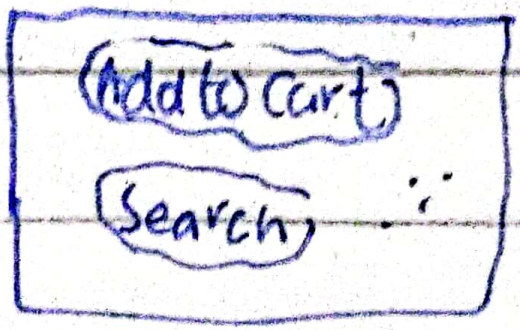
→ MicroServices Architecture :

⊙ Each application has different services / responsibilities and it is single codeways



: 12 cores, 8 GB

(we need to update system)

⊙ Monolithic Architecture so it grows and bottleneck occurs.

→ So we make responsibilities separate and make the services small (only one responsibility will be provided), it is called microservices. (Independent Projects)

Can Scale Independently (Up to down)
(Resources will be distributed.

→ Services will communicate with each other:

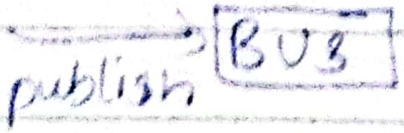
(i) Synchronous (Directly Access by Api) (ii) Asynchronous

→ Each microservice is implemented as api and access it via URL.

→ Use HttpClient and class reciever service and call it and data is fetched.

→ Asynchronous System:

⊙ Message Bus → KAFKA, RabitMQ



→ Many databases of services, so same data is present for same product; so we make database eventually consistent.

→ If any change is there, then publish it on bus

→ Topic in Bus. e.g: Product Name/Details

→ Overhead of Microservices:

(i) Error handling.

(ii) Many manageable users.

→ [Mass Transient] → Library.

→ OData (Ch#10): [Enable Query]

→ Query options: Expand
→ Operators: Less than, greater than etc....

⊙ Add, Update, Delete OData:

: Get Already Done

Post → public IActionResult Post
([FromBody] product)
{
product.Id = Products.Max(p => p.Id)+1;
Products.Add(product);
return Created(product);
}

Put → public IActionResult Put

Delete → " " Delete.

For use previous and Now concept (mix)

⊙
[HttpGet("/expensive products")]
(we can call specific functions also for it).
(Generic and specific).

⊙ Local Storage :

→ Inspect :
cache, Cookies, Indexed DB, Local Storage

→ Temporary wanted to store data on browser.

→ Key - Value pair data.

→ Save (SetItem), Get (GetItem)