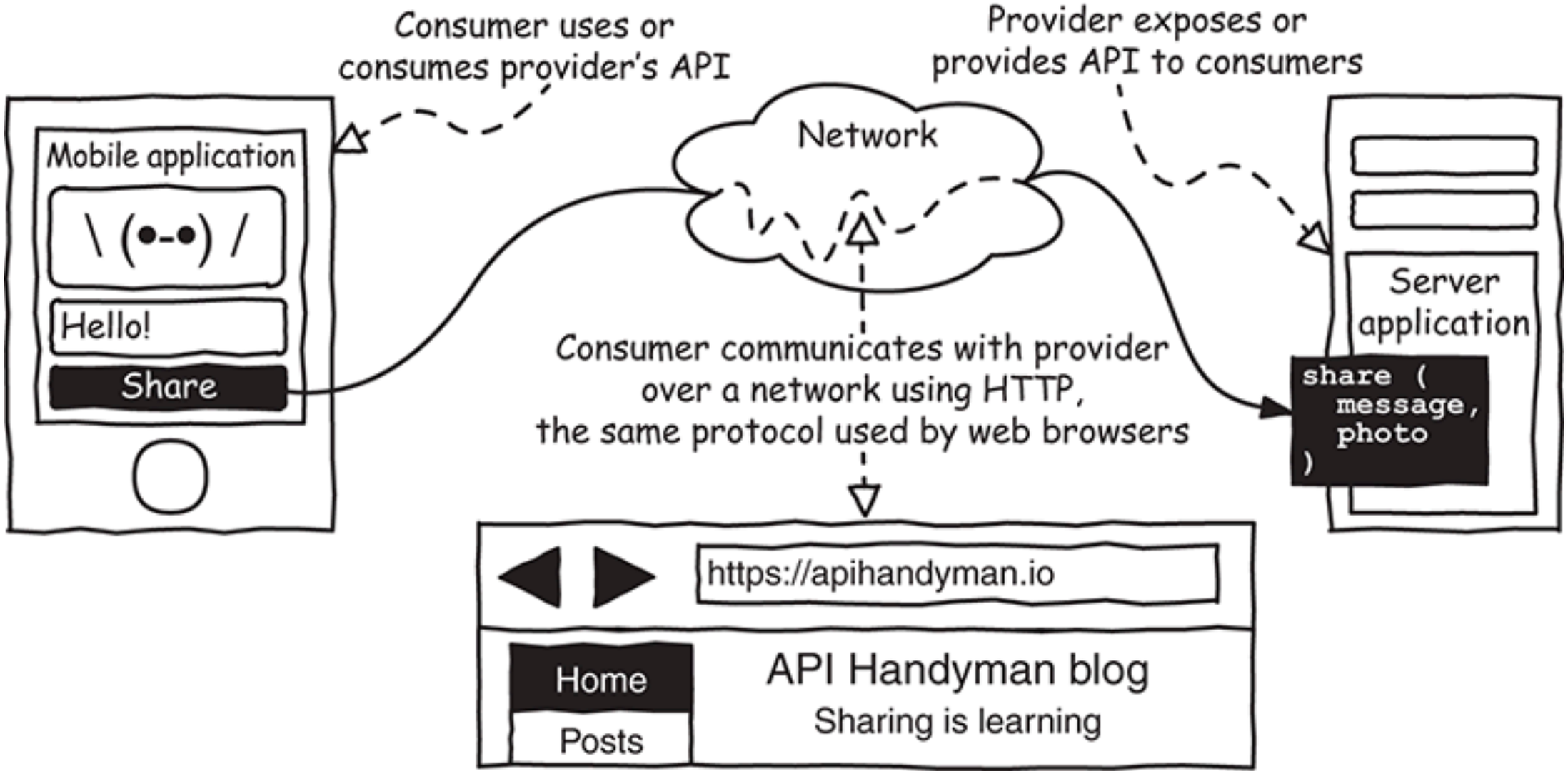
Foundation of World Wide Web (www)

Application layer protocol

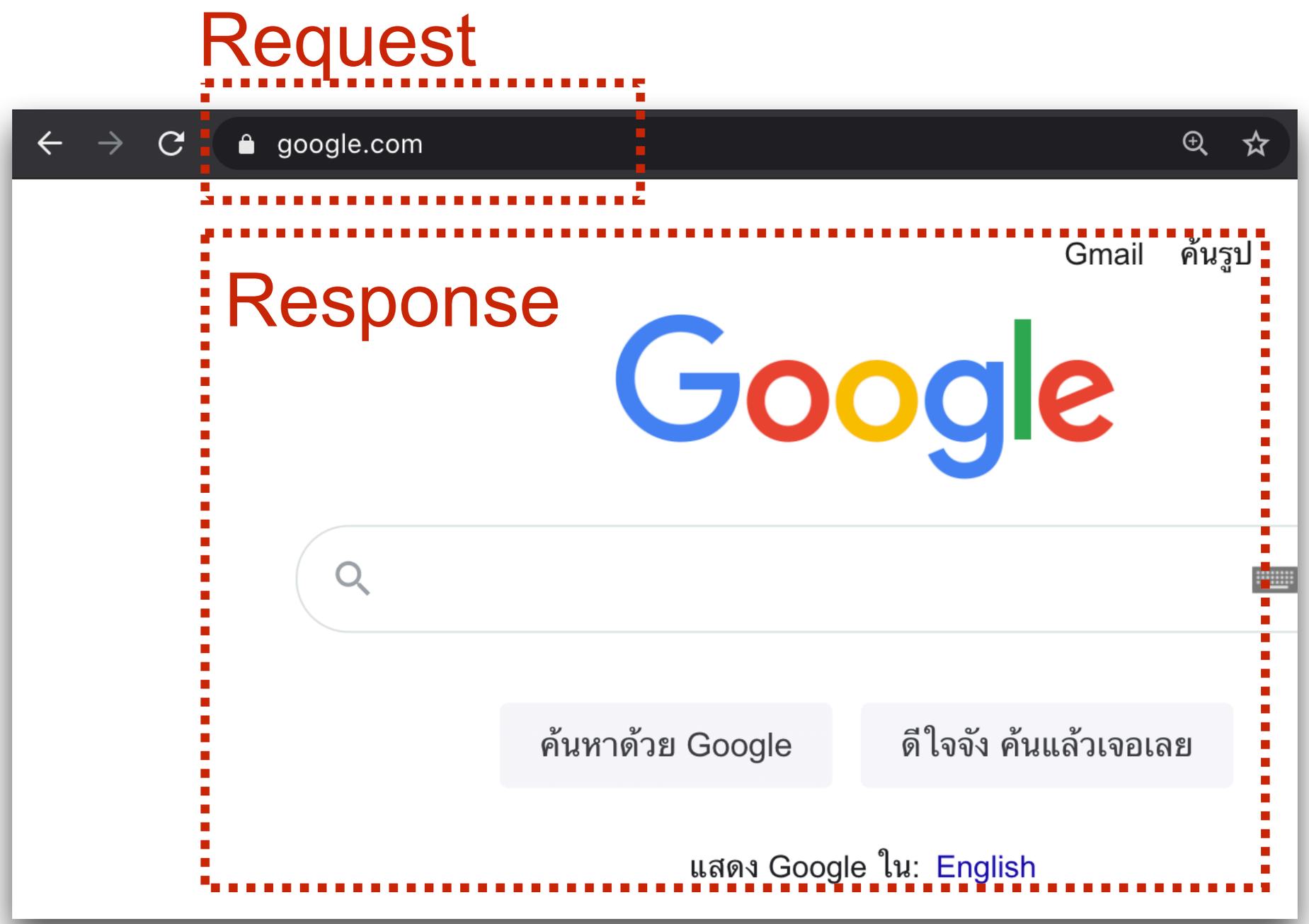
Designed to transfer information networked devices



HTTP

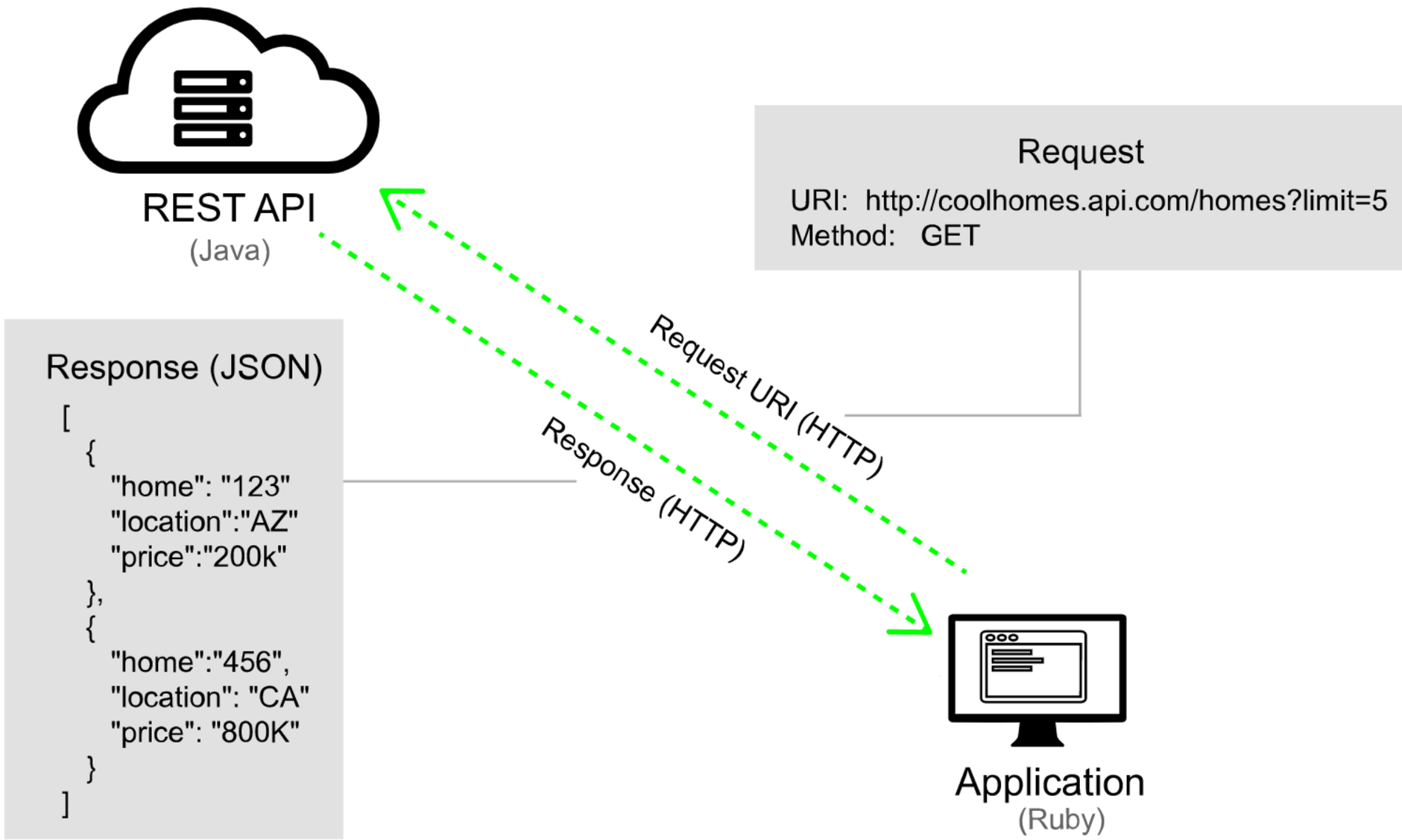


Example of request and response

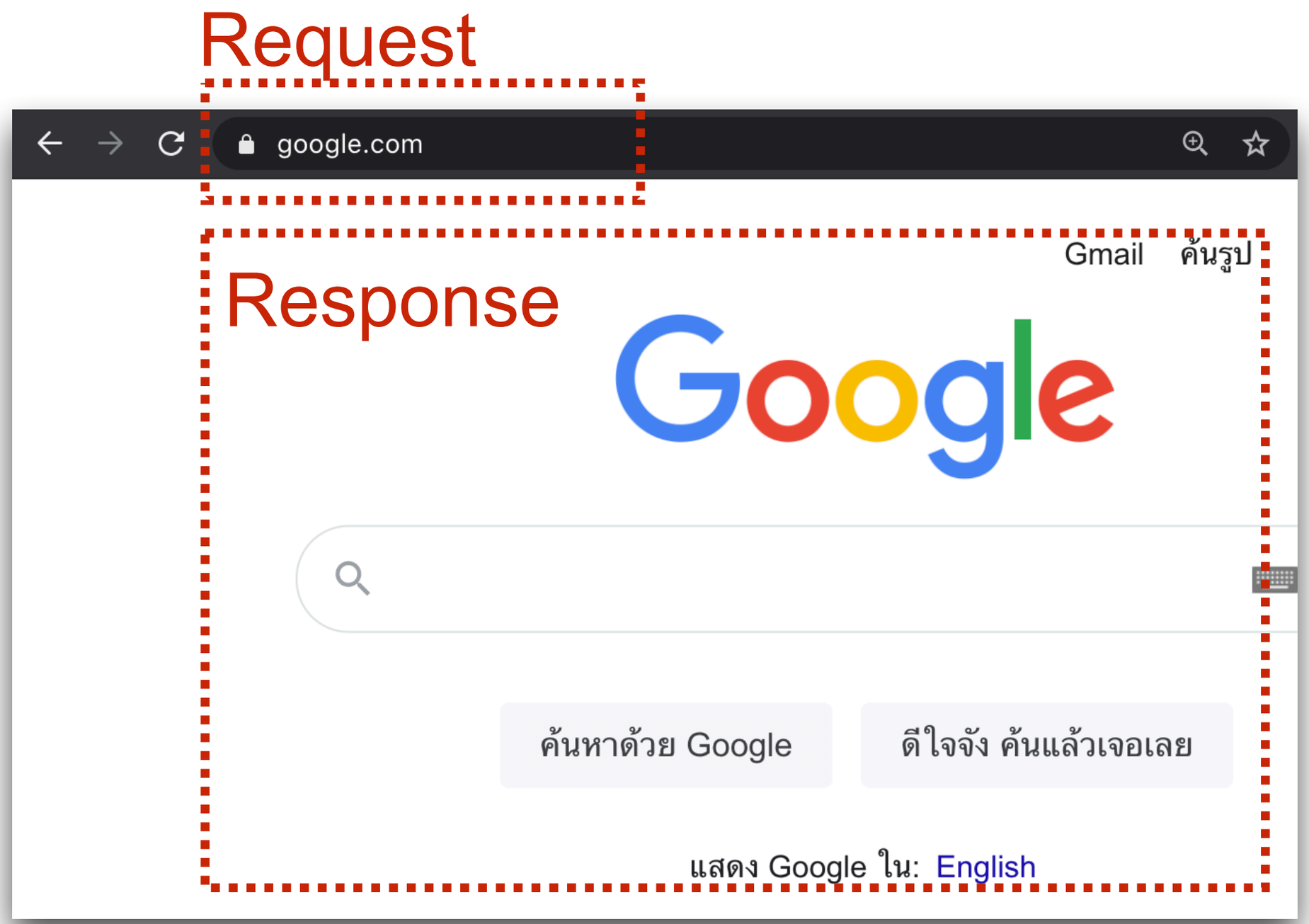


Web APIs + HTTP

Working with **Request** and **Response**



Example



Example !!

▼ General

Request URL: https://www.google.com/complete/search?q&cp=0&c

0jvOX9qyCv_az7sPufWhmAE.160735125044&dpr=2.5

Request Method: GET

Status Code: 200

Remote Address: 216.58.221.196:443

Referrer Policy: origin

▼ Response Headers

alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h

43=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"

cache-control: private, max-age=3600

content-disposition: attachment; filename="f.txt"

content-encoding: br

Example !!

▼ **Request Headers**

:authority: www.google.com

:method: GET

:path: /complete/search?q&cp=0&client=psy-ab&xssi=t&gs_ri
51250444&dpr=2.5

:scheme: https

accept: */*

accept-encoding: gzip, deflate, br

accept-language: en-US,en;q=0.9,th;q=0.8

cookie: CGIC=IocBdGV4dC9odG1sLGFwcGxpY2F0aW9uL3hodG1sK3h

Example !!

▼ **Query String Parameters** view source view URL encoded

q:

cp: `0`

client: `psy-ab`

xssi: `t`

gs_ri: `gws-wiz`

hl: `en-TH`

authuser: `0`

psi: `0jvOX9qyCv_az7sPufWhmAE.1607351250444`

dpr: `2.5`

Let’s Start with Web APIs

APIs

REST

RPC

GraphQL

WebSocket

WebHook

etc…

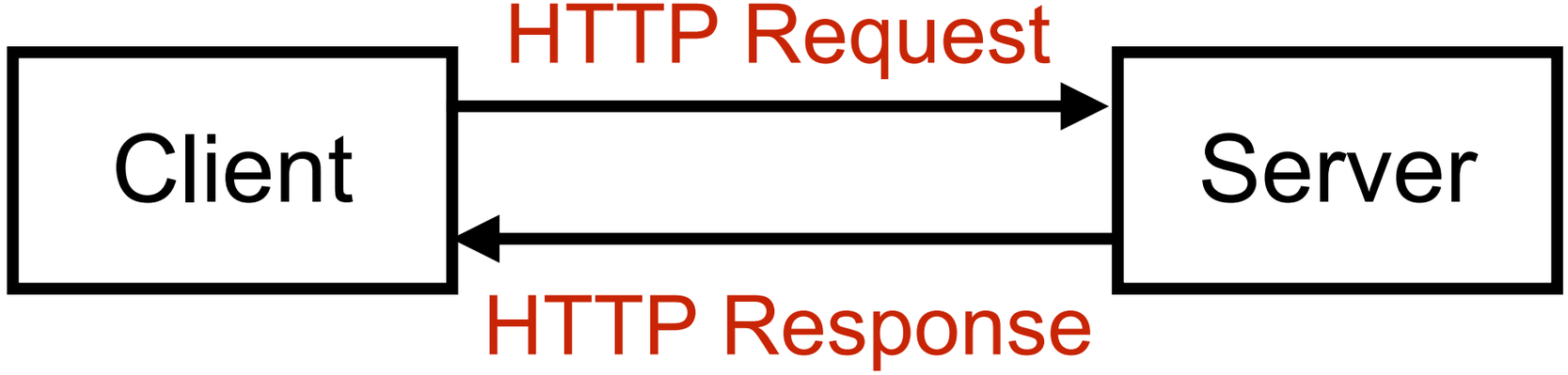
Request and Response APIs

Using HTTP
APIs define a set of **endpoints**
Client and **Server**

Working with HTTP

Client make HTTP requests for data to endpoint on server

Server returns responses



HTTP Request

HTTP Request ?

HTTP version
URL
HTTP method
HTTP request header
HTTP body (optional)

```
1 | GET / HTTP/1.1
2 | Host: developer.mozilla.org
3 | Accept-Language: fr
```

HTTP Request ?

```
1 | GET / HTTP/1.1
2 | Host: developer.mozilla.org
3 | Accept-Language: fr
```

HTTP version = 1.1

HTTP Request ?

```
1 GET / HTTP/1.1
2 Host: developer.mozilla.org
3 Accept-Language: fr
```

HTTP method = GET

HTTP Request ?

```
1 | GET / HTTP/1.1
2 | Host: developer.mozilla.org
3 | Accept-Language: fr
```

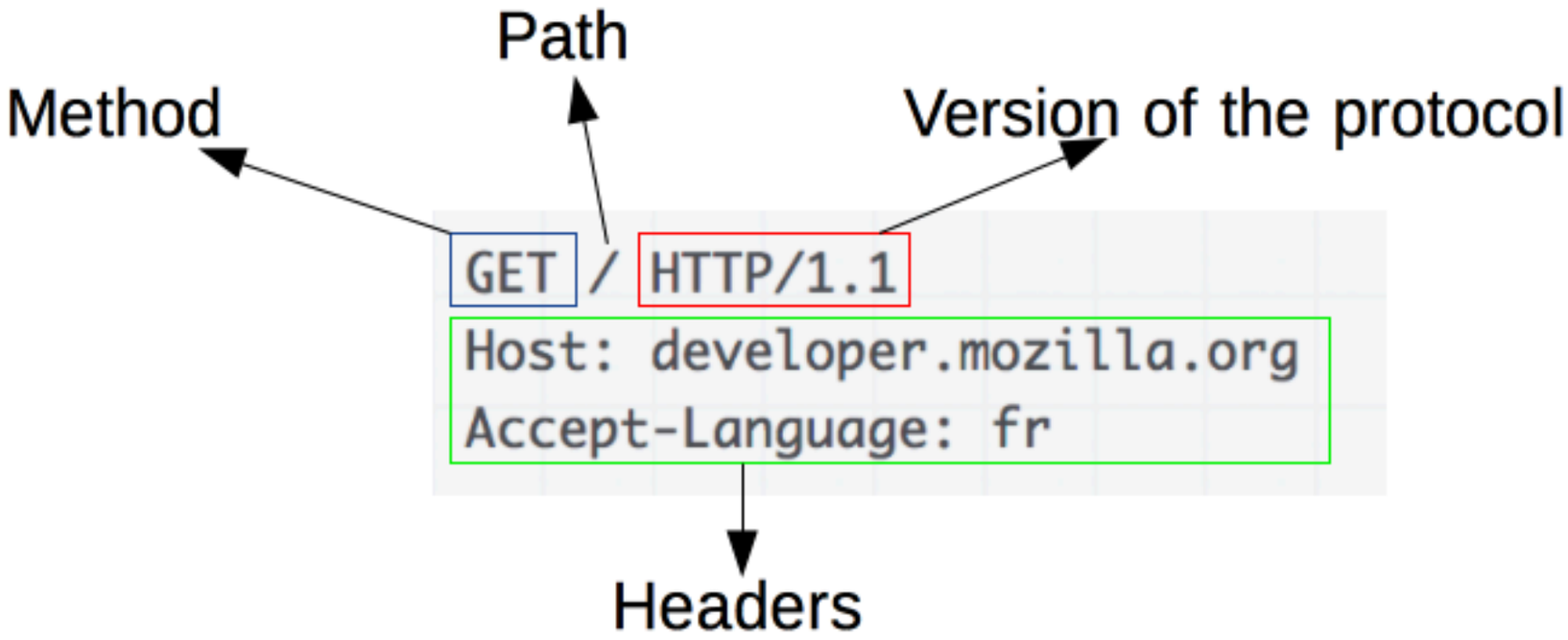
PATH = /

HTTP Request ?

```
1 GET / HTTP/1.1
2 Host: developer.mozilla.org
3 Accept-Language: fr
```

HTTP request header

HTTP Request ?



HTTP method ?

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given **resource**.

Called “HTTP Verbs”

HTTP method ?

HTTP method	Description
GET	Retrive data
POST	Create data
PUT	Update data
DELETE	Delete data

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Path ?

The path of the **resource** to fetch from URL

http://developer.mozilla.org/

URL = Uniform Resource Locator

https://en.wikipedia.org/wiki/URL

Path ?

The path of the **resource** to fetch from URL

http://developer.mozilla.org/

Protocol = http, https

https://en.wikipedia.org/wiki/URL

Path ?

The path of the **resource** to fetch from URL

http://developer.mozilla.org/

Domain name

https://en.wikipedia.org/wiki/Domain_name

Path ?

The path of the **resource** to fetch from URL

http://developer.mozilla.org:80/

TCP port = 80 (default)

https://en.wikipedia.org/wiki/Domain_name

Path ?

The path of the **resource** to fetch from URL

http://developer.mozilla.org:80/

Path = / = resource to access/use

https://en.wikipedia.org/wiki/Domain_name

Resources ?

Employee
Product
Message
Department

Example

HTTP method	Path	Description
GET	/product	Retrieve all product
GET	/product/1	Retrieve product detail of 1
POST	/product	Create new product
PUT	/product/1	Update existing product 1
DELETE	/product/1	Delete product 1

More complexity

/product/1 or **/product/?id=1**

Path variables Query parameters

More complexity

/product/1 or **/product/?id=1**

Path variables Query parameters

Path variable
vs
Query parameter

Recommendation

Path variables

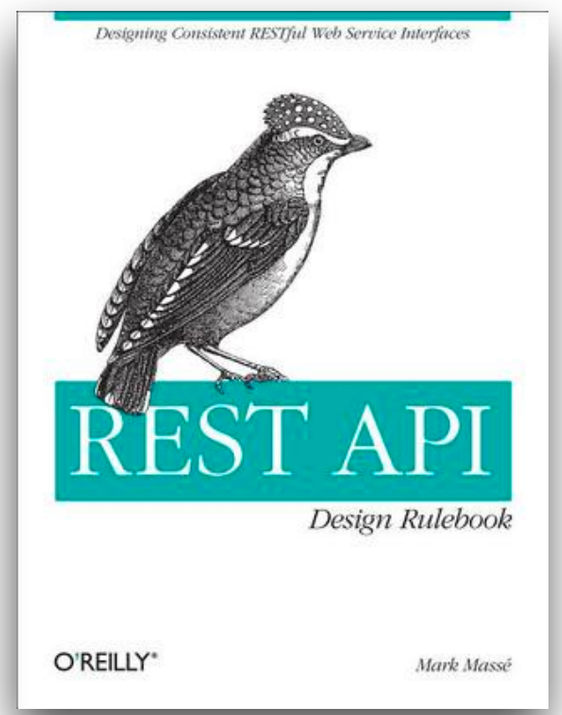
Used to **identify** from resource directly

Query parameters

Used to **sort** or **filter** from resource

https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9

REST API
Design rules



Use / for hierarchy data

Country => province

http://api/**countries**/thailand/**provinces**

Naming

Using **hyphen (-)** over underscore (_)

Using **lowercase**

http://api.com/blog/**hello-my-world**

Workshop

HTTP Response

HTTP Response ?

HTTP version
HTTP status code
HTTP response header
HTTP body (optional)

```
1  HTTP/1.1 200 OK
2  Date: Sat, 09 Oct 2010 14:28:02 GMT
3  Server: Apache
4  Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
5  ETag: "51142bc1-7449-479b075b2891b"
6  Accept-Ranges: bytes
7  Content-Length: 29769
8  Content-Type: text/html
9
10 <!DOCTYPE html... (here comes the 29769 bytes of the requ
```

HTTP Response ?

```
1  HTTP/1.1 200 OK
2  Date: Sat, 09 Oct 2010 14:28:02 GMT
3  Server: Apache
4  Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
5  ETag: "51142bc1-7449-479b075b2891b"
6  Accept-Ranges: bytes
7  Content-Length: 29769
8  Content-Type: text/html
9
10 <!DOCTYPE html... (here comes the 29769 bytes of the requ
```

HTTP version = 1.1

HTTP Response ?

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requ
```

HTTP status code = 200

HTTP Response ?

```
1  HTTP/1.1 200 OK
2  Date: Sat, 09 Oct 2010 14:28:02 GMT
3  Server: Apache
4  Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
5  ETag: "51142bc1-7449-479b075b2891b"
6  Accept-Ranges: bytes
7  Content-Length: 29769
8  Content-Type: text/html
9
10 <!DOCTYPE html... (here comes the 29769 bytes of the requ
```

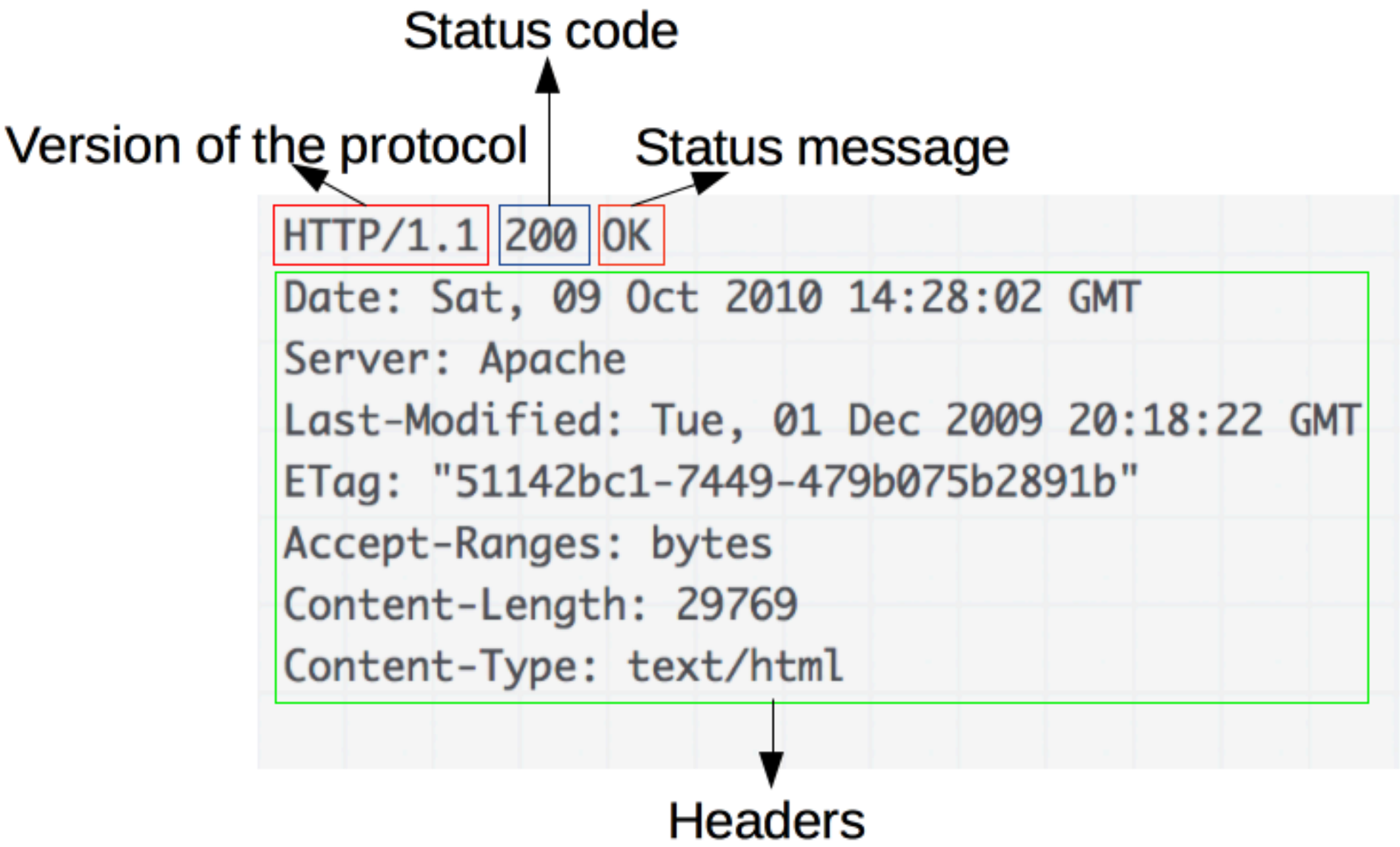
HTTP response header

HTTP Response ?

```
1  HTTP/1.1 200 OK
2  Date: Sat, 09 Oct 2010 14:28:02 GMT
3  Server: Apache
4  Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
5  ETag: "51142bc1-7449-479b075b2891b"
6  Accept-Ranges: bytes
7  Content-Length: 29769
8  Content-Type: text/html
9
10 <!DOCTYPE html... (here comes the 29769 bytes of the requ
```

HTTP body

HTTP Response ?

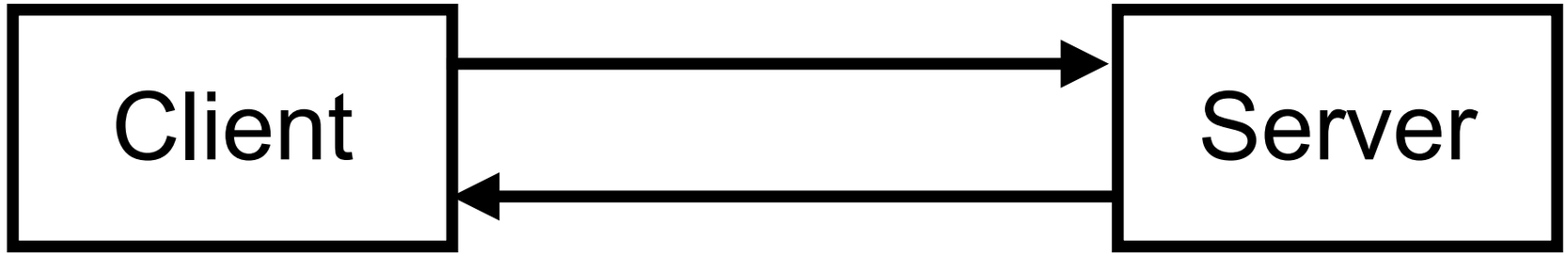


HTTP Status Code

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

HTTP Status Code

HTTP response status codes indicate whether a specific HTTP request has been successfully completed.



HTTP Response ?

5 groups of status codes

Code	Name
100-199	Information responses
200-299	Successful responses
300-399	Redirects
400-499	Client errors
500-599	Server error

Successful Responses

Code	Name
200	OK
201	Created
202	Accepted

Redirection message

Code	Name
301	Move permanently
302	Found, Move temporarily

Client Errors Responses

Code	Name
400	Bad request
401	Bad authorized
403	Forbidden
404	Not found
405	Method not allowed

Server Errors Responses

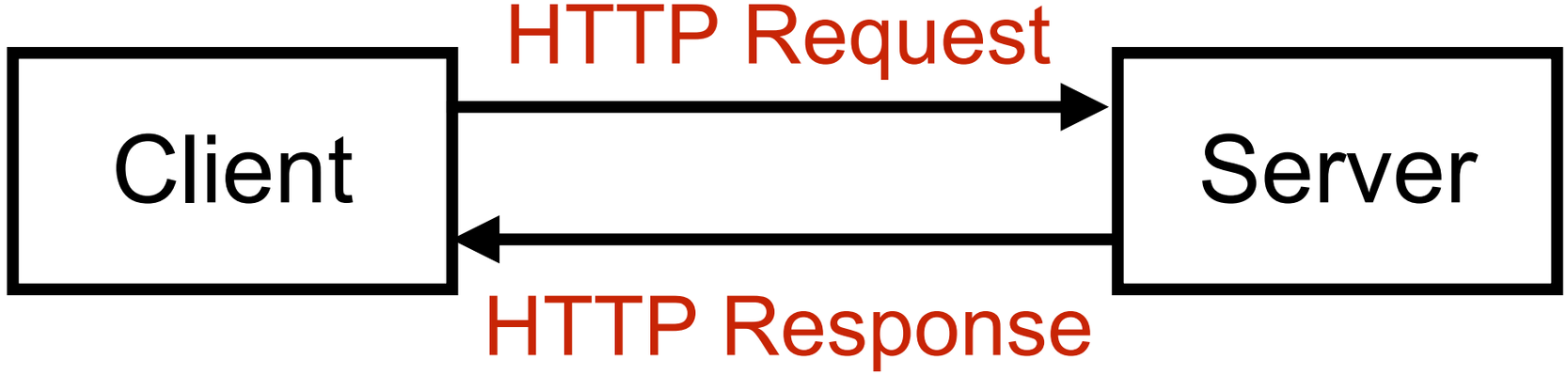
Code	Name
500	Internal server error
502	Bad gateway
503	Service unavailable
504	Gateway timeout

HTTP Body

HTTP Body

Information’s format of request and response body

Plain text, XML, **JSON**, Binary … etc



What is JSON ?

Java**S**cript **O**bject **N**otation

Lightweight format to store and transport data

Self-descriptive and easy to understand

https://www.json.org/json-en.html

Example of JSON message

List of products

```
{
    "products": [
        { "id": 1, "name": "name 1", "price": 1.00},
        { "id": 2, "name": "name 2", "price": 2.00},
        { "id": 3, "name": "name 3", "price": 3.00},
        { "id": 4, "name": "name 4", "price": 4.00},
        { "id": 5, "name": "name 5", "price": 5.00},
    ]
}
```

Example of JSON message

```
{
    "products": [
        { "id": 1, "name": "name 1", "price": 1.00},
        { "id": 2, "name": "name 2", "price": 2.00},
        { "id": 3, "name": "name 3", "price": 3.00},
        { "id": 4, "name": "name 4", "price": 4.00},
        { "id": 5, "name": "name 5", "price": 5.00},
    ]
}
```

{ } = Object

Example of JSON message

```
{
    "products": [
        { "id": 1, "name": "name 1", "price": 1.00},
        { "id": 2, "name": "name 2", "price": 2.00},
        { "id": 3, "name": "name 3", "price": 3.00},
        { "id": 4, "name": "name 4", "price": 4.00},
        { "id": 5, "name": "name 5", "price": 5.00},
    ]
}
```

[ ] = List or Array

Example of JSON message

Key

```
{
    "products": [
        { "id": 1, "name": "name 1", "price": 1.00},
        { "id": 2, "name": "name 2", "price": 2.00},
        { "id": 3, "name": "name 3", "price": 3.00},
        { "id": 4, "name": "name 4", "price": 4.00},
        { "id": 5, "name": "name 5", "price": 5.00},
    ]
}
```

Example of JSON message

Key

Value

```
{
    "products": [
        { "id": 1, "name": "name 1", "price": 1.00},
        { "id": 2, "name": "name 2", "price": 2.00},
        { "id": 3, "name": "name 3", "price": 3.00},
        { "id": 4, "name": "name 4", "price": 4.00},
        { "id": 5, "name": "name 5", "price": 5.00},
    ]
}
```

JSON syntax rules

Data is in name/value pairs

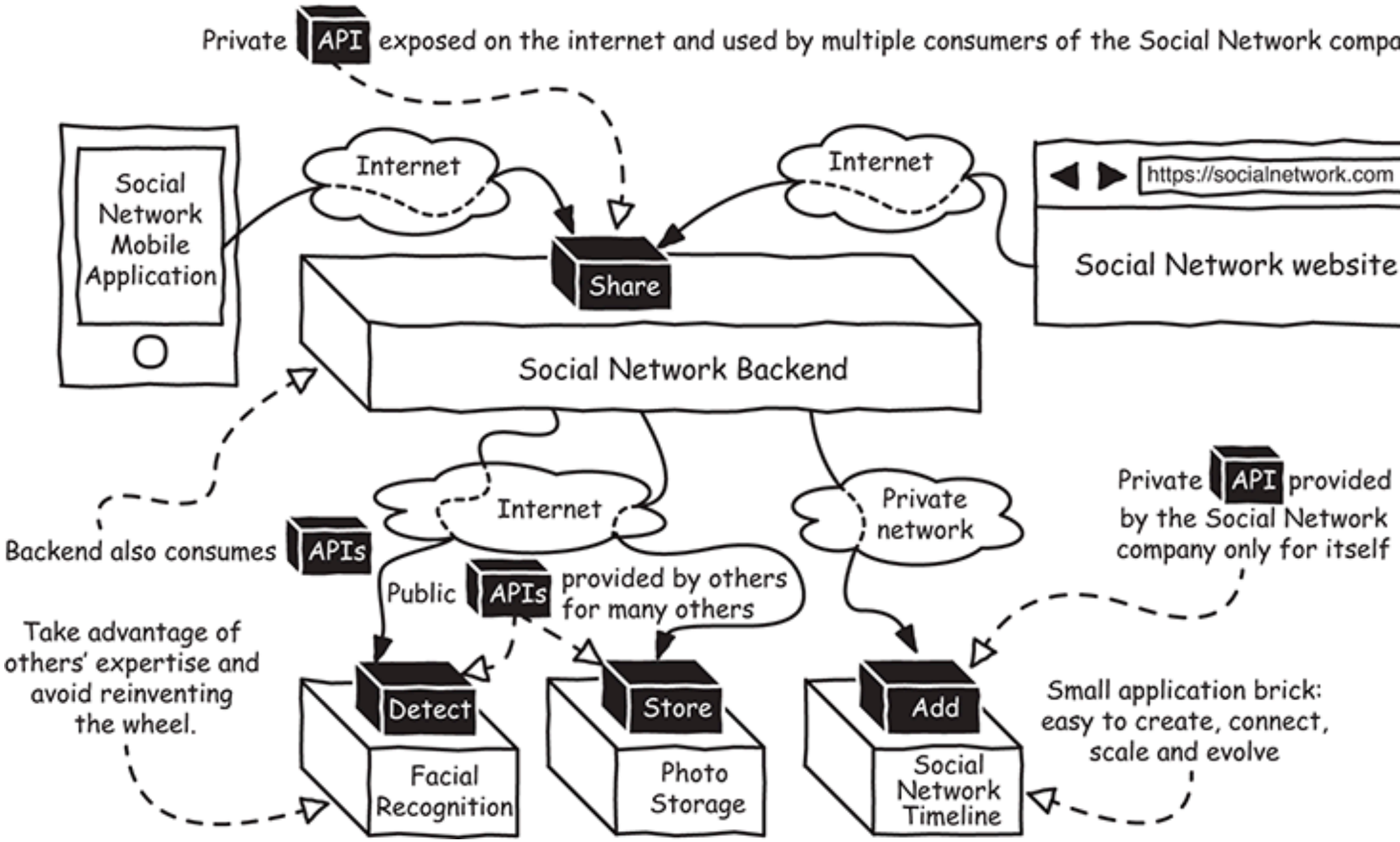
Data is separated by comma (,)

Curly braces ({}) holds objects

Square braces ([]) holds arrays/lists

Workshop

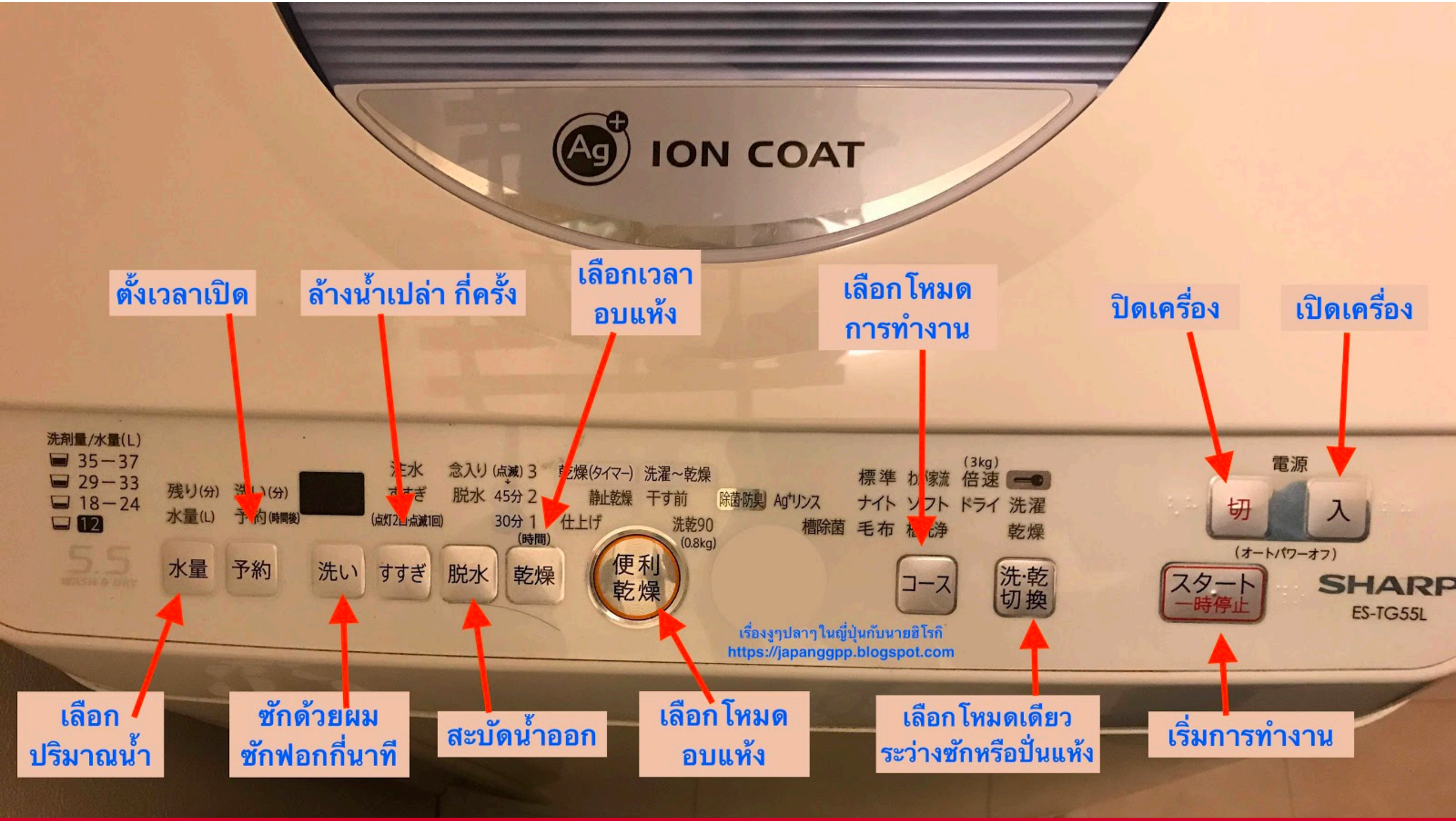
Complexity in real world



Need API design

Customer-First !!

Customer-First !!

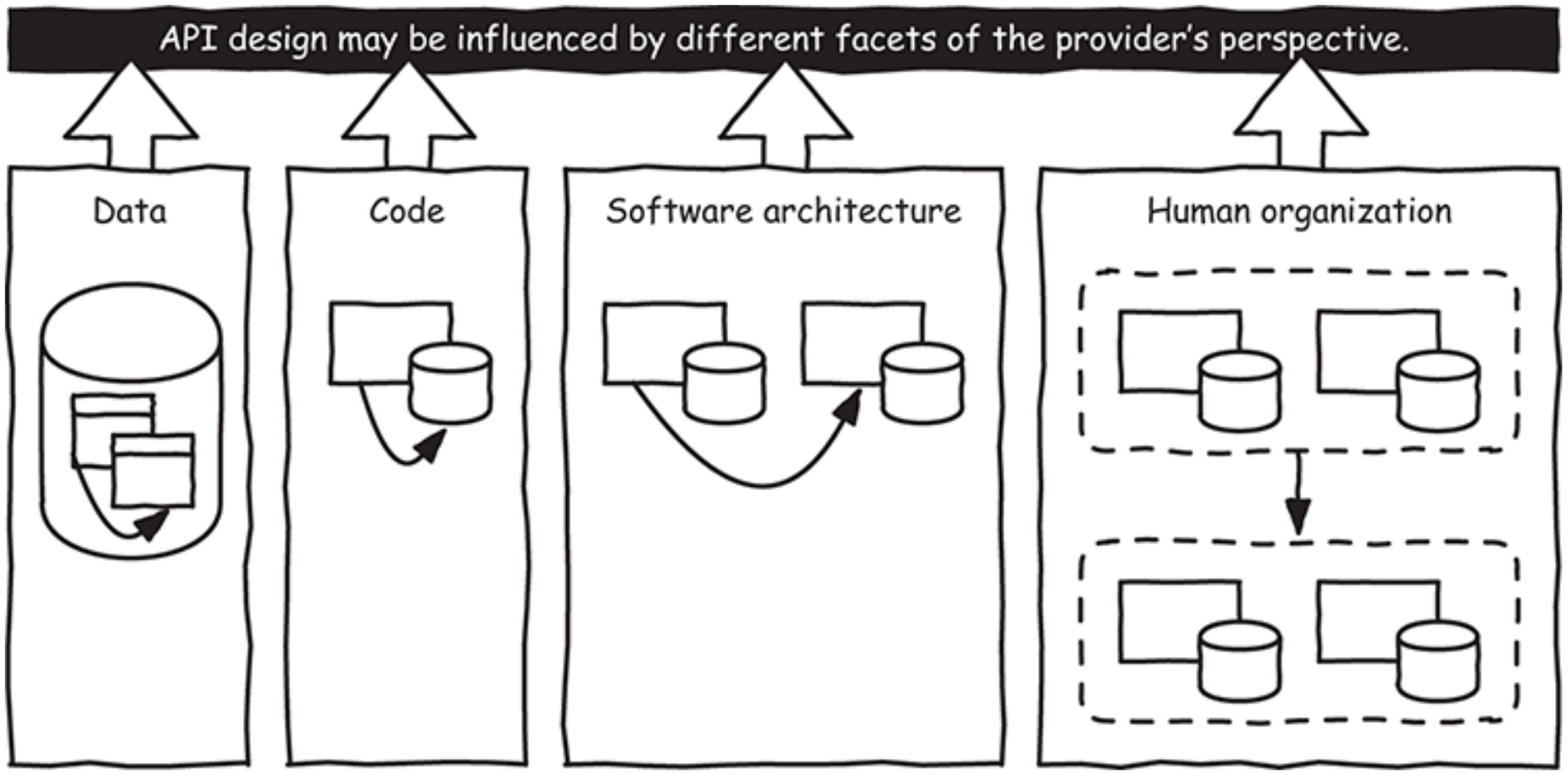


Provider vs Consumer View

How to design APIs ?

Identify API's goals

Avoid provider's view when designing



Identify API's Goals

Identify API's Goals

Who can use the API ?

What they can do ?

How they do it ?

What they need to do it ?

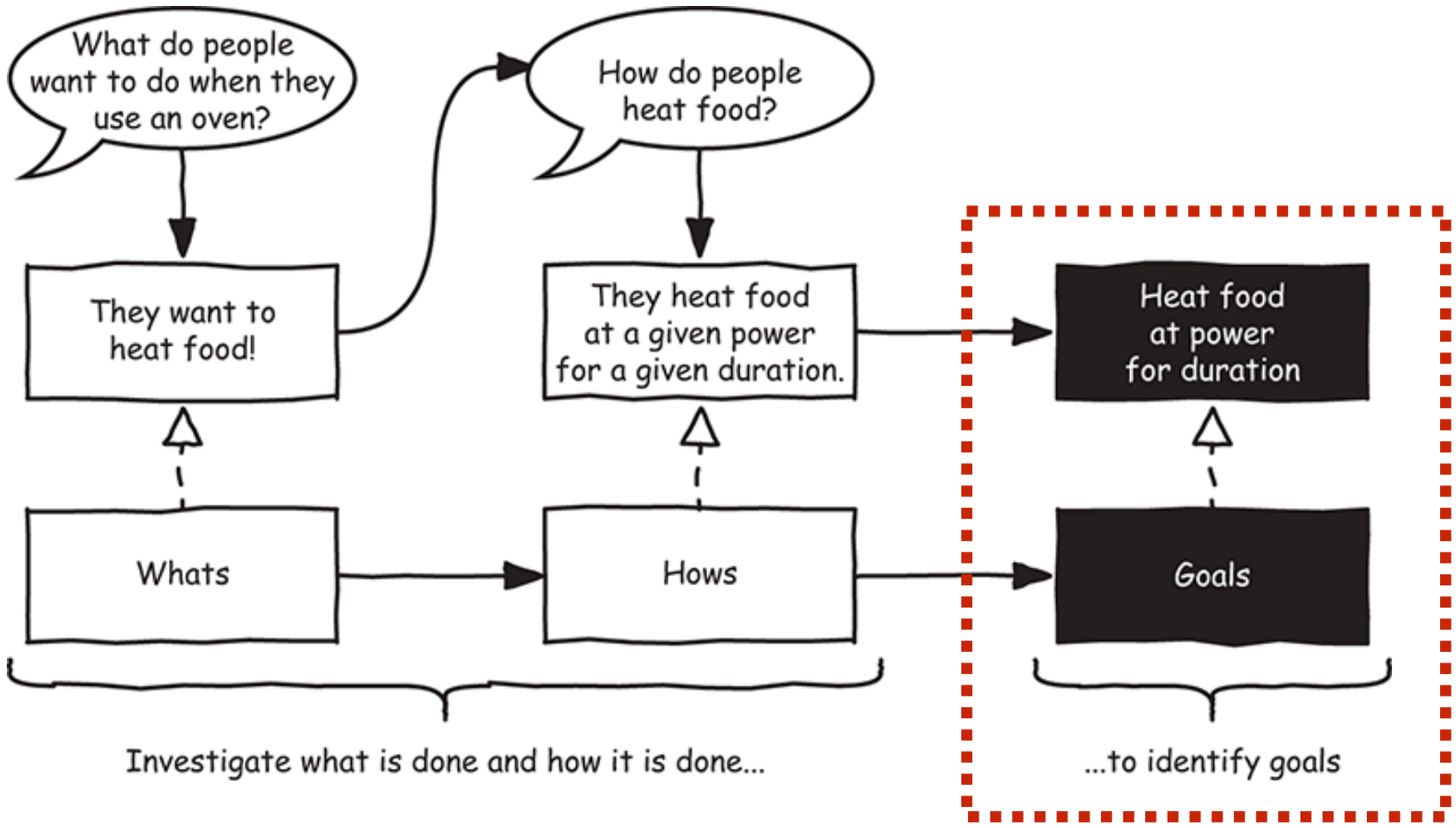
What they get in return ?

Let’s start

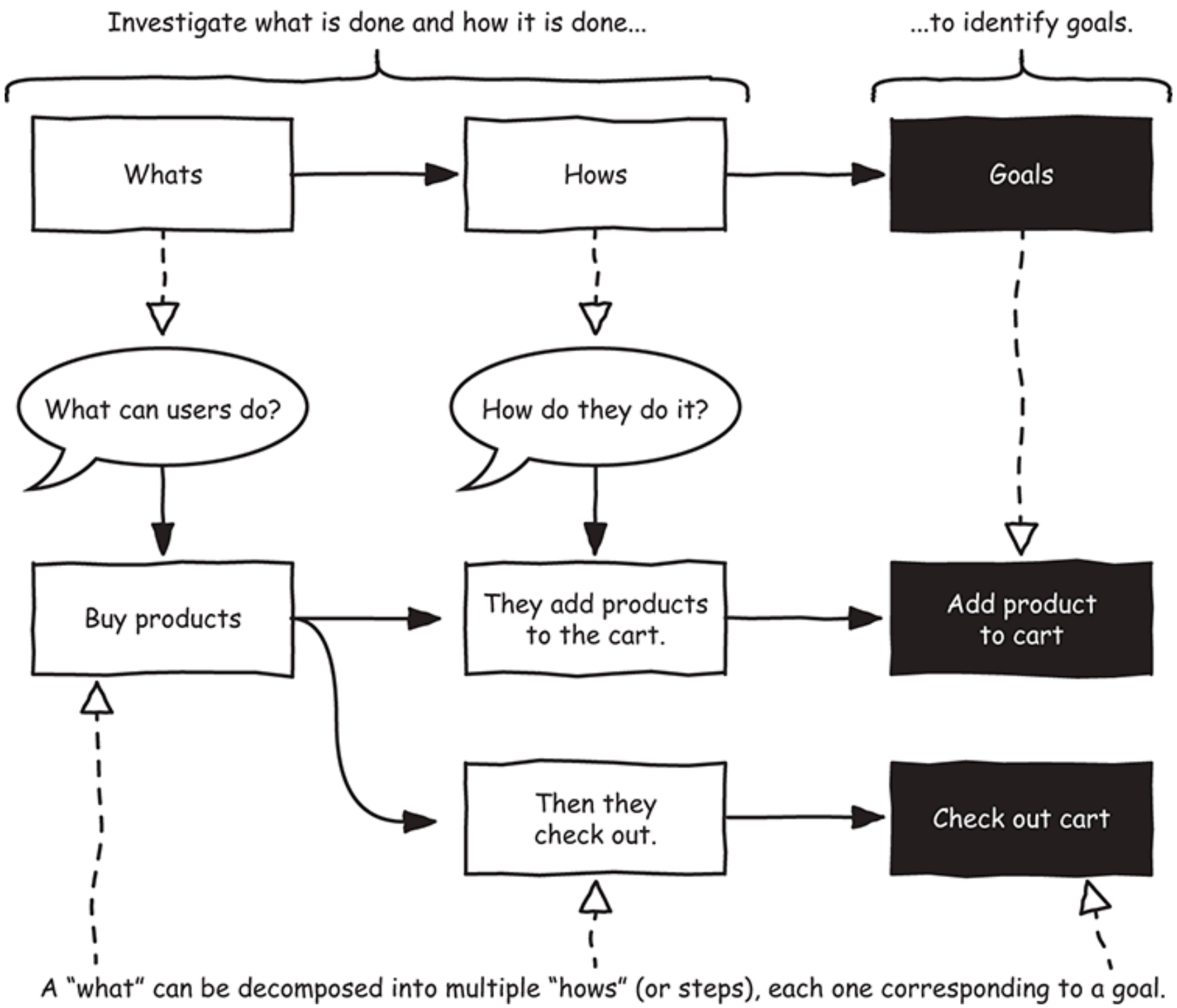
1. What and How ?

What can users do ?

How they do it ?



User buy a product

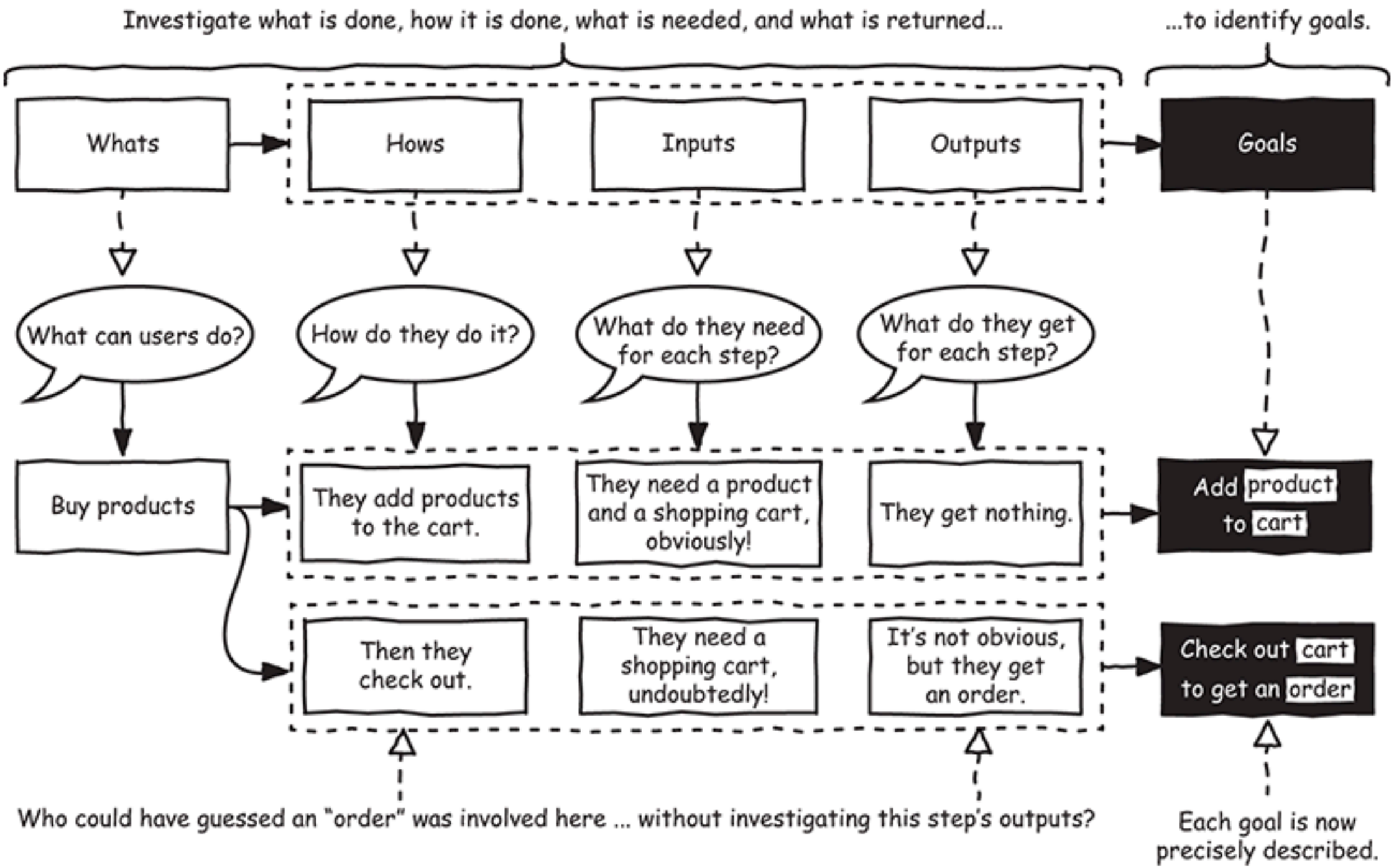


2. Input and Output ?

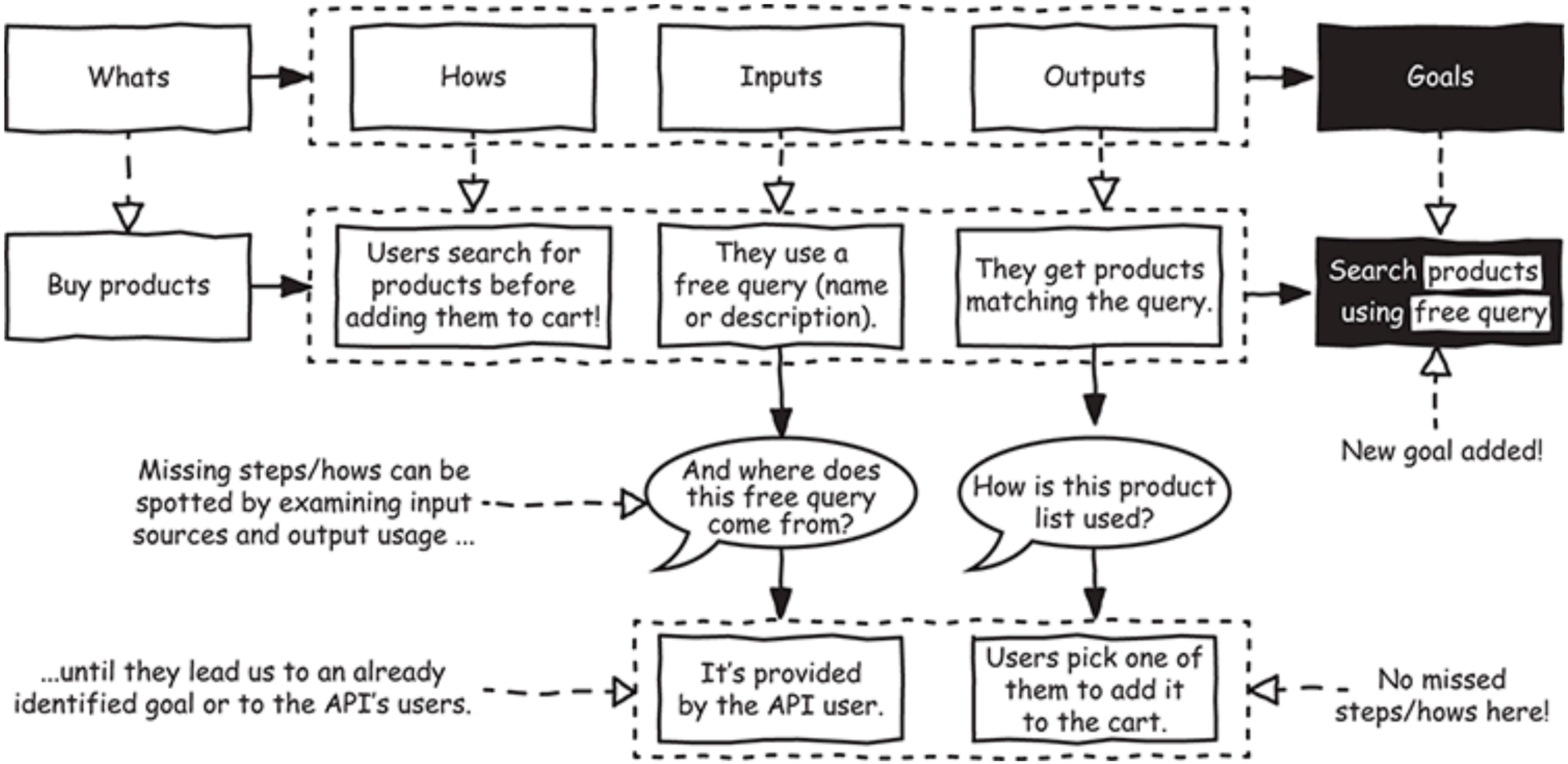
What they need to do it ?

What they get in return ?

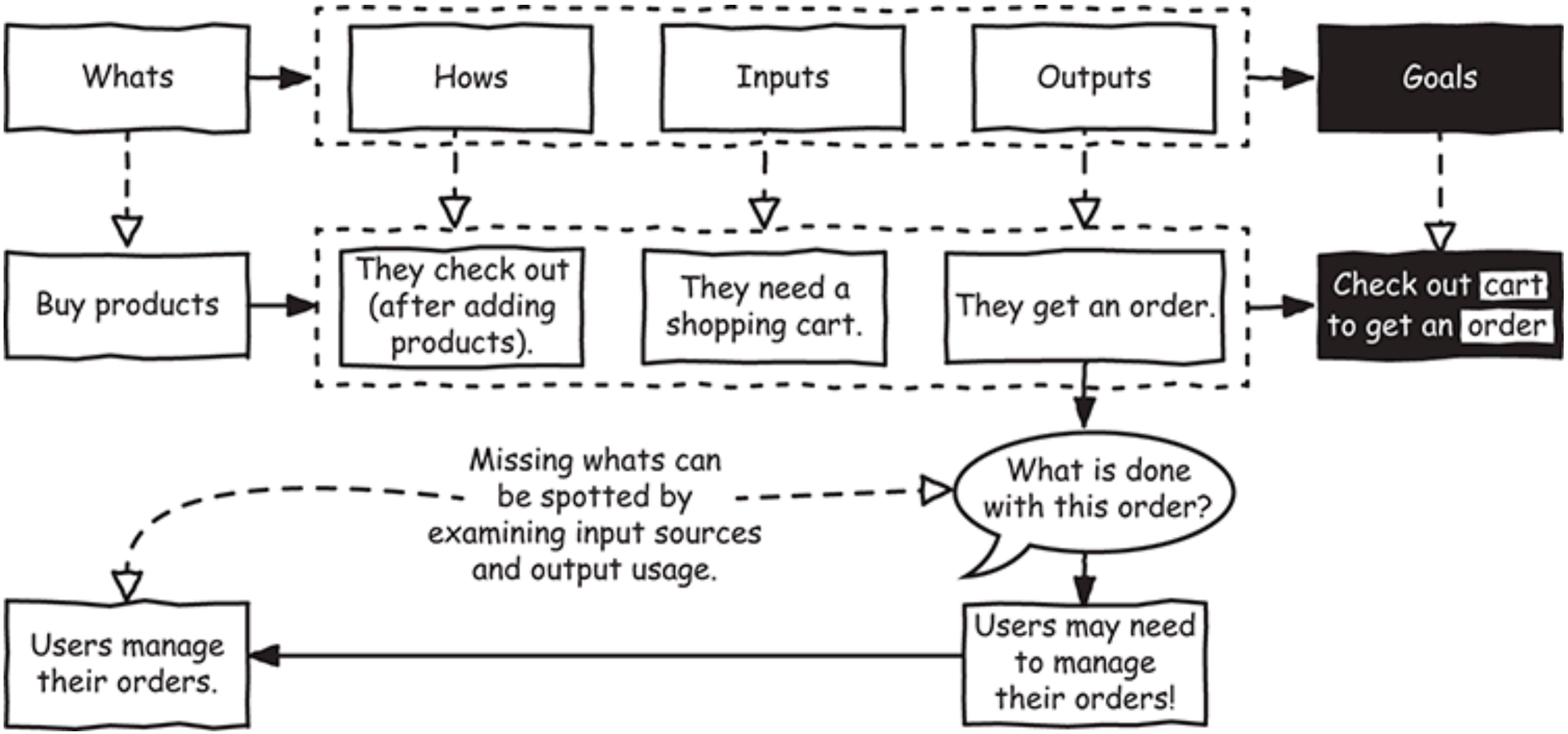
2. Input and Output ?



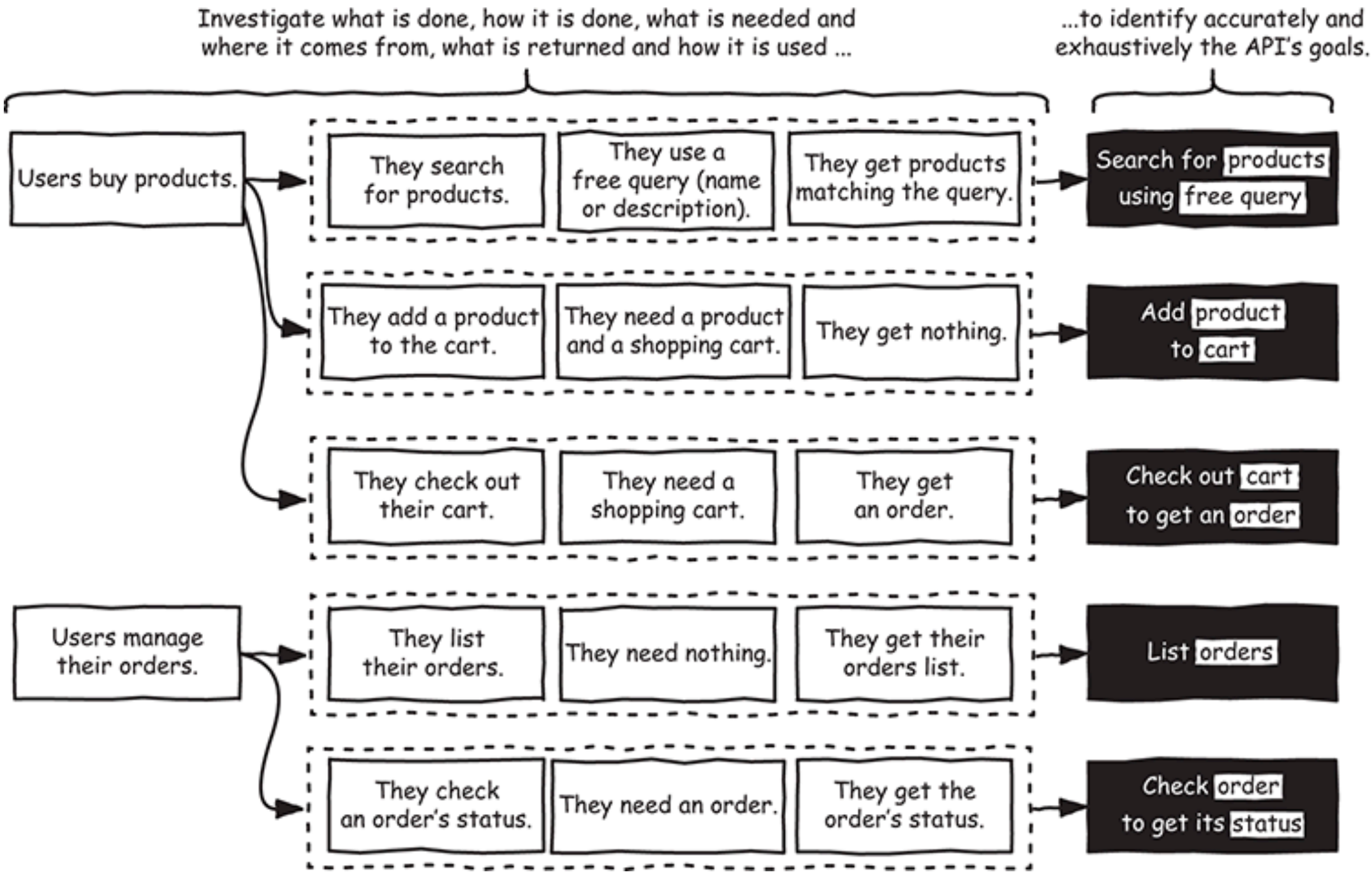
3. Find missing goals ?



3. Find missing goals ?

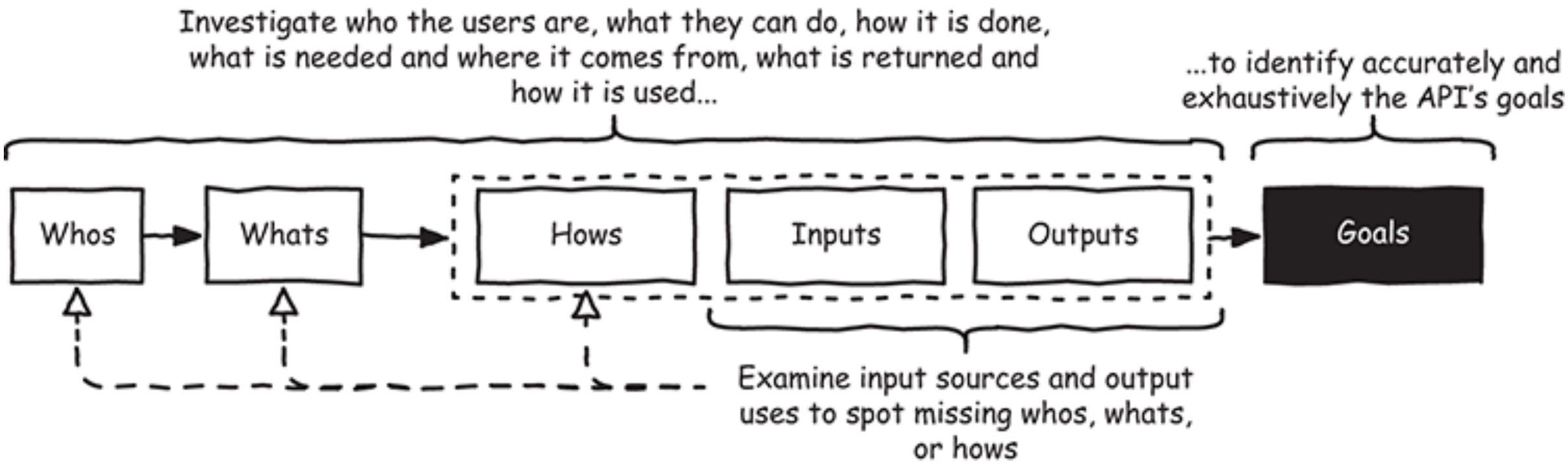


Add to Goals



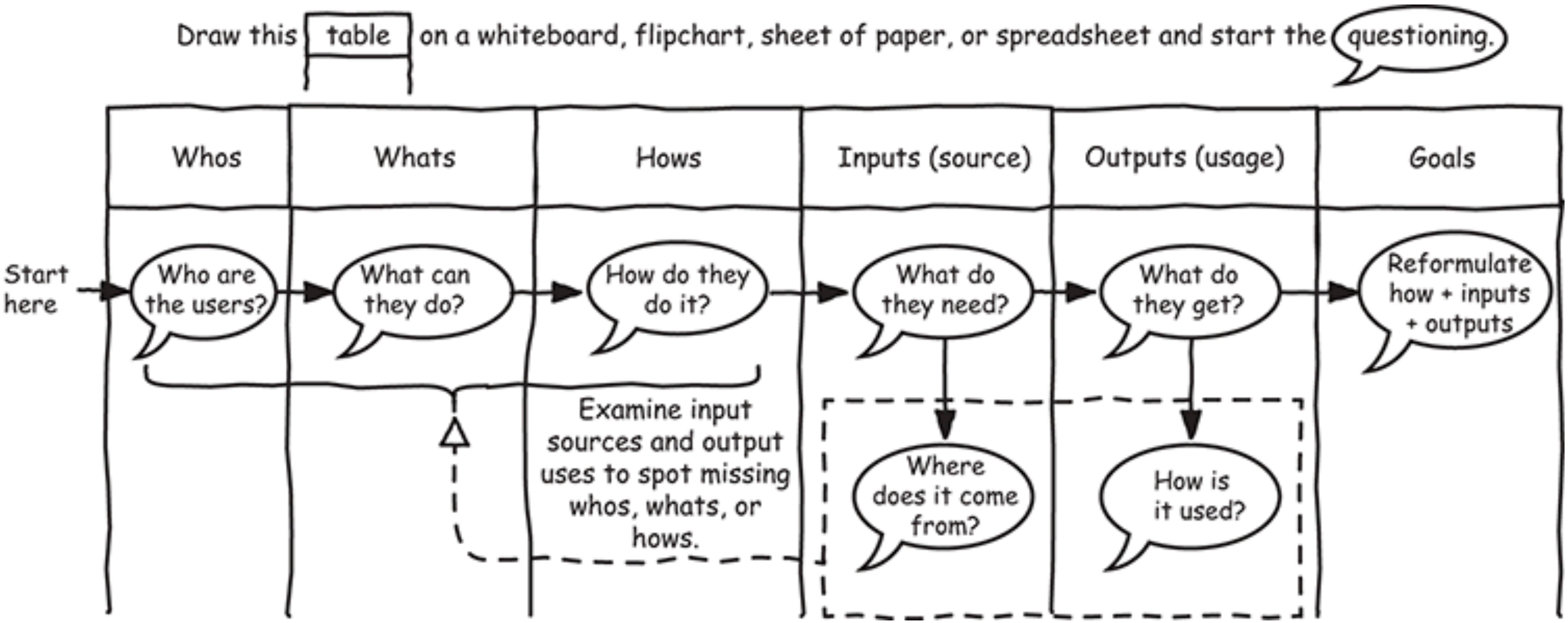
4. Identify all users (Who)

Who are the users ?



API Goals Canvas

API Goals Canvas



Example

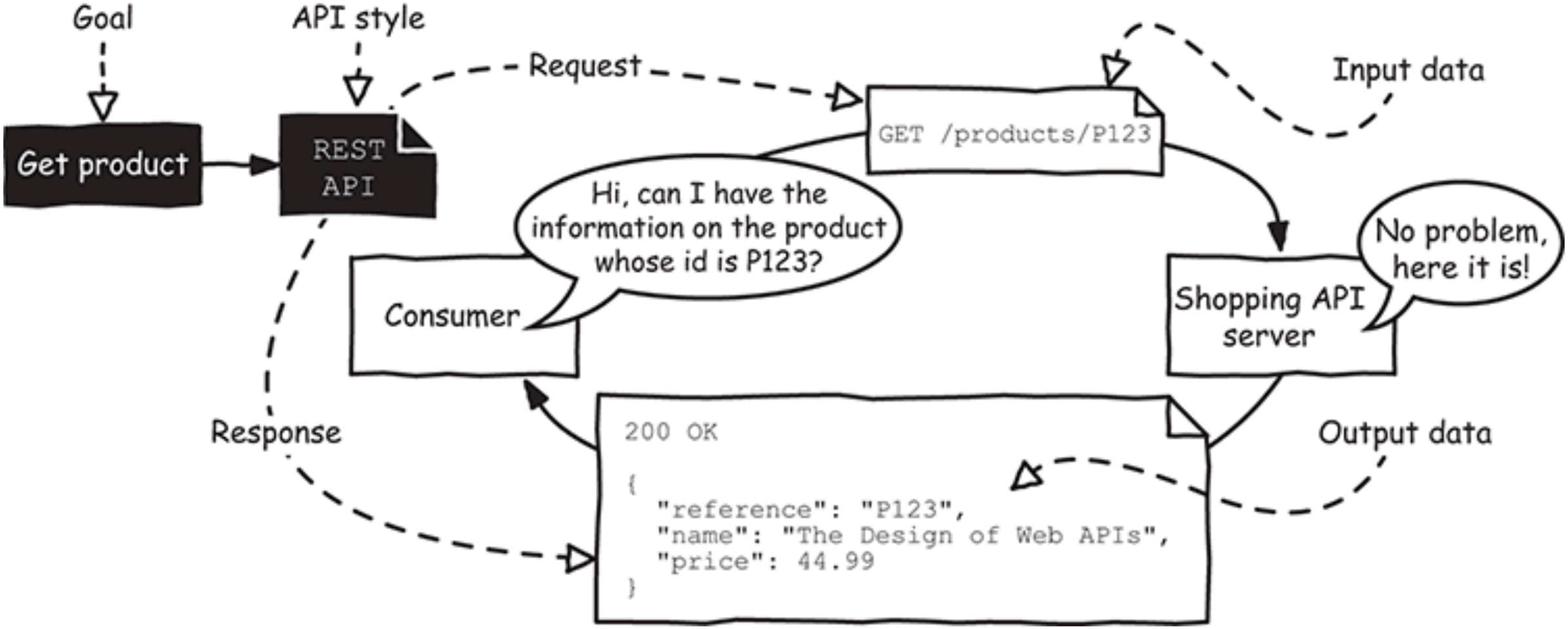
Whos	Whats	Hows	Inputs (source)	Outputs (usage)	Goals
Customers	Buy products	Search for products	Catalog (manage catalog), free query (provided by user)	Products (add product to cart)	Search for products in catalog using free query
		Add product to cart	Product (search for products), cart (owned by user)		Add product to cart
Admin	Manage catalog	Add product to catalog	Catalog (owned by user), product (provided by user)		Add product to catalog

Workshop

Next Step
Design Programming Interface

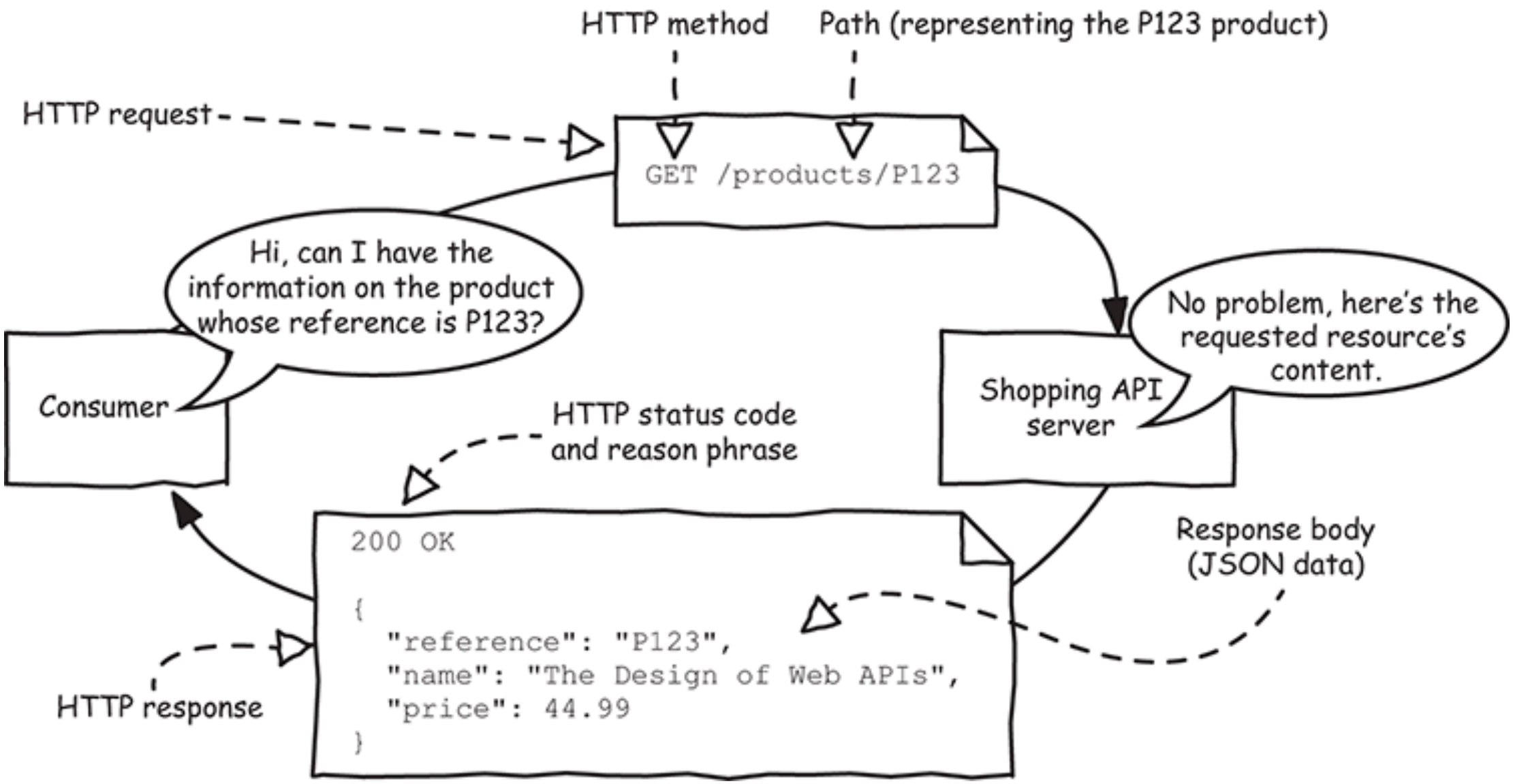
Example

Get product detail



Example

Get product detail



Next Step
Develop + Testing + Deploy