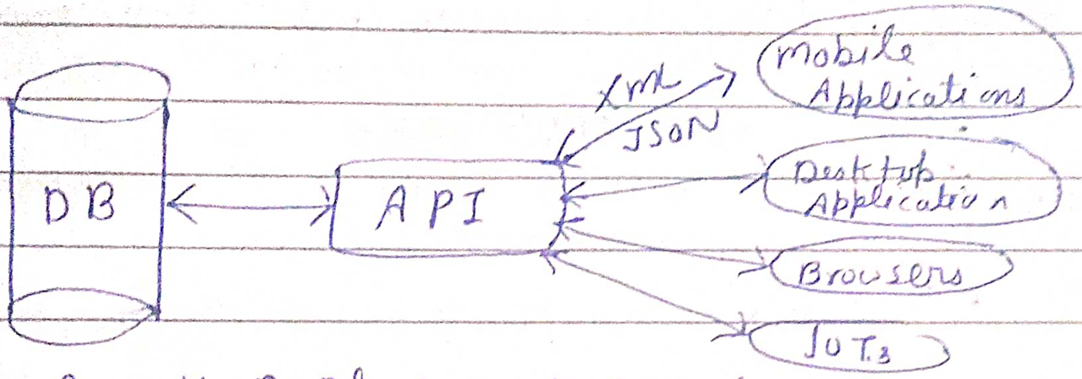
Application Programming Interface

★ API - defines the rules that we must follow to communicate with other software system.
↳ used to communicate betⁿ software.



~~Ex- Pay with PayPal, Login with Facebook, Login with Twitter/Google etc~~

★ REST
↳ Representational State Transfer
↳ ~~Architecture style / pattern~~
↳ Introduced by Roy Fielding in 2000.
↳ It's a software architecture that imposes conditions on how an API should work

Note- API that follow the REST architectural style are called REST API.
→ An API that uses web technology.
The web API that follows the REST architectural style, are called RESTful API.

↳ ~~REST uses various representations to represent a resource~~ like Text, JSON & XML. JSON is ~~no~~ the most ~~[illegible]~~ popular format being used in web services
↳ Here each resource is identified by URI/Global IDs.

★ Principles of REST architecture

1. Client server → uses HTTP protocol to communicate.
2. Stateless → does not make session
3. Cacheable → client can store received information.
4. Uniform Interface → indicate that server transfers info. in a standard format.
   - Request should identify resources → By URI. (uniform resource identifier)
   - Client have enough info. in the resource representation for modification if they want. ← required by sending metadata, that describes resource further
   - Client receive info. how to process the representation further

★ How do RESTful API work?

The client contact the server by using the API when it requires a resources. → It can be image, text, data etc.

- API developers explain how the client should use the REST API in the server application API documentation
- These are the general steps for any REST api call -
  - Client send a request → The client follows the API doc to format the request.
  - Server authenticates the clients.
  - Server receive request & process it internally.
  - The server returns a response to clients.

☆ What does the RESTful API client request contain.
- Unique Resource Identifier
- Methods / Verbs → GET, POST, PUT, DELETE
- HTTP headers — Request headers are the metadata exchanged betⁿ the client & server. For ex – the req header indicates th format of req. & response, provides info. about req status & so on.
- Data { Rest API req might include data for the POST, PUT }
- Parameters
  - Path parameters
  - Query string

☆ What are RESTful API authentication methods?
- HTTP authentication
  - Basic Auth. → client sends the username & password in the req. header. It encodes them with base64.
  - Bearer Auth. → refers to the process of giving access control to token bearer. The bearer token is typically an encrypted string of characters that server generates in response to login request.
- API keys → In this, the server assigns a unique generated value to a first-time client, whenever the client tries to access resource, it uses the unique API key to verify itself.
- OAuth → combines passwords & tokens for highly secure login access to any system. The server first req a password and then ask for additional token to complete the authorization process.

* What does the RESTful API server response contain?

- Status codes → that represent the req. is success/ failure
  - 2xx – indicates success (200, 201)
    - 201 ↳ POST succe
  - 3xx – indicate url redirection
  - 4xx & 5xx indicate errors. (400, 404)
    - 400 → Incorrect req./Bad req.
    - 404 ↳ Resour not fou
- Message Body
  ↳ contains the resource representation
- Headers
  ↳ metadata about response
  ↳ give more context about the response.

★ ASP .NET Web API

- ↳ allows users to access a particular resource using HTTP protocol.
- ↳ We can build WebAPI using different technologies like .NET, PHP, Java, Python, etc.
- ↳ In .NET, Microsoft has created a framework for Web API called ASP.NET WebAPI.
- ↳ ASP .NET Web API is a framework that makes it easy to build HTTP web services (Restful HTTP services)
- ↳ Provided by .NET framework.
- ↳ Very similar to ASP .NET MVC since it contains the MVC features
- ↳ Web API is often used to provide an interface for websites & client applications to have data access
- ↳ Web API can be used to access data from a database & save data back to the database

Ex- ~~Uniform~~ Interface in REST

Resource	VERB	Result
/Students	GET	Get list of students
/Students/1	GET	Get student with id=1
/Students	POST	Create a new student
/Students/1	PUT	Update student with id=1
/Students/1	DELETE	Delete student with id=1
/Students/1	Patch	Update student with id=1

Note-

delete old one & create new one, so we have to send all attributes

Put - Completely replaces an existing resource otherwise created new one. In it, we have to send all the attributes of data

Patch - Update only necessary attributes of an resource
Allows the client to send only the changes attribute rather than whole entity.

	ASP.NET MVC	ASP.NET Web API
①	returns both data & views	returns only data. ↳ in various formats such as JSON/XML & other based on the request header.
②	MVC returns the data in the JSON format by using JSONResult.	
③		The Web API helps the creation of RESTful services over the .NET framework but MVC does not support.
④	In it, the request is mapped to actions based on action name & controller name	In Web API, the request is mapped to the actions based on HTTP verbs & controller name
⑤	It does not support.	It supports convention based crud action.
⑥	We can manage the state of data by making session	We can't do this it ASP.NET WebAPI because REST by design is stateless.
⑦		By adding session to WebAPI, we are making it stateful & defeating any purpose of having a RESTful API

* how to create ASP.NET Web API ?
  - By using WebAPI template
  - By using Empty template with MVC & WebAPI check box
  - Each controller in Web API inherits the ApiController Class.
  - Each controller in ASP.NET Web API has 'Controller' word in their naming.
  - Call to action method is done via controller name & HTTP verbs.
  - Returns types of action method in API is IHttpActionResult.
  - In it, Action name can be anything. By default, it should be naming as Http verbs functionality.
  - Action method can return Ok(); , NotFound(), etc.

Ex-
```
public class StudentsController : ApiController
{
    StudentsEntities db = new StudentsEntities();
    [HttpGet]
    public IHttpActionResult Get()
    {
        List<students> ls = db.Students.ToList();
        return Ok(ls);
    }

    [HttpGet]
    public IHttpActionResult Get(int id)
    {
        students s = db.Students.where(m=>m.id==id).FirstOrDefault();
        return Ok(s);
    }
    [HttpPost]
    public IHttpActionResult Post(Students s)
    {

    }
```

/api/Students $\xrightarrow{\text{Get}}$ Get()

/api/Students/5 $\xrightarrow{\text{Get}}$ Get(int id)

/api/Students $\xrightarrow{\text{Post}}$ Post(Students s)

★ What is Ok() in Web API?
Web API writes the serialized value into the response body.
The response status code is 200 (OK).

★ Postman tool
↳ very useful in API testing.
↳ It's a Http client that is used to tests the HTTP requests.
↳ we can utilize API in GUI
↳ By using Postman, we can obtain different types of responses comes from WebAPI
↳ By using it, we create better APIs & test it faster
↳ This tool allows us to design, test, debug, automated testing, document, monitor & publish the APIs
↳ also called API development Platform.

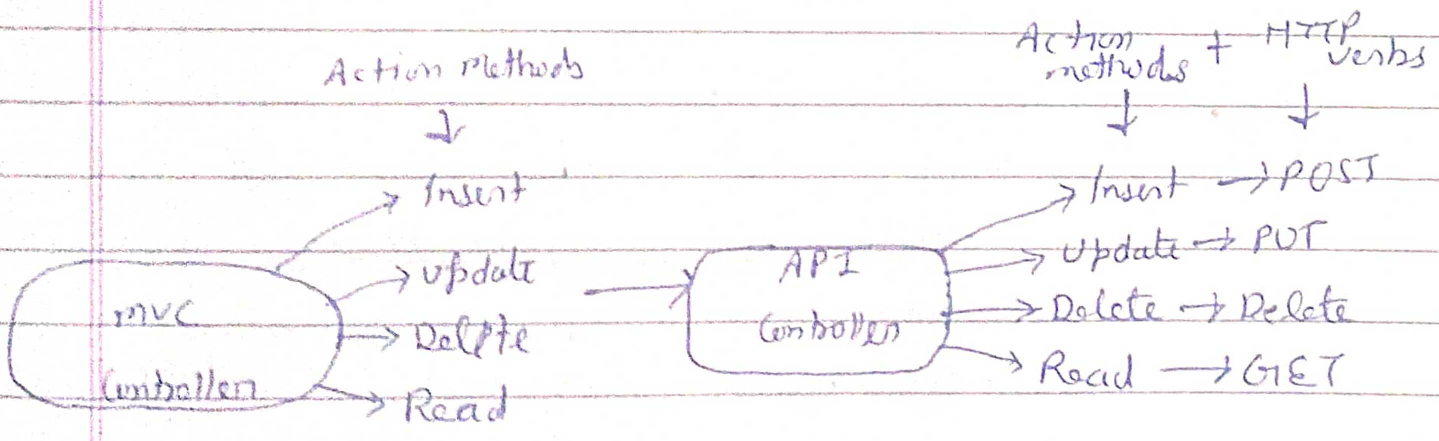
★ Consuming ASP.NET WebAPI in ASP.NET MVC.

$$\begin{tikzcd}
\text{ASP.NET WebAPI} \arrow[r, leftrightarrow] \arrow[d] & \text{SQL Server DB.} \\
\text{http://localhost:45294/api/} \arrow[d] & \\
\text{Consume} \arrow[r] \arrow[d] & \text{To consume, Add References in ASP.NET MVC application, System.NET.Http, Microsoft.AspNET.Webapi.Client} \\
\text{ASP.NET MVC} \arrow[d] & \\
\text{HTML Table.} &
\end{tikzcd}$$

★ Http Client
↳ used to send requests & retrive their responses.

Ex
```
[HttpGet]
public IActionResult Index()
{
  HttpClient HC = new HttpClient();
  HC.BaseAddress = new Uri("http://localhost:49594/api/Students");
  var res = HC.GetAsync(HC.BaseAddress);
  res.Wait();
  if (res.Result.IsSuccessStatusCode)
  {
       var
   List<Students> ls = res.Result.Content.ReadAsAsync<List<Student>>().
                                                               Result;
   return View(ls);
  }
   return View(null);
}
```

* Diagram to show functionality of an app with API



★ Insert data to DB by consuming WebAPI

API code

```
public IHttpActionResult Post(Students s)
{
    db.Students.Add(s);
    db.SaveChanges();
    return Ok();
}
```

```
public IActionResult Insert(Students s)
{
HttpClient HC = new HttpClient();
HC.BaseAddress = new
    Uri("https://localhost:44542/api/students");
                     Json
var res = HC.PostAsAsync<Students>
          (HC.BaseAddress, s);
           "Students"       ↳ object of students
              ↳ can also be written this in place of HC.BaseAddress

res.Wait();
if (res.Result.IsSuccessStatusCode)
   ViewBag.msg = "Saved Successfully";
else
   ViewBag.msg = "Failed to Save";
return View();
}
```

★ Note:- These all code works same for other Http verbs just with a little change of method in HttpClient object while requesting

★ Methods of HttpClient object for different Http verbs. -

```
GetAsync
                        → Generic Name for class
PostAsJsonAsync<        >(        ,        )
PutAsJsonAsync<         >(        ,        )
                             ↳ Url     ↳ obj to Post/Put
DeleteAsync
```

Negotiate means to bargain/ to discuss

* Content Negotiation

↳ The process of selecting the best representation for a given response when there are multiple representation available..

↳ One of the standards of the REST service is that client should have the ability to decide in which format they want the response whether in XML/JSON etc., called

↳ Two main headers, responsible of content negotiation.

$$\begin{tikzcd}
\text{Content-Type} & \text{in response} \arrow[l] \arrow[r] & \text{XML/JSON} \\
\text{Accept} \arrow[d] & \text{in request} \arrow[l] \arrow[r] & \text{XML/JSON} \\
\text{Ex- Accept: application/xml} & &
\end{tikzcd}$$

Accept: application/json.