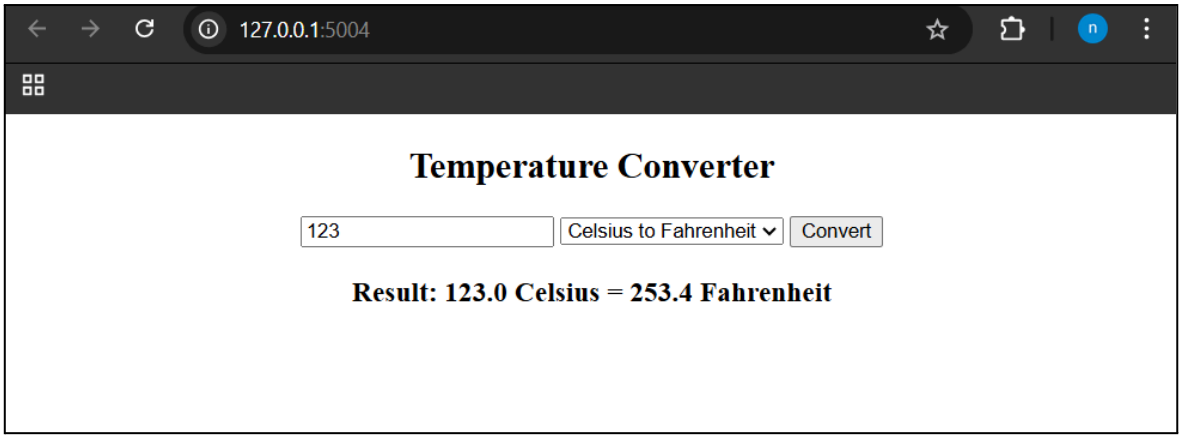


*Хэрэглэгчтэй харьцах interface^^*

Service	Port	Roles
soap_service.py	8000	SOAP service: Хэрэглэгчээс ирсэн хүсэлтэд температурын хөрвүүлэлт хийж өгнө.
ctof.py	5001	REST API: Celsius-ийг Fahrenheit руу хөрвүүлж өгнө.
ftoc.py	5002	REST API: Fahrenheit-ийг Celsius руу хөрвүүлж өгнө.
gateway.py	5003	API Gateway: Хэрэглэгчээс ирсэн хүсэлтийг шалгаж, тохирох REST API руу дамжуулан, хариуг буцаана.
client.py	5004	Client interface: Хэрэглэгчээс оруулсан өгөгдлийг gateway.py руу илгээнэ.

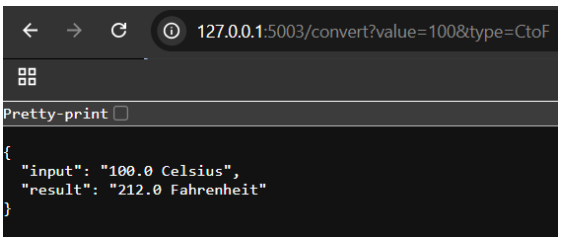
Дараалал:

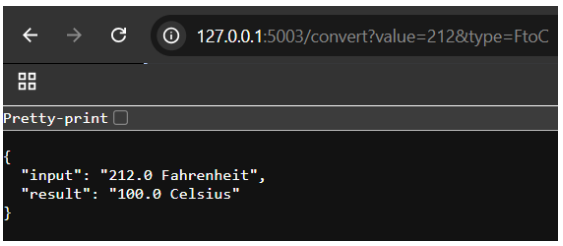
- Client (index.html) хүсэлт үүсгэж, gateway.py (5003) руу илгээнэ.
  - Жишээ: http://127.0.0.1:5003/convert?value=100&type=CtoF
  - value=100 (Хөрвүүлэх утга)
  - type=CtoF (Хөрвүүлэх төрөл)

```
 * Running on http://127.0.0.1:5003
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 283-284-821
127.0.0.1 - - [12/Feb/2025 01:45:34] "GET /convert?value=123&type=CtoF HTTP/1.1" 200 -
127.0.0.1 - - [12/Feb/2025 01:46:18] "GET /convert?value=123&type=FtoC HTTP/1.1" 200 -
```

- Gateway (gateway.py, API Gateway) хүсэлтийг хүлээн авч, тохирох REST API руу дамжуулна.
  - type=CtoF => http://127.0.0.1:5001 / (ctof.py)
  - type=FtoC => http://127.0.0.1:5002 / (ftoc.py)

- Хөрвүүлэлтийн API (ctof.py эсвэл ftoc.py) тооцоолол хийж, хариу JSON буцаана.
    - Жишээ: { "input": "100°C", "result": "212°F" }





- Gateway (gateway.py) уг хариуг буцааж, client-д дамжуулна.
- Client (index.html) хөрвүүлсэн утгыг харуулна.

Асуулт:

5. Create a diagram of how the communication between the services and the interfaces are flowing.

6. Answer the following:

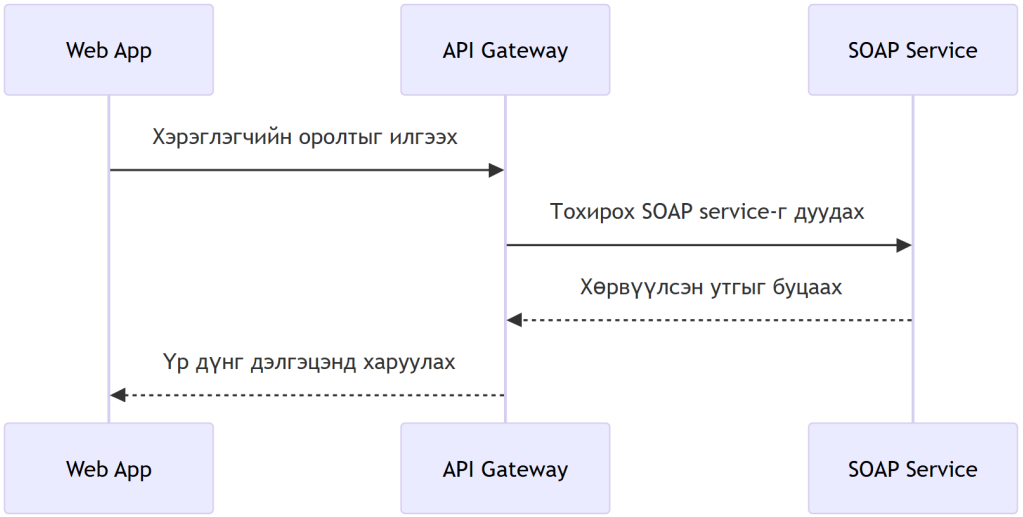
* What is the difference between a Microservice architecture and a Service Oreinted Architecture?

* Define the contract between the the service interactions

* How is SOA in the tasks above improve reusability, interoperability, scalability, and loose coupling.

Хариулт:

5.



6.

- Microservice architecture болон SOA нь хоёулаа программ хангамжийг олон жижиг сервисүүдэд хувааж, хооронд нь холбож ажиллуулах зорилготой. Гэхдээ тэдний ажиллах зарчим өөр.

  SOA нь ихэвчлэн том байгууллагуудын системд ашиглагддаг. Бүх сервисүүд нь нэг гол төв (ESB) рүү холбогдож, түүгээр дамжин харилцдаг. Энэ нь бүх сервисүүдийг нэг дор удирдах боломжтой болгож, уялдаа холбоог сайжруулдаг. Гэхдээ нэг газар асуудал гарвал бүх системд нөлөөлж болдог.

  Microservice architecture нь харин сервисүүдийг илүү жижиг, бие даасан байдлаар ажиллуулдаг. Сервисүүд тус бүр өөрийн өгөгдлийн сантай бөгөөд API ашиглан хоорондоо харилцдаг. Энэ нь уян хатан бөгөөд нэг сервист өөрчлөлт оруулахад бусад нь хэвийн ажиллаж үлддэг.

  Мөн SOA нь SOAP, XML гэх мэт технологи ашигладаг бол microservice нь REST, JSON, gRPC гэх мэт хурдан, хөнгөн технологи хэрэглэдэг. Энэ нь микросервисийг хөгжүүлэх, сайжруулахад илүү амар болгодог.

- Сервис хоорондын харилцан үйлчлэлийн гэрээ нь тухайн сервис яаж ажиллахыг тодорхойлсон дүрэм, зааврыг багтаасан баримт бичиг. Энэ нь хөгжүүлэгчдэд тухайн сервисийг хэрхэн ашиглах талаар мэдээлэл өгч, сервисүүдийг хооронд нь зөв холбох боломжтой болгодог.

  Ихэвчлэн сервисийн интерфейс, өгөгдлийн бүтэц, ашиглах арга, протокол, шаардлагатай параметрүүдийн мэдээлэл багтана.

  Энэ нь сервис хэрэглэгчид тухайн сервисийг ойлгоход тусалж, системийн уялдаа, найдвартай ажиллагааг сайжруулахад тус болдог.

-

Reusability	Нэгэнт хөгжүүлсэн сервис олон системд ашиглагдаж болно.
Interoperability	SOAP, REST, gRPC зэрэг стандарт протоколууд ашигласнаар өөр өөр технологи бүхий системүүд хоорондоо ажиллах боломжтой.
Scalability	SOA нь шинэ сервис нэмж өргөтгөхөд хялбар байдаг.
Loose Coupling	Сервисүүд бие даасан байдлаар ажиллаж, нэг сервист асуудал үүсэх үед бусдададаа нөлөөлөхгүй.