| **Layer** | **Technology** |
| --- | --- |
| **Microservices Framework** | Spring Boot |
| **Service Discovery** | Spring Cloud Netflix Eureka or Consul |
| **API Gateway** | Spring Cloud Gateway or Zuul |
| **Configuration Management** | Spring Cloud Config |
| **Inter-Service Communication** | REST (Feign Client) or gRPC |
| **Database** | PostgreSQL / MySQL / MongoDB |
| **Caching** | Redis |
| **Message Broker** | RabbitMQ / Apache Kafka |
| **Authentication** | Spring Security + JWT / OAuth2 |
| **Containerization** | Docker |
| **Orchestration** | Kubernetes (K8s) |
| **Monitoring** | Prometheus + Grafana, ELK Stack |
| **CI/CD** | Jenkins / GitHub Actions / GitLab CI |

1. **User Service** – Handles registration, login, profile
2. **Product/Content Service** – Manages core business data
3. **Order/Transaction Service** – Handles purchases or actions
4. **Notification Service** – Email/SMS/Push notifications
5. **API Gateway** – Entry point for all client requests
6. **Config Server** – Centralized configuration
7. **Discovery Server** – Service registry
8. Use **asynchronous processing** (e.g., @Async, messaging queues)
9. Implement **circuit breakers** (e.g., Resilience4j)
10. Enable **rate limiting** and **throttling**
11. Use **load balancers** (e.g., NGINX, Kubernetes Ingress)
12. Apply **database sharding** or **replication** for scale
13. **Unit Testing**: JUnit, Mockito
14. **Integration Testing**: TestContainers, Spring Boot Test
15. **Load Testing**: JMeter, k6, Gatling
16. **Monitoring**: Actuator endpoints, Prometheus, Grafana