



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

Understanding and using APIs!!!

Network Programmability
(ACCB5029)

Applied Computing - WISA

Previously,

- ✓ Overview, Linux and Python Skills Review
- ✓ Devnet Developer Environment
- ✓ Software Development & Design
- ✓ Implement software version control using GIT
- ✓ Methods, Functions, Modules and Classes
- ✓ Data Formats

This week,

- ✓ Introducing APIs, its Design, Architecture styles,
- ✓ REST APIs

Previous week's Tutorial

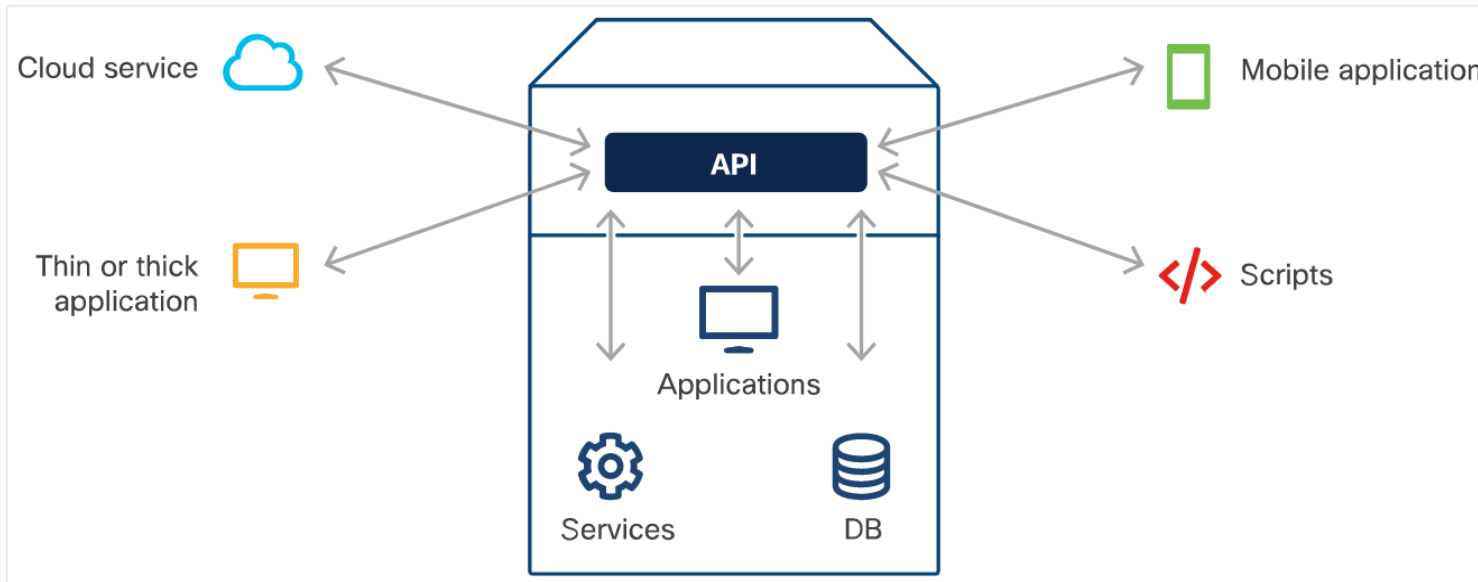
- Setup Lab environments, Review Linux and Python, PIP, and Python virtual environments, Version Control Systems, OOPS, Code Review and Testing.

Previously, we discussed about Devnet skills, and course work briefing. We discussed Software Development and Design. I hope you had a clarity on Software Development life cycle and its development methodologies. Followed by discussion on version control systems. The last week's discussion on importance of clean coding, Object oriented programming, Unit Testing must put you in a better platform to understand today's discussion on APIs, its designs, its use, architectural styles etc.

- ✓ This week's Tutorial and Independent learning
 - **Lab 5: Working with APIs**
 - **Lab 6: Integrating REST API in a Python Application**

API

- An **Application Programming Interface (API)** allows **one piece of software talk to another**.
- It uses common web-based interactions or communication protocols and its own proprietary standards.
- An **API determines** what **type of data, services, and functionality** the application exposes to third parties.
- By providing APIs, applications can control what they expose in a secure way.



- APIs have existed for decades, but **exposure and consumption of APIs has grown exponentially** in the last 10 years or so.
- Most modern APIs are designed into the product and are thoroughly tested.
- Simplified **coding languages such as Python** have made it possible for **non-software engineers** to **build applications and consume APIs**.

Why APIs

- APIs are built to be **consumed programmatically by other applications** and they can also be **used by humans who want to interact with the application manually**.
- Use cases of APIs are as follows:
 - **Automation tasks** – Build a script that performs manual tasks automatically and programmatically.
 - **Data integration** – An application can consume or react to data provided by another application.
 - **Functionality** – An application can integrate another application's functionality into its product.

API Design Styles

- A product's set of APIs may consist of both **synchronous and asynchronous designs**, where each API's design is independent of the others.
- The application consuming the API manages the response differently depending on the API design.

Synchronous APIs respond to a request directly by providing data immediately.

When are APIs synchronous?

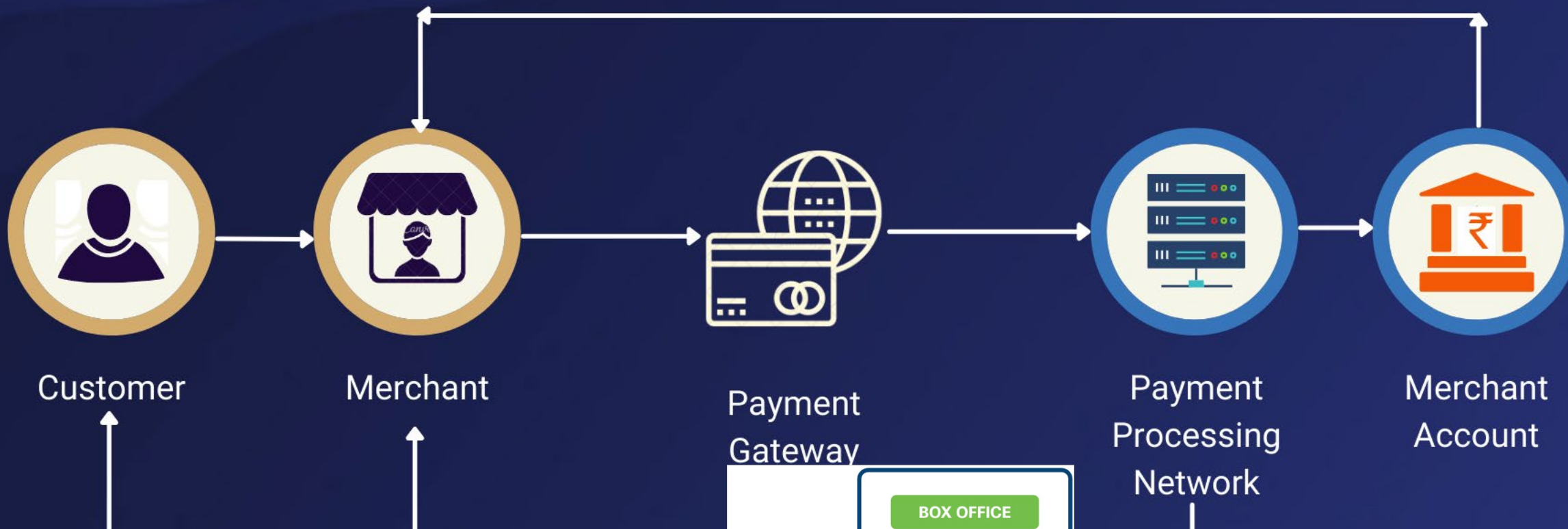
- APIs are synchronous when the data for the request is readily available.

Benefits of a synchronous API design

- Synchronous APIs enable the application to receive data immediately. If the API is designed correctly, the application performance will be better.

Client-side processing

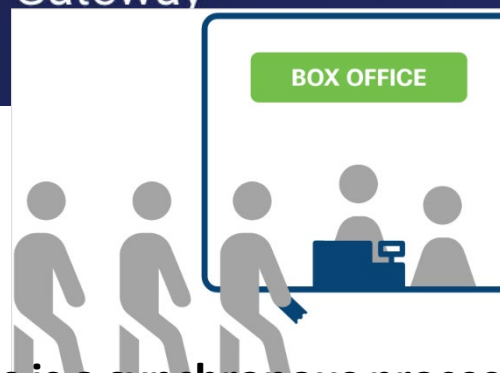
- The application that is making the API request must wait for the response before performing any additional code execution tasks



<https://www.lyra.com/in/everything-you-need-to-know-about-card-authorization/>

API Design Styles

Tickets are sold in first-come, first served order. This is a synchronous process.



Synchronous API design

API Design Styles

- A product's set of APIs may consist of both **synchronous and asynchronous designs**, where each API's design is independent of the others.
- The application consuming the API manages the response differently depending on the API design.

Asynchronous APIs provide a response (with no data) to signify that the request has been received.

When are APIs asynchronous?

- APIs are asynchronous when the request takes some time for the server to process or if data isn't readily available.

Benefit of asynchronous API design

- Asynchronous APIs allow the application to continue execution without being blocked till the server processes the request, thus resulting in better performance.

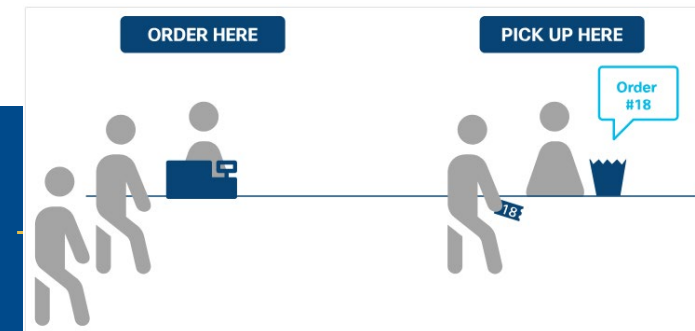
Client-side processing

- With asynchronous processing, the design of the API on the server side defines the requirement on the client side.



API Design Styles

Asynchronous API design

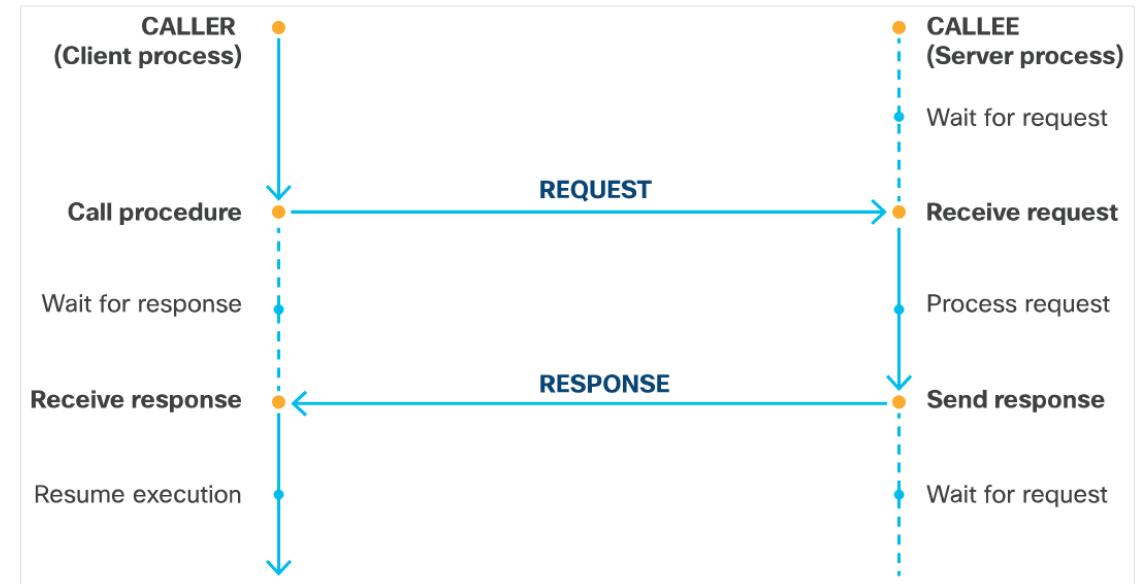


API Architectural Styles

There are certain standards, protocols, and specific architectural styles which make it easier for consumers of the API to learn and understand the API.

Simple Object Access Protocol (SOAP) is an XML-based messaging protocol. It is used for communication between applications on different platforms or built with different programming languages.

- ✓ **Independent:** *All applications can communicate with each other and run on different operating systems*
- ✓ **Extensible:** *Add features like reliability & security*
- ✓ **Neutral:** *Can be used over any protocol, including HTTP, SMTP, TCP, UDP, or JMS*



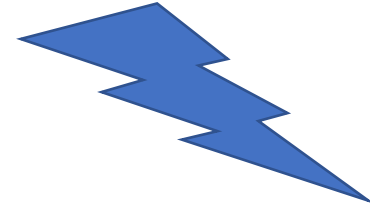
Remote Procedure Call (RPC) is a **request-response model** that allows an application to make a procedure call to another application.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Query request too large.</faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

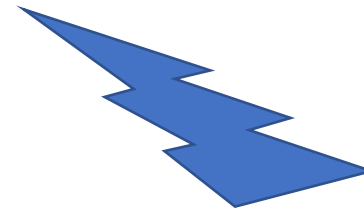
API Architectural Styles

Simple Object Access Protocol (SOAP) is an XML- based messaging protocol. It is used for communication between applications on different platforms or built with different programming languages.

- Used in Client – Server Architectures
- Make calls to remote procedures/ functions over a network
- Uses protocols like HTTP, TCP.
- Comparatively more lightweight, faster and simpler



- Exchange structure data between web services
- Heavily rely on XML Schema to communicate
- Used in enterprise level applications that require additional features like security etc.

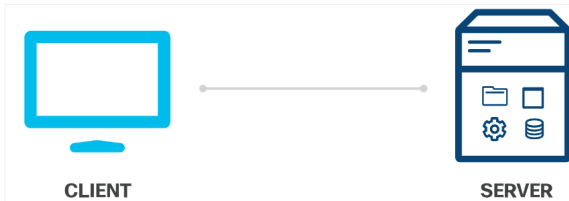


Remote Procedure Call (RPC) is a **request-response model** that allows an application to make a procedure call to another application.

API Architectural Styles

REpresentational State Transfer (REST) is an architectural style authored by Roy Thomas Fielding. He established six constraints that can be applied to any protocol in REST.

REST is scalable, simple and modifiable.

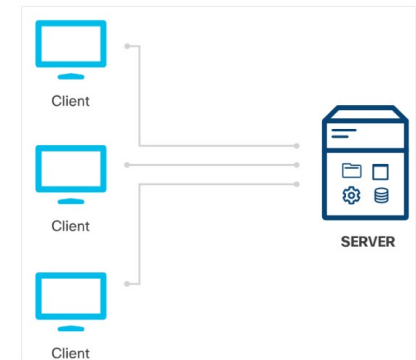


Client-server

- The client and server should be independent of each other.
- This will enable the client to be built for multiple platforms which will simplify the server-side components.

Stateless

- Requests from the client to the server must contain REST client-server model and all the information which the server needs to make the request.
- The server cannot contain session states.



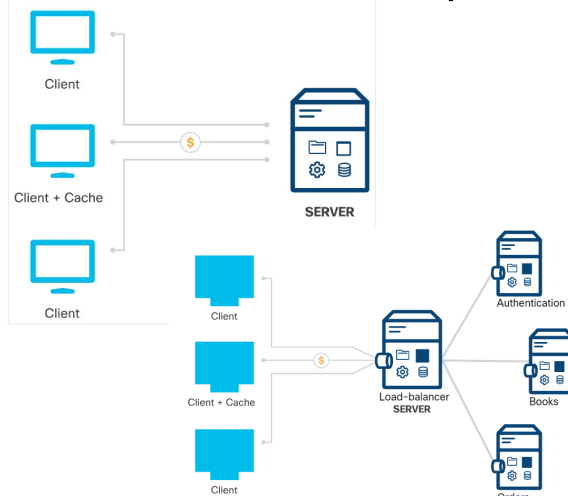
Uniform Interface: The interface between the client and the server adhere to the four principles

Identification of resources	Manipulation of resources through representations
Self-descriptive messages	Hypermedia as the engine of application state.

API Architectural Styles

REpresentational State Transfer (REST) is an architectural style authored by Roy Thomas Fielding. He established six constraints that can be applied to any protocol in REST.

REST is scalable, simple and modifiable.



Cache model:

- Responses from the server must state whether the response is cacheable or non-cacheable.
- If it is cacheable, the client can use the data from the response for later requests.

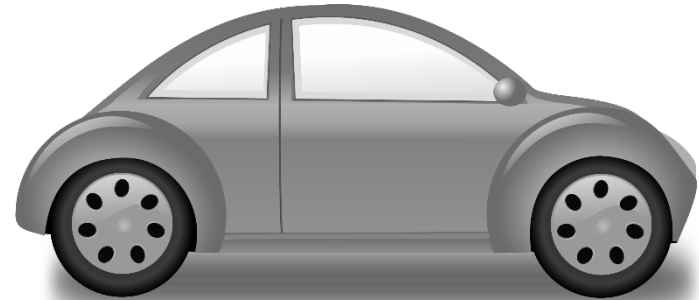
Layered system:

- The Layered system consists of different hierarchical layers in which each layer provides services only to the layer above it.
- As a result, it consumes services from the layer below.

Code-on-demand:

- The information returned by a REST service can include executable code (for example, javascript) or links intended to usefully extend client functionality.
- The constraint is optional because execution of third-party codes introduces potential security risks.

API: Design & Architecture Example



Which?



- Sports Car
- SUV
- Hatchback
- Etc

- SOAP
- REST
- RPC

How?



- Hire Purchase
- Personal Contract Purchase
- Etc

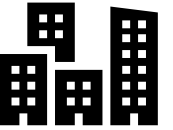
- Synchronous API design
- Asynchronous API design

What?



- Transportation (Self-Drive, Book a Cab, Walk, Cycle etc)
- Act (Keys/ Ice Scraper/ Book using an app/ call a taxi etc)
- Travel (Get in the car and reach the destination)

- Automation tasks
- Data integration
- Functionality



REST APIs

- A REST Web service API (**REST API**) is a ***programming interface that communicates over HTTP***.
- REST APIs use the same concepts as the HTTP protocol which are as follows:
 - HTTP requests/responses
 - HTTP verbs
 - HTTP status codes
 - HTTP headers/body
- REST API requests are **HTTP requests** that are a way for an application (client) to ask the server to perform a function.

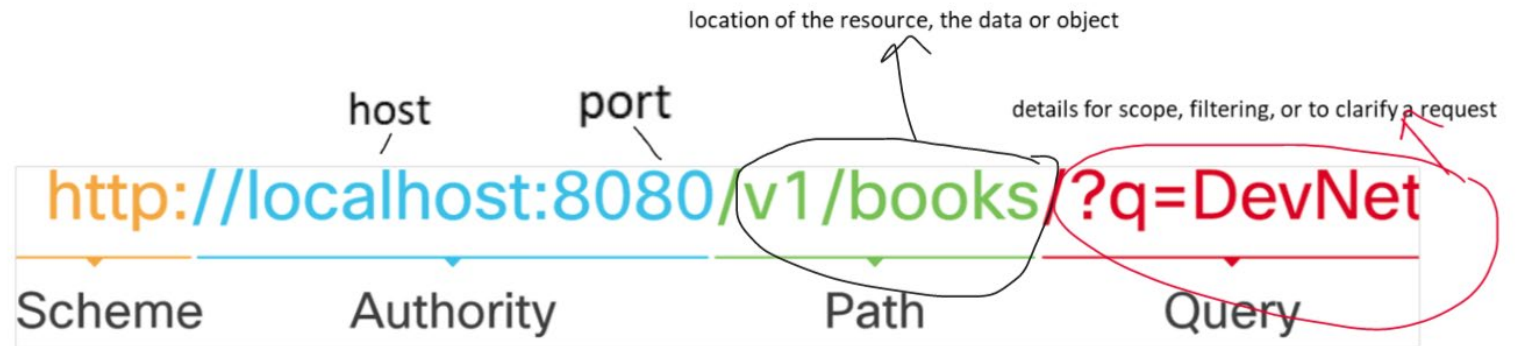


REST APIs - Request

- REST API requests are made up of four major components:

- **Uniform Resource Identifier (URI)**
- HTTP Method
- Header
- Body

The Uniform Resource Identifier (URI), also referred to as Uniform Resource Locator (URL), identifies which resource the client wants to manipulate.



REST APIs - Request

○ REST API requests are made up of four major components:

- **Uniform Resource Identifier (URI)**
- HTTP Method
- Header
- Body

The Uniform Resource Identifier (URI), also referred to as Uniform Resource Locator (URL), identifies which resource the client wants to manipulate.

REST APIs use the standard HTTP methods to communicate to the web services for which action is being requested for the given resource.

HTTP Method	Action	Description
POST	Create	Create a new object or resource.
GET	Read	Retrieve resource details from the system.
PUT	Update	Replace or update an existing resource.
PATCH	Partial Update	Update some details from an existing resource.
DELETE	Delete	Remove a resource from the system.

The body of the REST API request contains the data pertaining to the resource that the client wants to manipulate.

REST APIs - Request

- REST API requests are made up of four major components:
 - Uniform Resource Identifier (URI)
 - **HTTP Method**
 - Header
 - Body

REST APIs use the standard HTTP methods to communicate to the web services for which action is being requested for the given resource.

HTTP Method	Action	Description
POST	Create	Create a new object or resource.
GET	Read	Retrieve resource details from the system.
PUT	Update	Replace or update an existing resource.
PATCH	Partial Update	Update some details from an existing resource.
DELETE	Delete	Remove a resource from the system.

REST APIs - Request

- REST API requests are made up of four major components:

- Uniform Resource Identifier (URI)
- HTTP Method
- **Header**
- Body

HTTP headers are formatted as name-value pairs that are separated by a colon (:), [name]:[value].

- **Request headers** include additional information that does not relate to the content of the message.
 - + Like content type of the request, authorization information, and cache control directives.
- **Entity headers** include additional information describing the content and format of the body of the message.
 - + Content-Type, Content-Length, Content-Encoding, Last-Modified, ETag
- **Response headers** include additional information that describes the content of the body of the message.
 - + Like the content type of the response, cache control directives, and information about the server

REST APIs - Request

- REST API requests are made up of four major components:
 - Uniform Resource Identifier (URI)
 - HTTP Method
 - Header
 - **Body**
- The body of the REST API request contains the data pertaining to the resource that the client wants to manipulate.
- REST API requests that use the HTTP method POST, PUT, and PATCH typically include a body.
 - + The body is optional depending on the HTTP method.
 - + If the data is provided in the body, then the data type must be specified in the header using the Content-Type key.

REST APIs - Responses

- REST API responses are **HTTP responses that communicate the results of a client's HTTP request.**
- REST API Responses are made up of three major components:
 - **HTTP Status** The HTTP status code help the client **determine the reason for the error and can sometimes provide suggestions** for fixing the problem.
 - Header
 - Body HTTP status codes consists of three digits, where the first digit is the response category, and the other two digits are assigned in numerical order.

There are five different categories of HTTP status codes:

- + 1xx – Informational – for informational purposes, responses do not contain a body
- + 2xx – Success – the server received and accepted the request
- + 3xx – Redirection – the client has an additional action to take to get the request completed
- + 4xx -- Client Error – the request contains an error such as bad syntax or invalid input
- + 5xx -- Server Error – unable to fulfill the valid requests.

REST APIs - Responses

- REST API responses are **HTTP responses that communicate the results of a client's HTTP request.**
- REST API Responses are made up of three major components:

➤ **HTTP Status**

HTTP Status Code	Status Message	Description
200	Ok	Request was successfully and typically containing a payload (body)
201	Created	Request was fulfilled and the requested resource was created
202	Accepted	Request has been accepted for processing and is in process
400	Bad Request	Request will not be processed due to an error with the request
401	Unauthorized	Request does not have valid authentication credentials to perform the request
403	Forbidden	Request was understood but has been rejected by the server
404	Not Found	Request cannot be fulfilled because the resource path of the request was not found on the server
500	Internal Server Error	Request cannot be fulfilled due to a server error
503	Service Unavailable	Request cannot be fulfilled because currently the server cannot handle the request

REST APIs - Responses

- REST API responses are **HTTP responses that communicate the results of a client's** HTTP request.
- REST API Responses are made up of three major components:
 - HTTP Status
 - **Header**
The header in the response is to provide additional information between the server and the client in the name-value pair format that is separated by a colon (:), [name]:[value]
 - Body
- **Response headers** contains additional information that doesn't relate to the content of the message.
 - + Like set-cookie, Cache control etc
- **Entity headers** contains additional information that describes the content of the body of the message.
 - + Like Content-Type, Content-Length etc
- **Response Pagination** enables data to be broken into chunks.
- **Compressed response data** Compressed data reduces large amount of data that cannot be paginated

REST APIs - Authentication

REST APIs require authentication so that **random users cannot access, create, update, or delete information** incorrectly or maliciously. Authentication depends on the information dealt with. Uniform Resource Identifier (URI)

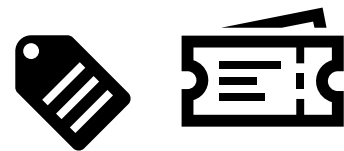
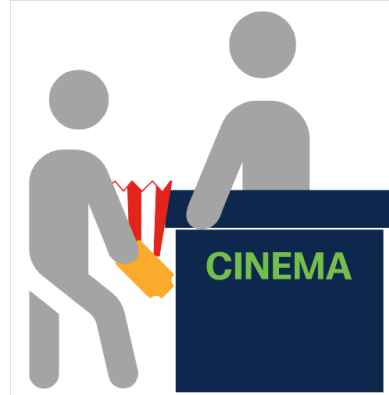
- **Authentication:** to prove the user's identity
 - + **Basic authentication:** It transmits credentials as username/password pairs separated with a colon (:) and encoded using Base64.
 - + **Bearer authentication:** It uses a bearer token, which is a string generated by an authentication server such as an Identity Service (IdS).
 - + **API Key:** It is a unique alphanumeric string generated by the server and assigned to a user. The two types of API keys are public and private.



REST APIs - Authentication

REST APIs require authentication so that **random users cannot access, create, update, or delete information** incorrectly or maliciously. Authentication depends on the information dealt with. Uniform Resource Identifier (URI)

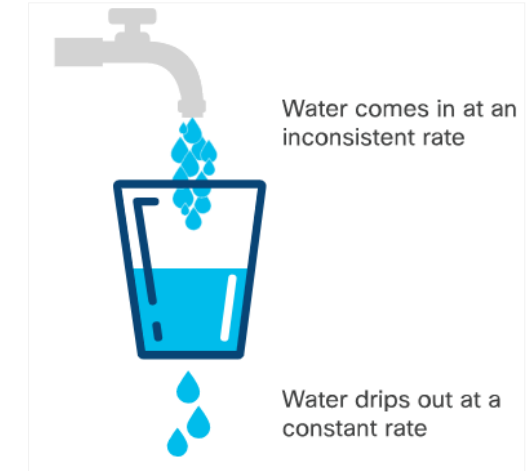
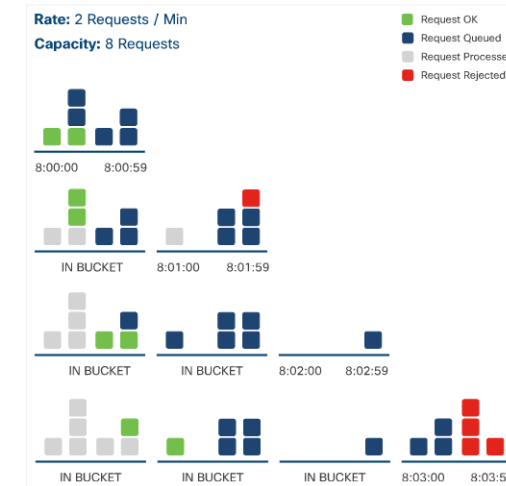
- **Authorization:** to define user access. It is the act where the user is proving to have permissions to perform the requested action on that resource.
 - + **Open Authorization (OAuth)** combines authentication with authorization.
 - + OAuth has increased security as compared to other options.
 - + There are two versions of OAuth - **OAuth 1.0 and OAuth 2.0**, where OAuth 2.0 is not backwards compatible.
 - + OAuth 2.0 authorization framework **enables a third-party application to obtain limited access** to an HTTP service.
 - + OAuth **allows the user to provide credentials directly to the authorization server** [Identity Provider (IdP) or an Identity Service (IdS)], to obtain an access token to share with the application.
 - + The **process of obtaining the token is called a flow**.



APIs – Rate Limits

An API rate limit is a way for a web service **to control the number of requests a user or an application can make per defined unit of time.**

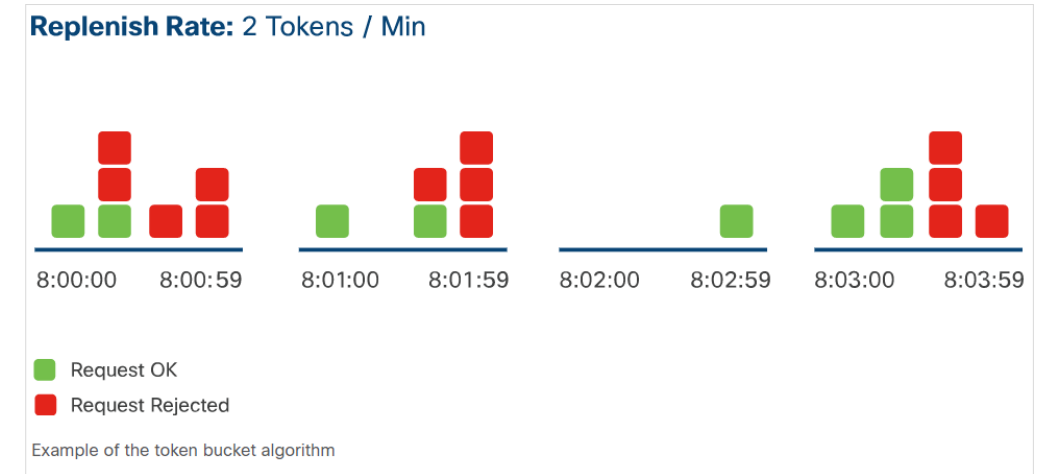
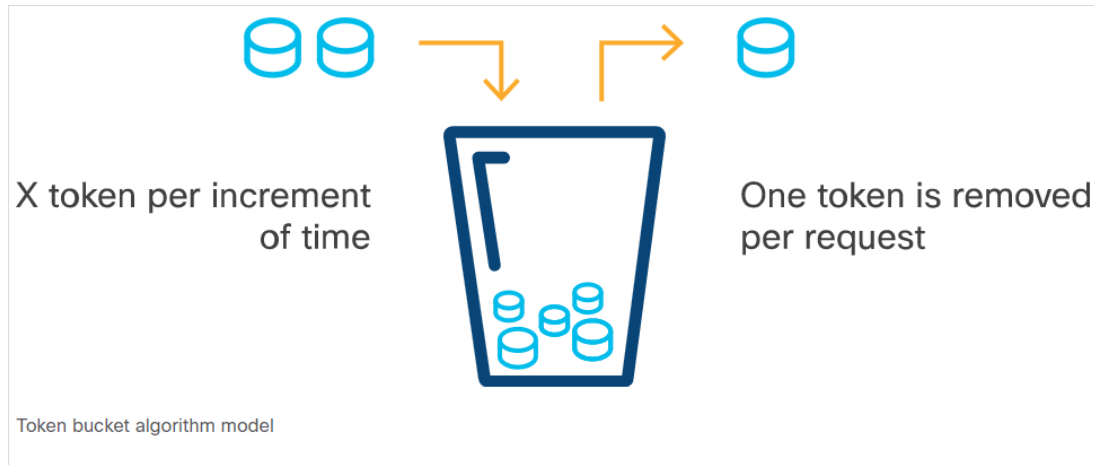
- **Rate limiting helps to :**
 - + avoid a server **overload from too many requests** at once
 - + provide **better service and response time** to all users
 - + protect against a **Denial-of-Service (DoS) attack**



- **Leaky bucket** algorithm puts all incoming requests into a request queue in the order in which they were received.
 - + The incoming requests can come in at any rate, but the server will process the requests from the queue at a fixed rate.
 - + If the request queue is full, the request is rejected

APIs – Rate Limits

An API rate limit is a way for a web service **to control the number of requests a user or an application can make per defined unit of time.**



- **Token bucket** algorithm gives each user a defined number of tokens they can use within a certain increment of time.
 - + When the client makes a request, the server checks the bucket to make sure it contains at least one token. If so, it removes that token and processes the request. If there isn't a token available, it rejects the request.
 - + The client must calculate the number of tokens he currently has to avoid rejected requests.

APIs – Rate Limits

- **Fixed window counter** algorithm, fixes a window time and assigns a counter to represent how many requests can be processed during that period.
- **Sliding window counter** algorithm allows a fixed number of requests to be made in a set duration of time.
- **Many rate limiting APIs add details about the rate limit in the response's header.**
- The commonly used Rate Limit keys include:
 - + X-Rate Limit-Limit - The maximum number of requests that can be made in a specified unit of time.
 - + X-Rate Limit-Remaining - The number of pending requests that the requester can make in the current rate limit window
 - + X-Rate Limit-Reset - The time when the rate limit window will reset.
- When the rate limit is exceeded, the server automatically rejects the request and sends back an HTTP response to the user.
- The most commonly used HTTP status codes are **429: Too Many Requests** or **403: Forbidden**.

APIs – Webhooks

- A Webhook is an HTTP callback, or an HTTP POST, to a specified URL that notifies the application when a particular activity or event is occurred in the resources.
 - + With webhooks, applications are more efficient.
 - + Webhooks are also known as reverse APIs, because applications subscribe to a webhook server by registering with the webhook provider.
 - + Multiple applications can subscribe to a single webhook server.

In order to receive a notification from a webhook provider, the application must meet certain requirements:

- + The application must be always running to receive HTTP POST requests.
- + application must register a URI on the webhook provider.
- + Also, the application must handle the incoming notifications from the webhook server.

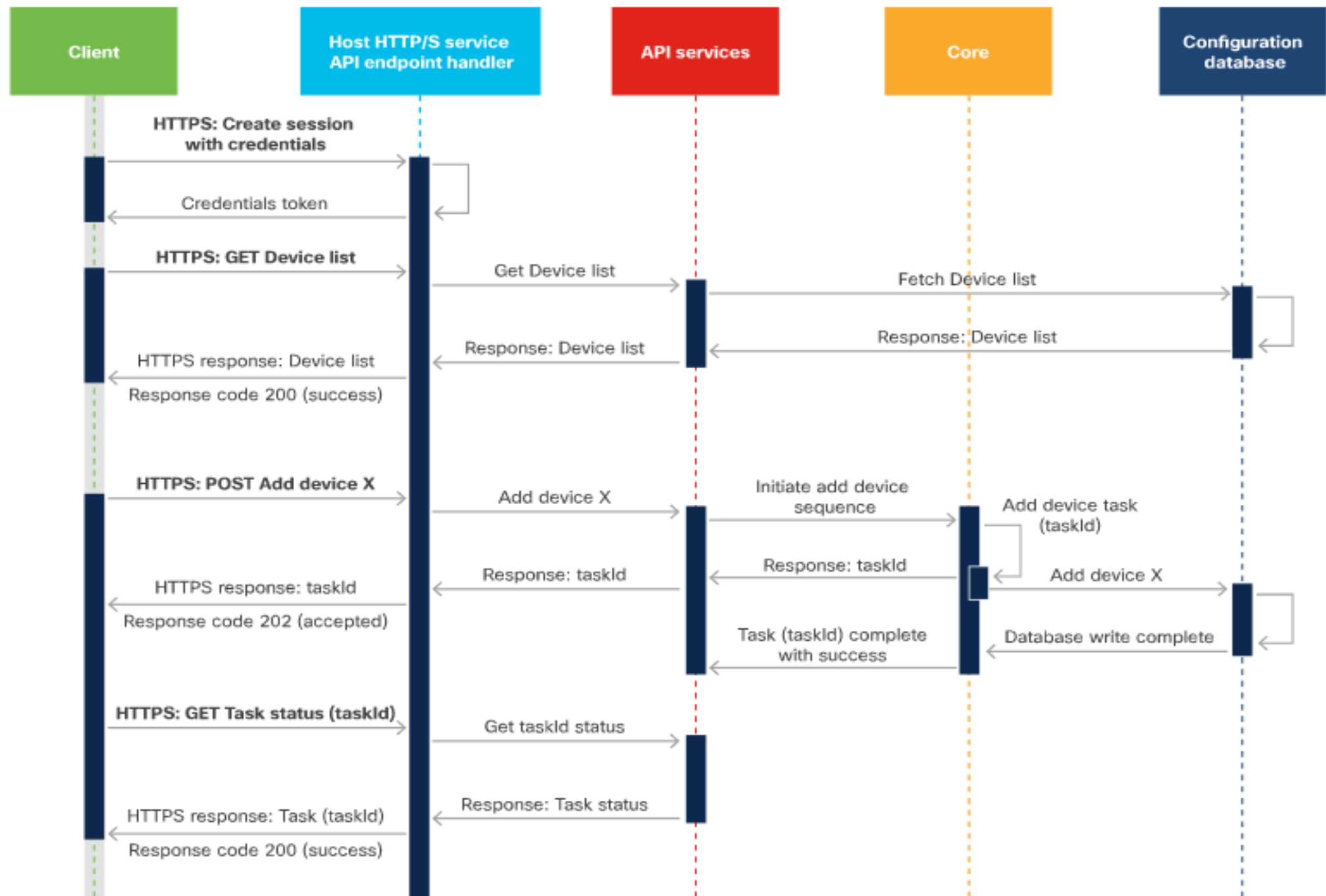


Sequence Diagram with REST API

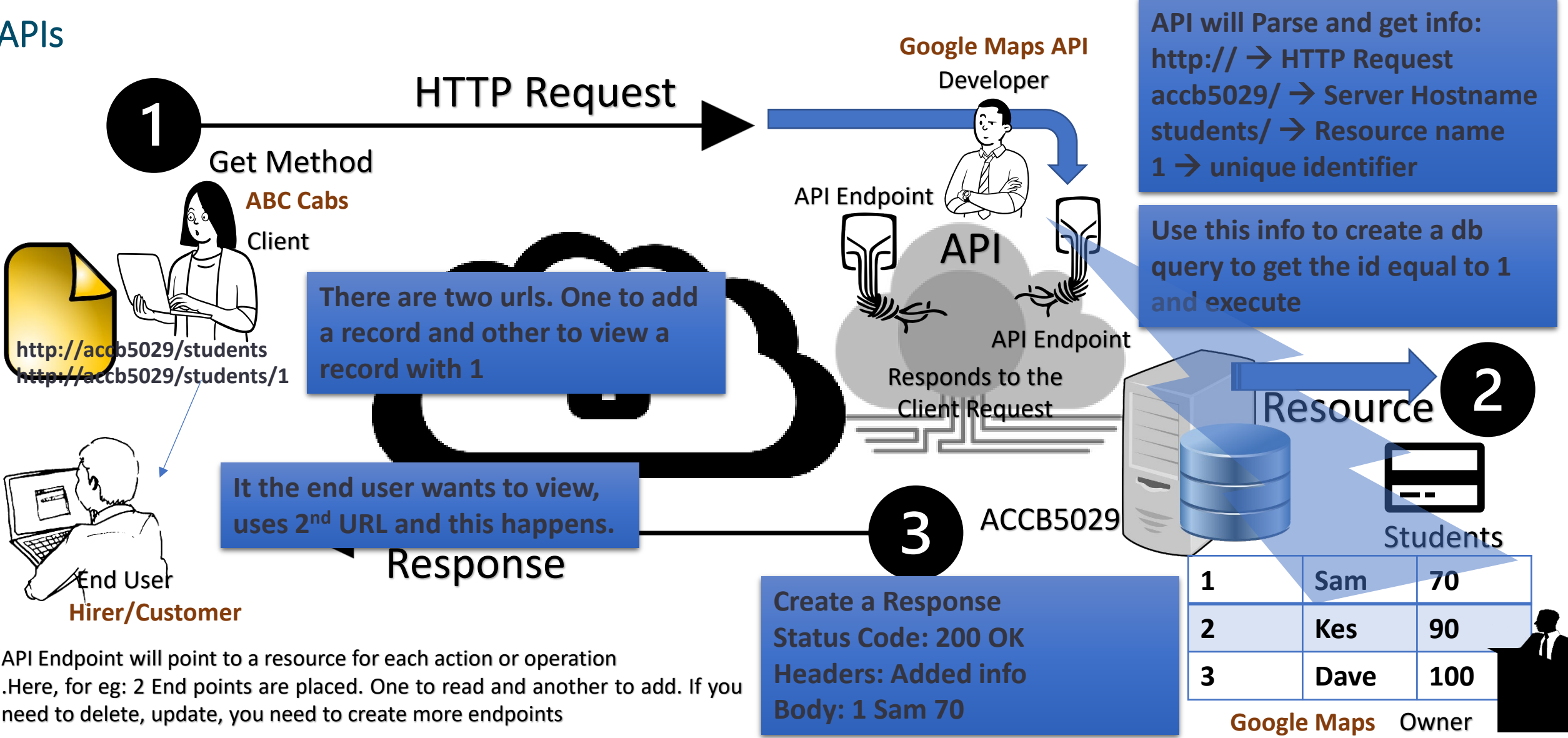
Create session
the starting request is labeled as HTTPS: Create Session w/credentials.

Get devices
request a list of devices from the platform.

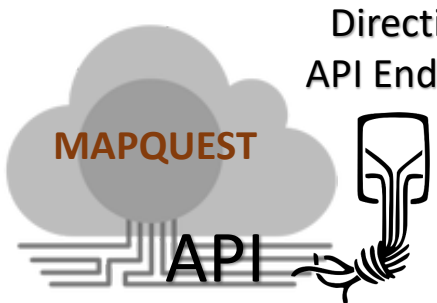
Create device
begins with a POST request to create a device.



APIs



Using APIs



Directions
API Endpoint

Mr10jQeogjS7dRHSL8up8XXX

- Identity of the user/ application to make a request
- Authorize the user/ application to make a request
- Account the user the level of access



Mapquest_dir.py

```
import urllib.parse
import requests
main_api = "https://www.mapquestapi.com/directions/v2/route?"
orig = "Swansea, UK"
dest = "London, UK"
key = "Mr10jQeogjS7dRHSL8up8CywI3MT4MU6"
url = main_api + urllib.parse.urlencode({"key":key, "from":orig, "to":dest})
json data = requests.get(url).json()
```

1 Get a MapQuest API Key
HTTP Request

<https://www.mapquestapi.com/directions/v2/route?key=Mr10jQeogjS7dRHSL8up8CywI3MT4MU6&from=Swansea%2C+UK&to=London%2C+UK>

details for scope, filtering, or to clarify a request

Query

<https://www.mapquestapi.com/directions/v2/route?key=Mr10jQeogjS7dRHSL8up8CywI3MT4MU6&from=Swansea%2C+UK&to=London%2C+UK>

protocol used to access
the resource

hostname of the server
hosting the API

path to the endpoint for
retrieving user data

user ID to retrieve

3

Response

```
{'lat': 51.6231, 'lng': -3.94086}, 'mapUrl': ''}, {'st  
ea5Type': 'City', 'adminArea4': 'London', 'adminArea4Ty  
Type': 'Country', 'postalCode': '', 'geocodeQualityCode  
'unknownInput': '', 'type': 's', 'latLng': {'lat': 51.5  
locationSequence': [0, 1]}, 'info': {'statusCode': 0,
```

Using APIs

```
{ 'lat': 51.6231, 'lng': -3.94086, 'mapUrl': '' }, { 'sta  
ea5Type': 'City', 'adminArea4': 'London', 'adminArea4Ty  
Type': 'Country', 'postalCode': '', 'geocodeQualityCode  
'unknownInput': '', 'type': 's', 'latLng': { 'lat': 51.5  
locationSequence': [0, 1] }, 'info': { 'statusCode': 0,
```

```
- info: {  
  statuscode: 0,  
  - copyright: {  
    text: "© 2022 MapQuest, Inc.",  
    imageUrl: "http://api.mqcdn.com/res/mqlogo.gif",  
    imageAltText: "© 2022 MapQuest, Inc."  
  },  
  messages: [ ]  
}
```

Response : Json data format

```
{  
  + route: { ... },  
  + info: { ... }  
}
```

API Status: 0

Request

Response

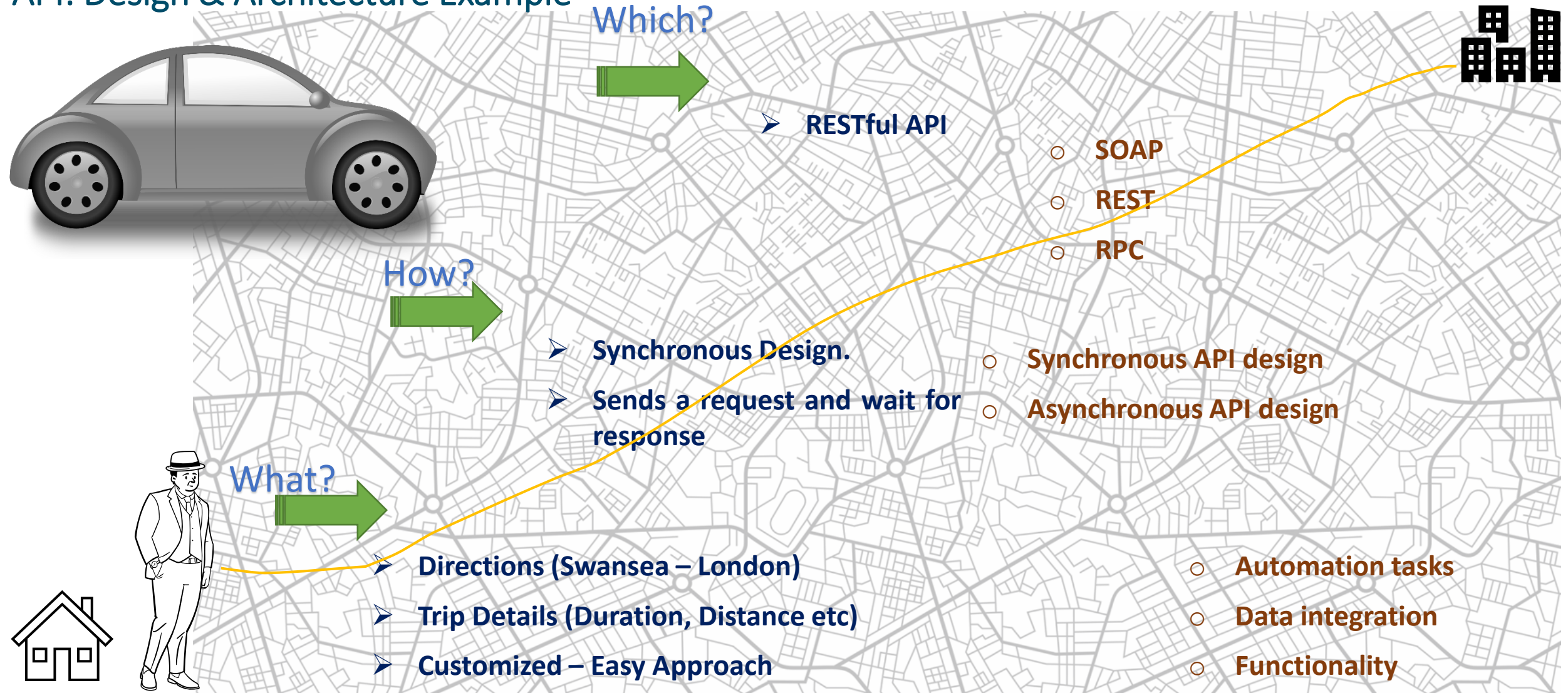
4

```
json_status = json_data["info"]["statusCode"]  
  
if json_status == 0:  
    print("API Status: " + str(json_status) + " = A successful route call.\n")
```

Status code 0 indicates that the API call was successful

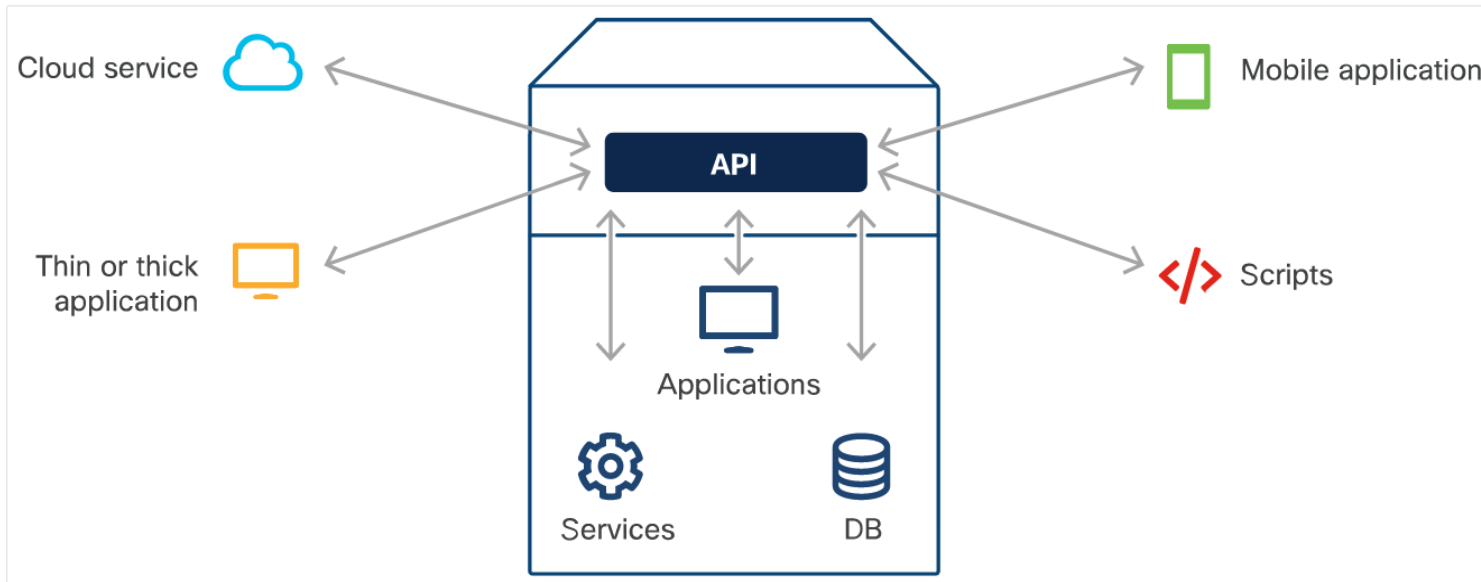
HTTP Status Code	Status Message	Description
400	Bad Request	Request will not be processed due to an error with the request
401	Unauthorized	Request does not have valid authentication credentials to perform the request
403	Forbidden	Request was understood but has been rejected by the server
404	Not Found	Request cannot be fulfilled because the resource path of the request was not found on the server

API: Design & Architecture Example



API

- An **Application Programming Interface (API)** allows **one piece of software talk to another**.
- It uses common web-based interactions or communication protocols and its own proprietary standards.
- An **API determines** what **type of data, services, and functionality** the application exposes to third parties.
- By providing APIs, applications can control what they expose in a secure way.



- APIs have existed for decades, but **exposure and consumption of APIs has grown exponentially** in the last 10 years or so.
- Most modern APIs are designed into the product and are thoroughly tested.
- Simplified **coding languages such as Python** have made it possible for **non-software engineers** to **build applications and consume APIs**.

Feedback/ Any Questions?

Module Expectations and Daily Feedback - ACCB5029

Tue, Oct 07, 2025

<https://vevox.app/#/m/136436531>

Session ID: 136-436-531

