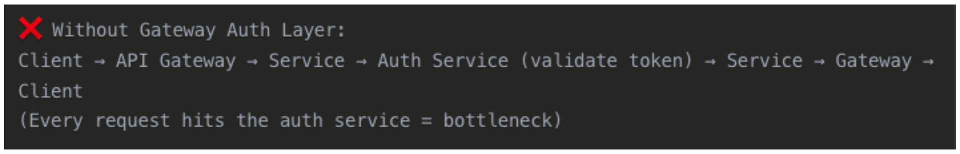
Use Case 3: API Gateway Service

Problem Statement:

Client requests must go through the Authentication Service every single time for token validation, and API routings are not centrally managed. This generates unnecessary service calls and results in bottleneck.

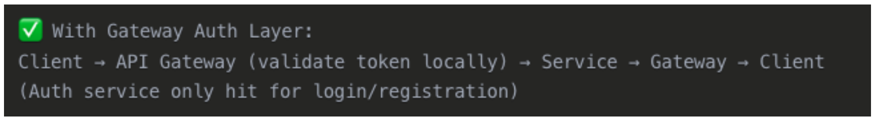


```
❌ Without Gateway Auth Layer:
Client → API Gateway → Service → Auth Service (validate token) → Service → Gateway →
Client
(Every request hits the auth service = bottleneck)
```

Solution:

Add an **API Gateway Service layer** to separate the roles from the Authentication Service and manage the routings in a centralized way. The Authentication Service takes the responsibility of **User Registration & Login**, **Token Generation, User Management, Password Management, Token Refresh, and User Roles,** etc. The API Gateway Service, on the other hand, manages **Token Validation, Route Protection, Role-based Routing, Performance, and Security Boundary**. Such architecture has the benefit of:

1. **Performance**: No auth service bottleneck for every request
2. **Scalability**: Gateway can handle thousands of token validations per second
3. **Security**: Single point of entry with consistent security policies
4. **Reliability**: Services remain accessible even if auth service is temporarily down (for existing valid tokens)
5. **Separation of Concerns**: Clear distinction between identity management and request authorization



```
✅ With Gateway Auth Layer:
Client → API Gateway (validate token locally) → Service → Gateway → Client
(Auth service only hit for login/registration)
```

General Architecture:

```
backend/apiGateway/
├── src/main/java/com/foodopia/apiGateway/
│   ├── ApiGatewayApplication.java
│   │
│   ├── config/
│   │
│   ├── filter/
│   │   ├── pre/
│   │   │
│   │   ├── post/
│   │   │
│   │   └── global/
│   │
│   ├── security/
│   │
│   ├── service/
│   │
│   ├── dto/
│   │
│   ├── exception/
│   │
│   ├── util/
│   │
│   └── monitoring/
│
├── src/main/resources/
│
└── pom.xml                                 # Dependencies
```

code language